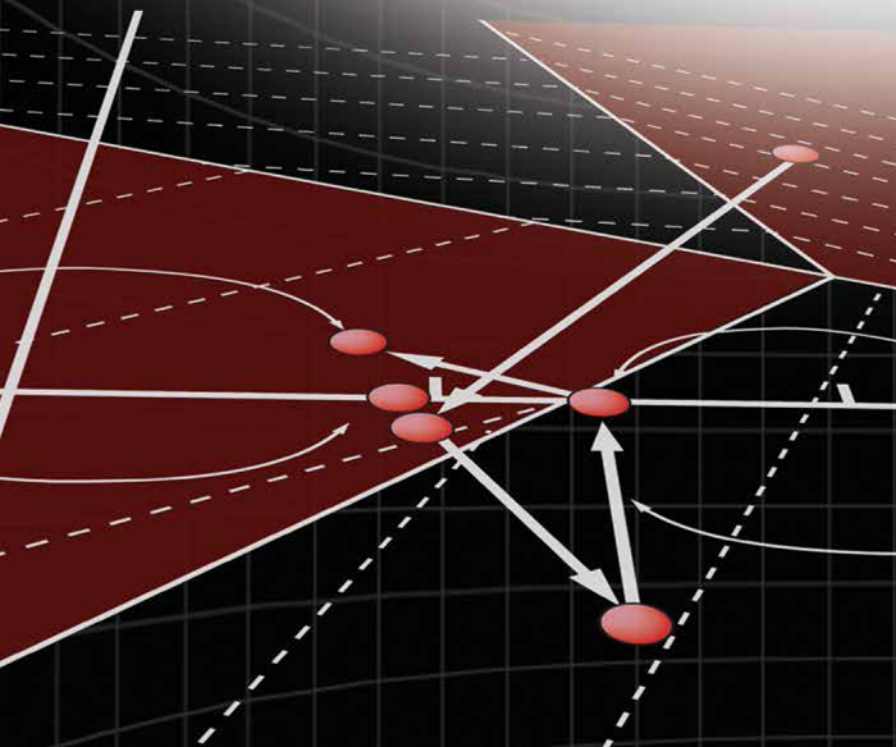


RONALD L. RARDIN

OPTIMIZATION IN OPERATIONS RESEARCH

SECOND EDITION



PRIMERS

1	Vectors	89
2	Derivatives and Partial Derivatives	110
3	Extreme Points and Directions of Polyhedral Sets	203
4	Matrices, Matrix Arithmetic, and Transposes	213
5	Simultaneous Equations, Singularity, and Bases	223
6	Identity and Inverse Matrices	261
7	Second Derivatives and Hessian Matrices	938
8	Positive and Negative (Semi) Definite Matrices	945

ALGORITHMS

3A	Continuous Improving Search	106
3B	Two-Phase Improving Search	129
5A	Rudimentary Simplex Search for Linear Programs	235
5B	Two-Phase Simplex Search	247
5C	Revised Simplex Search for Linear Programs	271
5D	Lower- and Upper-Bounded Revised Simplex	278
6A	Dual Simplex Search for Linear Programs	363
6B	Primal-Dual Simplex Search for Linear Programs	369
7A	Affine Scaling Search for Linear Programs	407
7B	Newton Step Barrier Search for Linear Programs	419
7C	Primal-Dual Interior-Point LP Search	424
9A	One to All (No Negative Dicycles); Bellman–Ford Shortest Paths	496
9B	All-to-All (No Negative Dicycles); Floyd–Warshall Shortest Paths	502
9C	One to All (Nonnegative Costs); Dijkstra Shortest Paths	509
9D	Shortest One to All Paths (Acyclic Digraph) Shortest Paths	518

9E	CPM Early Start Scheduling	525
10A	Rudimentary Cycle Direction Network Search	582
10B	Cycle Cancelling for Network Flows	587
10C	Network Simplex Search	599
10D	Hungarian Algorithm for Linear Assignment	612
10E	Maxflow-Mincut Search	622
10F	Greedy Search for a Min/Max Spanning Tree	634
12A	LP-Based Branch and Bound (0-1 ILPs)	761
12B	Branch and Cut (0-1 ILP's)	779
13A	Delayed Column Generation	816
13B	Branch and Price Search (0-1 ILPs)	820
13C	Subgradient Lagrangian Search	835
13D	Dantzig–Wolfe Decomposition	841
13E	Benders Decomposition	846
15A	Rudimentary Constructive Search	880
15B	Discrete Improving Search	887
15C	Tabu Search	895
15D	Simulated Annealing Search	898
15E	Genetic Algorithm Search	904
16A	Golden Section Search	927
16B	Three-Point Pattern	931
16C	Quadratic Fit Search	934
16D	Gradient Search	955
16E	Newton's Method	962
16F	BFGS Quasi-Newton Search	968
16G	Nelder–Mead Derivative-Free Search	974
17A	Sequential Unconstrained Penalty Technique (SUMT)	1034
17B	Sequential Unconstrained Barrier Technique	1038
17C	Reduced Gradient Search	1048
17D	Active Set Method for Quadratic Programs	1059
17E	Sequential Quadratic Programming (SQP)	1063

Optimization in Operations Research

This page intentionally left blank

Optimization in Operations Research

SECOND EDITION

RONALD L. RARDIN

University of Arkansas

PEARSON

Boston Columbus Indianapolis Hoboken New York San Francisco Amsterdam
Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial
Director, ECS: Marcia J. Horton
Executive Editor: Holly Stark
Editorial Assistant: Amanda Brands
Field Marketing Manager: Demetrius Hall
Senior Product Marketing Manager:
Bram van Kempen
Marketing Assistant: Jon Bryant
Senior Managing Editor: Scott Disanno
Program Manager: Erin Ault
Production Project Manager: Greg Dulles

Director of Operations: Nick Sklitsis
Operations Specialist: Maura Zaldivar-Garcia
Cover Designer: Black Horse Designs
Manager, Rights and Permissions:
Rachel Youdelman
Associate Project Manager, Rights and
Permissions: William Opaluch
Composition: Integra Software Services, Inc.
Printer/Binder: RR Donnelley/Crawfordsville
Cover Printer: Phoenix Color/Hagerstown
Typeface: 10/12 Times Ten LT Std

Copyright © 2017, 1998 by Pearson Higher Education, Inc., Hoboken, NJ 07030.

All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright and permissions should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearsoned.com/permissions/.

Many of the designations by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Library of Congress Cataloging-in-Publication Data

Rardin, Ronald L.

Optimization in operations research / Ronald L. Rardin, Purdue University.—Second edition.
pages cm

Includes bibliographical references and index.

ISBN 978-0-13-438455-9—ISBN 0-13-438455-5

1. Operations research. 2. Mathematical optimization. 3. Programming (Mathematics) I. Title.

T57.7.R37 2016

519.7'2—dc23

2015019627

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN 10: 0-13-438455-5

ISBN 13: 978-0-13-438455-9

Contents

PREFACE *xxix*

ABOUT THE AUTHOR *xxxii*

CHAPTER 1 *PROBLEM SOLVING WITH MATHEMATICAL MODELS* **1**

- 1.1 OR Application Stories 1
 - 1.2 Optimization and the Operations Research Process 3
 - Decisions, Constraints, and Objectives* 4
 - Optimization and Mathematical Programming* 4
 - Constant-Rate Demand Assumption* 5
 - Back of Envelope Analysis* 5
 - Constant-Rate Demand Model* 7
 - Feasible and Optimal Solutions* 7
 - 1.3 System Boundaries, Sensitivity Analysis, Tractability, and Validity 9
 - EOQ Under Constant-Rate Demand* 9
 - System Boundaries and Sensitivity Analysis* 10
 - Closed-Form Solutions* 11
 - Tractability versus Validity* 11
 - 1.4 Descriptive Models and Simulation 12
 - Simulation over MM's History* 12
 - Simulation Model Validity* 12
 - Descriptive versus Prescriptive Models* 14
 - 1.5 Numerical Search and Exact Versus Heuristic Solutions 14
 - Numerical Search* 14
 - A Different Start* 15
 - Exact versus Heuristic Optimization* 16
 - 1.6 Deterministic Versus Stochastic Models 16
 - Random Variables and Realizations* 17
 - Stochastic Simulation* 17
 - Tradeoffs between Deterministic and Stochastic Models* 19
 - 1.7 Perspectives 19
 - Other Issues* 20
 - The Rest of This Book* 20
- Exercises** 20

CHAPTER 2 DETERMINISTIC OPTIMIZATION MODELS IN OPERATIONS RESEARCH 23

- 2.1 Decision Variables, Constraints, and Objective Functions 23
 - Decision Variables* 24
 - Variable-Type Constraints* 24
 - Main Constraints* 25
 - Objective Functions* 25
 - Standard Model* 26
- 2.2 Graphic Solution and Optimization Outcomes 27
 - Graphic Solution* 27
 - Feasible Sets* 27
 - Graphing Constraints and Feasible Sets* 27
 - Graphing Objective Functions* 30
 - Optimal Solutions* 33
 - Optimal Values* 34
 - Unique versus Alternative Optimal Solutions* 35
 - Infeasible Models* 36
 - Unbounded Models* 38
- 2.3 Large-Scale Optimization Models and Indexing 40
 - Indexing* 40
 - Indexed Decision Variables* 41
 - Indexed Symbolic Parameters* 42
 - Objective Functions* 43
 - Indexed Families of Constraints* 43
 - Pi Hybrids Application Model* 45
 - How Models Become Large* 46
- 2.4 Linear and Nonlinear Programs 46
 - General Mathematical Programming Format* 46
 - Right-Hand Sides* 47
 - Linear Functions* 48
 - Linear and Nonlinear Programs Defined* 50
 - Two Crude and Pi Hybrids Models are LPs* 51
 - Indexing, Parameters, and Decision Variables for E-mart* 51
 - Nonlinear Response* 51
 - E-mart Application Model* 52
- 2.5 Discrete or Integer Programs 53
 - Indexes and Parameters of the Bethlehem Application* 53
 - Discrete versus Continuous Decision Variables* 53
 - Constraints with Discrete Variables* 55
 - Bethlehem Ingot Mold Application Model* 56
 - Integer and Mixed-Integer Programs* 56
 - Integer Linear versus Integer Nonlinear Programs* 57
 - Indexing, Parameters, and Decision Variables for Purdue Finals Application* 59

	<i>Nonlinear Objective Function</i>	59
	<i>Purdue Final Exam Scheduling Application Model</i>	60
2.6	Multiobjective Optimization Models	60
	<i>Multiple Objectives</i>	61
	<i>Constraints of the DuPage Land Use Application</i>	62
	<i>DuPage Land Use Application Model</i>	63
	<i>Conflict among Objectives</i>	64
2.7	Classification Summary	65
2.8	Computer Solution and AMPL	65
	<i>Solvers versus Modeling Languages</i>	66
	<i>Indexing, Summations, and Symbolic Parameters</i>	67
	<i>Nonlinear and Integer Models</i>	70
	Exercises	73
	References	86

CHAPTER 3 IMPROVING SEARCH 87

3.1	Improving Search, Local, and Global Optima	87
	<i>Solutions</i>	88
	<i>Solutions as Vectors</i>	88
	<i>Example of an Improving Search</i>	93
	<i>Neighborhood Perspective</i>	94
	<i>Local Optima</i>	95
	<i>Local Optima and Improving Search</i>	95
	<i>Local versus Global Optima</i>	95
	<i>Dealing with Local Optima</i>	97
3.2	Search with Improving and Feasible Directions	98
	<i>Direction-Step Paradigm</i>	98
	<i>Improving Directions</i>	100
	<i>Feasible Directions</i>	102
	<i>Step Size: How Far?</i>	104
	<i>Search of the DClub Example</i>	105
	<i>When Improving Search Stops</i>	107
	<i>Detecting Unboundedness</i>	108
3.3	Algebraic Conditions for Improving and Feasible Directions	109
	<i>Gradients</i>	109
	<i>Gradient Conditions for Improving Directions</i>	112
	<i>Objective Function Gradients as Move Directions</i>	114
	<i>Active Constraints and Feasible Directions</i>	115
	<i>Linear Constraints</i>	117
	<i>Conditions for Feasible Directions with Linear Constraints</i>	118

3.4	Tractable Convex and Linear Cases	120
	<i>Special Tractability of Linear Objective Functions</i>	120
	<i>Constraints and Local Optima</i>	121
	<i>Convex Feasible Sets</i>	121
	<i>Algebraic Description of Line Segments</i>	123
	<i>Convenience of Convex Feasible Sets for Improving Search</i>	124
	<i>Global Optimality of Linear Objectives over Convex Feasible Sets</i>	125
	<i>Convexity of Linearly Constrained Feasible Sets</i>	126
	<i>Global Optimality of Improving Search for Linear Programs</i>	127
	<i>Blocking Constraints in Linear Programs</i>	127
3.5	Searching for Starting Feasible Solutions	129
	<i>Two-Phase Method</i>	129
	<i>Two Crude Model Application Revisited</i>	129
	<i>Artificial Variables</i>	130
	<i>Phase I Models</i>	130
	<i>Starting Artificial Solution</i>	131
	<i>Phase I Outcomes</i>	132
	<i>Concluding Infeasibility from Phase I</i>	133
	<i>Big-M Method</i>	135
	<i>Big-M Outcomes</i>	136
	Exercises	138
	References	141

CHAPTER 4 *LINEAR PROGRAMMING MODELS* 143

4.1	Allocation Models	144
	<i>Allocation Decision Variables</i>	145
	<i>Forest Service Allocation Model</i>	145
4.2	Blending Models	147
	<i>Ingredient Decision Variables</i>	148
	<i>Composition Constraints</i>	148
	<i>Swedish Steel Example Model</i>	150
	<i>Ratio Constraints</i>	150
4.3	Operations Planning Models	152
	<i>Tubular Products Operations Planning Model</i>	153
	<i>CFPL Decision Variables</i>	156
	<i>Continuous Variables for Integer Quantities</i>	157
	<i>CFPL Objective Function</i>	157
	<i>CFPL Constraints</i>	158
	<i>Balance Constraints</i>	158
	<i>CFPL Application Model</i>	160

- 4.4 Shift Scheduling and Staff Planning Models 162
 - ONB Decision Variables and Objective Function* 163
 - ONB Constraints* 164
 - Covering Constraints* 164
 - ONB Shift Scheduling Application Model* 165
- 4.5 Time-Phased Models 166
 - Time-Phased Decision Variables* 167
 - Time-Phased Balance Constraints* 168
 - IFS Cash Flow Model* 169
 - Time Horizons* 170
- 4.6 Models with Linearizable Nonlinear Objectives 171
 - Maxisum Highway Patrol Application Model* 172
 - Minimax and Maximin Objective Functions* 173
 - Nonlinear Maximin Highway Patrol Application Model* 173
 - Linearizing Minimax and Maximin Objective Functions* 173
 - Linearized Maximin Highway Patrol Example Model* 174
 - Nonlinear VP Location Model* 175
 - Min Deviation Objective Functions* 176
 - Linearizing Min Deviation Objective Functions* 176
 - Linearized VP Location Model* 177
- 4.7 Stochastic Programming 179
 - Deterministic Model of QA Example* 180
 - Stochastic Programming with Recourse* 181
 - Stochastic Programming Modeling of the QA Application* 182
 - Extensive Form versus Large-Scale Techniques* 184
- Exercises 185**
- References 200**

CHAPTER 5 SIMPLEX SEARCH FOR LINEAR PROGRAMMING 201

- 5.1 LP Optimal Solutions and Standard Form 201
 - Global Optima in Linear Programs* 203
 - Interior, Boundary, and Extreme Points* 204
 - Optimal Points in Linear Programs* 207
 - LP Standard Form* 208
 - Converting Inequalities to Nonnegativities with Slack Variables* 209
 - Converting Nonpositive and Unrestricted Variables to Nonnegative* 211
 - Standard Notation for LPs* 213
- 5.2 Extreme-Point Search and Basic Solutions 216
 - Determining Extreme Points with Active Constraints* 216
 - Adjacent Extreme Points and Edges* 216

	<i>Basic Solutions</i>	219
	<i>Existence of Basic Solutions</i>	221
	<i>Basic Feasible Solutions and Extreme Points</i>	225
5.3	The Simplex Algorithm	227
	<i>Standard Display</i>	227
	<i>Initial Basic Solution</i>	228
	<i>Simplex Directions</i>	228
	<i>Improving Simplex Directions and Reduced Costs</i>	231
	<i>Step Size and the Minimum Ratio Rule</i>	232
	<i>Updating the Basis</i>	234
	<i>Rudimentary Simplex Algorithm</i>	235
	<i>Rudimentary Simplex Solution of Top Brass Example</i>	236
	<i>Stopping and Global Optimality</i>	236
	<i>Extreme-Point or Extreme-Direction</i>	238
5.4	Dictionary and Tableau Representations of Simplex	238
	<i>Simplex Dictionaries</i>	239
	<i>Simplex Tableaux</i>	241
	<i>Simplex Algorithm with Dictionaries or Tableaux</i>	242
	<i>Correspondence to the Improving Search Paradigm</i>	242
	<i>Comparison of Formats</i>	243
5.5	Two Phase Simplex	243
	<i>Starting Basis in the Two Phase Simplex</i>	245
	<i>Three Possible Outcomes for Linear Programs</i>	247
	<i>Clever Clyde Infeasible Case</i>	247
	<i>Clever Clyde Optimal Case</i>	250
	<i>Clever Clyde Unbounded Case</i>	252
5.6	Degeneracy and Zero-Length Simplex Steps	253
	<i>Degenerate Solutions</i>	253
	<i>Zero-Length Simplex Steps</i>	255
	<i>Progress through Changing of Bases</i>	256
5.7	Convergence and Cycling with Simplex	257
	<i>Finite Convergence with Positive Steps</i>	257
	<i>Degeneracy and Cycling</i>	258
5.8	Doing it Efficiently: Revised Simplex	260
	<i>Computations with Basis Inverses</i>	260
	<i>Updating the Representation of B^{-1}</i>	264
	<i>Basic Variable Sequence in Revised Simplex</i>	266
	<i>Computing Reduced Costs by Pricing</i>	267
	<i>Revised Simplex Search of Top Brass Application</i>	269
5.9	Simplex with Simple Upper and Lower Bounds	272
	<i>Lower- and Upper-Bounded Standard Form</i>	272
	<i>Basic Solutions with Lower and Upper Bounds</i>	274
	<i>Unrestricted Variables with No Bounds</i>	274
	<i>Increasing and Decreasing Nonbasic Variable Values</i>	275
	<i>Step Size with Increasing and Decreasing Values</i>	276

<i>Case with No Basis Change</i>	277
<i>Lower- and Upper-Bounded Simplex Algorithm</i>	277
<i>Lower- and Upper-Bounded Simplex on Top Brass Application</i>	277
Exercises	280
References	285

CHAPTER 6 DUALITY, SENSITIVITY, AND OPTIMALITY IN LINEAR PROGRAMMING 287

6.1	Generic Activities Versus Resources Perspective	288
	<i>Objective Functions as Costs and Benefits</i>	288
	<i>Choosing a Direction for Inequality Constraints</i>	288
	<i>Inequalities as Resource Supplies and Demands</i>	288
	<i>Equality Constraints as Both Supplies and Demands</i>	289
	<i>Variable-Type Constraints</i>	290
	<i>Variables as Activities</i>	290
	<i>LHS Coefficients as Activity Inputs and Outputs</i>	290
6.2	Qualitative Sensitivity to Changes in Model Coefficients	293
	<i>Relaxing versus Tightening Constraints</i>	293
	<i>Swedish Steel Application Revisited</i>	293
	<i>Effects of Changes in Right-Hand Sides</i>	294
	<i>Effects of Changes in LHS Constraint Coefficients</i>	296
	<i>Effects of Adding or Dropping Constraints</i>	297
	<i>Effects of Unmodeled Constraints</i>	297
	<i>Changing Rates of Constraint Coefficient Impact</i>	298
	<i>Effects of Objective Function Coefficient Changes</i>	299
	<i>Changing Rates of Objective Function Coefficient Impact</i>	301
	<i>Effects of Adding or Dropping Variables</i>	303
6.3	Quantifying Sensitivity to Changes in LP Model Coefficients: A Dual Model	304
	<i>Primals and Duals Defined</i>	304
	<i>Dual Variables</i>	304
	<i>Dual Variable Types</i>	305
	<i>Two Crude Application Again</i>	306
	<i>Dual Variables as Implicit Marginal Resource Prices</i>	307
	<i>Implicit Activity Pricing in Terms of Resources Produced and Consumed</i>	308
	<i>Main Dual Constraints to Enforce Activity Pricing</i>	309
	<i>Optimal Value Equality between Primal and Dual</i>	310
	<i>Primal Complementary Slackness between Primal Constraints and Dual Variable Values</i>	311
	<i>Dual Complementary Slackness between Dual Constraints and Primal Variable Values</i>	312

6.4	Formulating Linear Programming Duals	313
	<i>Form of the Dual for Nonnegative Primal Variables</i>	314
	<i>Duals of LP Models with Nonpositive and Unrestricted Variables</i>	316
	<i>Dual of the Dual is the Primal</i>	317
6.5	Computer Outputs and What If Changes of Single Parameters	318
	<i>CFPL Example Primal and Dual</i>	318
	<i>Constraint Sensitivity Outputs</i>	320
	<i>Right-Hand-Side Ranges</i>	322
	<i>Constraint What If's</i>	324
	<i>Variable Sensitivity Outputs</i>	326
	<i>Objective Coefficient Ranges</i>	328
	<i>Variable What If's</i>	330
	<i>Dropping and Adding Constraint What If's</i>	332
	<i>Dropping and Adding Variable What If's</i>	333
6.6	Bigger Model Changes, Reoptimization, and Parametric Programming	335
	<i>Ambiguity at Limits of the RHS and Objective Coefficient Ranges</i>	335
	<i>Connection between Rate Changes and Degeneracy</i>	337
	<i>Reoptimization to Make Sensitivity Exact</i>	338
	<i>Parametric Variation of One Coefficient</i>	338
	<i>Assessing Effects of Multiple Parameter Changes</i>	340
	<i>Parametric Multiple-RHS Change</i>	341
	<i>Parametric Change of Multiple Objective Function Coefficients</i>	343
6.7	Duality and Optimality in Linear Programming	344
	<i>Dual of the Dual</i>	345
	<i>Weak Duality between Objective Values</i>	345
	<i>Unbounded and Infeasible Cases</i>	347
	<i>Complementary Slackness and Optimality</i>	349
	<i>Strong Duality and Karush-Kuhn-Tucker (KKT) Optimality Conditions for Linear Programs</i>	351
	<i>Models in Standard Form</i>	352
	<i>Standard Form LPs in Partitioned Basic Format</i>	354
	<i>Basic Solutions in Partitioned Form</i>	355
	<i>Complementary Dual Basic Solutions</i>	355
	<i>Primal Simplex Optimality and Necessity of KKT Conditions</i>	357
6.8	Dual Simplex Search	359
	<i>Choosing an Improving Direction</i>	361
	<i>Determining a Dual Step Size to Retain Dual Feasibility</i>	361
	<i>Changing the Primal Solution and Basis Update</i>	362

- 6.9 Primal-Dual Simplex Search 365
 Choosing an Improving Dual Direction 367
 Determining a Dual Step Size 368
Exercises 371
References 384

CHAPTER 7 *INTERIOR POINT METHODS FOR LINEAR PROGRAMMING* 385

- 7.1 Searching through the Interior 385
 Interior Points 386
 Objective as a Move Direction 386
 Boundary Strategy of Interior Point Methods 387
 Interior in LP Standard Form 389
 Projecting to Deal with Equality Constraints 390
 Improvement with Projected Directions 394
- 7.2 Scaling with the Current Solution 396
 Affine Scaling 396
 Diagonal Matrix Formalization of Affine Scaling 396
 Affine-Scaled Standard Form 399
 Projecting on Affine-Scaled Equality Constraints 401
 Computational Effort in Interior Point Computations 402
- 7.3 Affine Scaling Search 402
 Affine Scaling Move Directions 402
 Feasibility and Improvement of Affine Scaling Directions 404
 Affine Scaling Step Size 404
 Termination in Affine Scaling Search 407
 Affine Scaling Search of the Frannie's Firewood Application 408
- 7.4 Log Barrier Methods for Interior Point Search 408
 Barrier Objective Functions 408
 Problems with Gradient Directions 411
 Newton Steps for Barrier Search 412
 Newton Step Barrier Search Step Sizes 415
 Impact of the Barrier Multiplier μ 417
 Barrier Algorithm Multiplier Strategy 418
 Newton Step Barrier Algorithm 418
 Newton Barrier Solution of Frannie's Firewood Application 419
- 7.5 Primal-Dual Interior-Point Search 421
 KKT Optimality Conditions 421
 Strategy of Primal-Dual Interior-Point Search 422
 Feasible Move Directions 422
 Management of Complementary Slackness 423
 Step Size 423
 Solving the Conditions for Move Directions 423

7.6 Complexity of Linear Programming Search 428
Length of Input for LP Instances 428
Complexity of Simplex Algorithms for LP 429
Complexity of Interior-Point Algorithms for LP 430
Exercises 430
References 435

CHAPTER 8 **MULTIOBJECTIVE OPTIMIZATION AND GOAL PROGRAMMING** 437

8.1 Multiobjective Optimization Models 437
Bank Three Example Objectives 438
Bank Three Example Model 439
Dynamometer Ring Design Model 440
Hazardous Waste Disposal Model 442

8.2 Efficient Points and the Efficient Frontier 443
Efficient Points 443
Identifying Efficient Points Graphically 444
Efficient Frontier 445
Plots in Objective Value Space 446
Constructing the Efficient Frontier 446

8.3 Preemptive Optimization and Weighted Sums of Objectives 448
Preemptive Optimization 448
Preemptive Optimization of the Bank Three Application 448
Preemptive Optimization and Efficient Points 451
Preemptive Optimization and Alternative Optima 451
Weighted Sums of Objectives 451
Weighted-Sum Optimization of the Hazardous Waste Application 452
Weighted-Sum Optimization and Efficient Points 453

8.4 Goal Programming 454
Goal or Target Levels 454
Goal Form of Bank Three Application 454
Soft Constraints 455
Deficiency Variables 455
Expressing Soft Constraints in Mathematical Programs 456
Goal Program Objective Function: Minimizing (Weighted) Deficiency 457
Goal Linear Program Model of the Bank Three Application 457
Alternative Deficiency Weights in the Objective 458
Preemptive Goal Programming 459
Preemptive Goal Programming of the Bank Three Application 459

Preemptive Goal Programming by Weighting the Objective 461
Practical Advantage of Goal Programming in Multiobjective Problems 461
Goal Programming and Efficient Points 462
Modified Goal Program Formulation to Assure Efficient Points 464
Exercises 465
References 475

CHAPTER 9 *SHORTEST PATHS AND DISCRETE DYNAMIC PROGRAMMING* 477

9.1 Shortest Path Models 477
Nodes, Arcs, Edges, and Graphs 478
Paths 479
Shortest Path Problems 481
Classification of Shortest Path Models 481
Undirected and Directed Graphs (Digraphs) 482
Two Ring Application Model 485

9.2 Dynamic Programming Approach to Shortest Paths 485
Families of Shortest Path Models 485
Functional Notation 486
Optimal Paths and Subpaths 487
Negative Dicycles Exception 488
Principle of Optimality 489
Functional Equations 489
Functional Equations for One Node to All Others 489
Sufficiency of Functional Equations in the One to All Case 490
Functional Equations for All Nodes to All Others 493
Solving Shortest Path Problems by Linear Programming 494

9.3 Shortest Paths from One Node to All Others:
 Bellman–Ford 494
 Solving the Functional Equations 495
 Repeated Evaluation Algorithm: Bellman–Ford 495
 Bellman–Ford Solution of the Two Ring Circus Application 496
 Justification of the Bellman–Ford Algorithm 498
 Recovering Optimal Paths 499
 Encountering Negative Dicycles with Bellman–Ford 500

9.4 Shortest Paths from All Nodes to All Others:
 Floyd–Warshall 501
 Floyd–Warshall Algorithm 501
 Floyd–Warshall Solution of the Littleville Application 503
 Recovering Optimal Paths 507
 Detecting Negative Dicycles with Floyd–Warshall 507

9.5	Shortest Path from One Node to All Others with Costs	
	Nonnegative: Dijkstra	509
	<i>Permanently and Temporarily Labeled Nodes</i>	509
	<i>Least Temporary Criterion for Next Permanent Node</i>	510
	<i>Dijkstra Algorithm Solution of the Texas Transfer</i>	
	Application	510
	<i>Recovering Paths</i>	514
	<i>Justification of the Dijkstra Algorithm</i>	514
9.6	Shortest Paths from One Node to All Others in Acyclic Digraphs	515
	<i>Acyclic Digraphs</i>	515
	<i>Shortest Path Algorithm for Acyclic Digraphs</i>	518
	<i>Acyclic Shortest Path Example</i>	518
	<i>Longest Path Problems and Acyclic Digraphs</i>	519
9.7	CPM Project Scheduling and Longest Paths	520
	<i>Project Management</i>	520
	<i>CPM Project Networks</i>	521
	<i>CPM Schedules and Longest Paths</i>	523
	<i>Critical Paths</i>	523
	<i>Computing an Early Start Schedule for the We Build Construction</i>	
	Application	524
	<i>Late Start Schedules and Schedule Slack</i>	526
	<i>Acyclic Character of Project Networks</i>	527
9.8	Discrete Dynamic Programming Models	528
	<i>Sequential Decision Problems</i>	528
	<i>States in Dynamic Programming</i>	529
	<i>Digraphs for Dynamic Programs</i>	530
	<i>Dynamic Programming Solutions as an Optimal Path</i>	531
	<i>Dynamic Programming Functional Equations</i>	532
	<i>Dynamic Programming Models with Both Stages and States</i>	532
	<i>Dynamic Programming Modeling of the President's Library</i>	
	Application	534
	<i>Backward Solution of Dynamic Programs</i>	534
	<i>Multiple Problem Solutions Obtained Simultaneously</i>	537
9.9	Solving Integer Programs with Dynamic Programming	537
	<i>Dynamic Programming Modeling of Electoral Vote</i>	
	Knapsack	538
9.10	Markov Decision Processes	541
	<i>Elements of MDP Models</i>	541
	<i>Solution of the Breast Cancer MDP</i>	545
	Exercises	546
	References	556

CHAPTER 10 NETWORK FLOWS AND GRAPHS 557

- 10.1 Graphs, Networks, and Flows 557
 - Digraphs, Nodes, and Arcs* 557
 - OOI Application Network* 558
 - Minimum Cost Flow Models* 559
 - Sources, Sinks, and Transshipment Nodes* 560
 - OOI Application Model* 560
 - Total Supply = Total Demand* 562
 - Starting Feasible Solutions* 563
 - Artificial Network Flow Model* 563
 - Time-Expanded Flow Models and Networks* 565
 - Time-Expanded Modeling of Agrico Application* 567
 - Node–Arc Incidence Matrices and Matrix Standard Form* 568
- 10.2 Cycle Directions for Network Flow Search 570
 - Chains, Paths, Cycles, and Dicycles* 570
 - Cycle Directions* 571
 - Maintaining Flow Balance with Cycle Directions* 573
 - Feasible Cycle Directions* 574
 - Improving Cycle Directions* 576
 - Step Size with Cycle Directions* 577
 - Sufficiency of Cycle Directions* 578
 - Rudimentary Cycle Direction Search for Network Flows* 580
 - Rudimentary Cycle Direction Search of the OOI Application* 580
- 10.3 Cycle Cancelling Algorithms for Optimal Flows 582
 - Residual Digraphs* 582
 - Feasible Cycle Directions and Dicycles of Residual Digraphs* 584
 - Improving Feasible Cycle Directions and Negative Dicycles of Residual Digraphs* 585
 - Using Shortest Path Algorithms to Find Cycle Directions* 586
 - Cycle Cancelling Solution of the OOI Application* 586
 - Polynomial Computational Order of Cycle Cancelling* 589
- 10.4 Network Simplex Algorithm for Optimal Flows 591
 - Linear Dependence in Node–Arc Matrices and Cycles* 591
 - Spanning Trees of Networks* 594
 - Spanning Tree Bases for Network Flow Models* 595
 - Network Basic Solutions* 596
 - Simplex Cycle Directions* 597
 - Network Simplex Algorithm* 598
 - Network Simplex Solution of OOI Application* 598

10.5	Integrality of Optimal Network Flows	601
	<i>When Optimal Network Flows Must Be Integer</i>	601
	<i>Total Unimodularity of Node–Arc Incidence Matrices</i>	603
10.6	Transportation and Assignment Models	604
	<i>Transportation Problems</i>	604
	<i>Standard Form for Transportation Problems</i>	605
	<i>Assignment Problems</i>	607
	<i>Balancing Unequal Sets with Dummy Elements</i>	610
	<i>Integer Network Flow Solution of Assignment Problems</i>	610
	<i>CAM Assignment Application Model</i>	610
10.7	Hungarian Algorithm for Assignment Problems	611
	<i>Primal-Dual Strategy and Initial Dual Solution</i>	611
	<i>Equality Subgraph</i>	613
	<i>Labeling to Search for a Primal Solution in the Equality Subgraph</i>	614
	<i>Dual Update and Revised Equality Subgraph</i>	616
	<i>Solution Growth Along Alternating Paths</i>	617
	<i>Computational Order of the Hungarian Algorithm</i>	617
10.8	Maximum Flows and Minimum Cuts	618
	<i>Improving Feasible Cycle Directions and Flow Augmenting Paths</i>	620
	<i>The Max Flow Min Cut Algorithm</i>	621
	<i>Solution of Max Flow Application of Figure 10.25(a) with Algorithm 10E</i>	621
	<i>Equivalence of Max Flow and Min Cut Values</i>	624
	<i>Computational Order of Algorithm 10E Effort</i>	625
10.9	Multicommodity and Gain/Loss Flows	625
	<i>Multicommodity Flows</i>	625
	<i>Multicommodity Flow Models</i>	627
	<i>Tractability of Multicommodity Flow Models</i>	629
	<i>Flows with Gains and Losses</i>	630
	<i>Gain and Loss Network Flow Models</i>	631
	<i>Tractability of Network Flows with Gains and Losses</i>	632
10.10	Min/Max Spanning Trees	633
	<i>Minimum/Maximum Spanning Trees and the Greedy Algorithm</i>	633
	<i>Solution of the WE Application 10.8 by Greedy Algorithm 10F</i>	633
	<i>Representing Greedy Results in a Composition Tree</i>	635
	<i>ILP Formulation of the Spanning Tree Problem</i>	635
	<i>Computational Order of the Greedy Algorithm</i>	638
	Exercises	639
	References	653

CHAPTER 11 DISCRETE OPTIMIZATION MODELS 655

- 11.1 Lumpy Linear Programs and Fixed Charges 655
 - Swedish Steel Application with All-or-Nothing Constraints* 655
 - ILP Modeling of All-or-Nothing Requirements* 656
 - Swedish Steel Model with All-or-Nothing Constraints* 656
 - ILP Modeling of Fixed Charges* 658
 - Swedish Steel Application with Fixed Charges* 658
- 11.2 Knapsack and Capital Budgeting Models 661
 - Knapsack Problems* 661
 - Capital Budgeting Models* 662
 - Budget Constraints* 663
 - Modeling Mutually Exclusive Choices* 664
 - Modeling Dependencies between Projects* 665
 - NASA Application Model* 665
- 11.3 Set Packing, Covering, and Partitioning Models 666
 - Set Packing, Covering, and Partitioning Constraints* 667
 - Minimum Cover EMS Model* 669
 - Maximum Coverage EMS Model* 670
 - Column Generation Models* 672
- 11.4 Assignment and Matching Models 675
 - Assignment Constraints* 675
 - CAM Linear Assignment Application Revisited* 676
 - Linear Assignment Models* 676
 - Quadratic Assignment Models* 677
 - Mall Layout Application Model* 678
 - Generalized Assignment Models* 680
 - CDOT Application Model* 682
 - Matching Models* 683
 - Superfi Application Model* 684
 - Tractability of Assignment and Matching Models* 684
- 11.5 Traveling Salesman and Routing Models 685
 - Traveling Salesman Problem* 685
 - Symmetric versus Asymmetric Cases of the TSP* 686
 - Formulating the Symmetric TSP* 687
 - Subtours* 688
 - ILP Model of the Symmetric TSP* 690
 - ILP Model of the Asymmetric TSP* 690
 - Quadratic Assignment Formulation of the TSP* 692
 - Problems Requiring Multiple Routes* 693
 - KI Truck Routing Application Model* 694
- 11.6 Facility Location and Network Design Models 695
 - Facility Location Models* 695
 - ILP Model of Facilities Location* 696

	<i>Tmark Facilities Location Application Model</i>	697
	<i>Network Design Models</i>	699
	<i>Wastewater Network Design Application Model</i>	701
11.7	Processor Scheduling and Sequencing Models	702
	<i>Single-Processor Scheduling Problems</i>	703
	<i>Time Decision Variables</i>	703
	<i>Conflict Constraints and Disjunctive Variables</i>	704
	<i>Handling of Due Dates</i>	706
	<i>Processor Scheduling Objective Functions</i>	706
	<i>ILP Formulation of Minmax Scheduling Objectives</i>	708
	<i>Equivalences among Scheduling Objective Functions</i>	710
	<i>Job Shop Scheduling</i>	710
	<i>Custom Metalworking Application Decision Variables and Objective</i>	711
	<i>Precedence Constraints</i>	711
	<i>Conflict Constraints in Job Shops</i>	712
	<i>Custom Metalworking Application Model</i>	713
	Exercises	715
	References	729

CHAPTER 12 EXACT DISCRETE OPTIMIZATION METHODS 731

12.1	Solving by Total Enumeration	731
	<i>Total Enumeration</i>	732
	<i>Swedish Steel All-or-Nothing Application</i>	732
	<i>Exponential Growth of Cases to Enumerate</i>	733
12.2	Relaxations of Discrete Optimization Models and Their Uses	734
	<i>Constraint Relaxations</i>	735
	<i>Linear Programming Relaxations</i>	737
	<i>Relaxations Modifying Objective Functions</i>	738
	<i>Proving Infeasibility with Relaxations</i>	738
	<i>Solution Value Bounds from Relaxations</i>	739
	<i>Optimal Solutions from Relaxations</i>	742
	<i>Rounded Solutions from Relaxations</i>	744
	<i>Stronger LP Relaxations</i>	747
	<i>Choosing Big-M Constants</i>	749
12.3	Branch and Bound Search	751
	<i>Partial Solutions</i>	752
	<i>Completions of Partial Solutions</i>	752
	<i>Tree Search</i>	753
	<i>Incumbent Solutions</i>	756
	<i>Candidate Problems</i>	757
	<i>Terminating Partial Solutions with Relaxations</i>	758

- LP-Based Branch and Bound* 760
- Branching Rules for LP-Based Branch and Bound* 761
- LP-Based Branch and Bound Solution of the River Power Application* 762
- 12.4 **Refinements to Branch and Bound** 764
 - Branch and Bound Solution of NASA Capital Budgeting Application* 764
 - Rounding for Incumbent Solutions* 765
 - Branch and Bound Family Tree Terminology* 768
 - Parent Bounds* 769
 - Terminating with Parent Bounds* 769
 - Stopping Early: Branch and Bound as a Heuristic* 770
 - Bounds on the Error of Stopping with the Incumbent Solution* 771
 - Depth First, Best First, and Depth Forward Best Back Sequences* 772
- 12.5 **Branch and Cut** 777
 - Valid Inequalities* 777
 - Branch and Cut Search* 778
 - Branch and Cut Solution of the River Power Application* 779
- 12.6 **Families of Valid Inequalities** 782
 - Gomory Cutting Planes (Pure Integer Case)* 782
 - Gomory Mixed-Integer Cutting Planes* 785
 - Families of Valid Inequalities from Specialized Models* 787
- 12.7 **Cutting Plane Theory** 788
 - The Convex Hull of Integer Feasible Solutions* 789
 - Linear Programs over Convex Hulls* 791
 - Faces, Facets, and Categories of Valid Inequalities* 792
 - Affinely Independent Characterization of Facet-Inducing Valid Inequalities* 794
 - Partial Dimensional Convex Hulls and Valid Equalities* 795
- Exercises** 797
- References** 810

CHAPTER 13 LARGE-SCALE OPTIMIZATION METHODS 811

- 13.1 **Delayed Column Generation and Branch and Price** 811
 - Models Attractive for Delayed Column Generation* 813
 - Partial Master Problems* 815
 - Generic Delayed Column Generation Algorithm* 815
 - Application of Algorithm 13A to Application 13.1* 815
 - Generating Eligible Columns to Enter* 817
 - Branch and Price Search* 819

- 13.2 Lagrangian Relaxation 822
 - Lagrangian Relaxations* 822
 - Tractable Lagrangian Relaxations* 824
 - Lagrangian Relaxation Bounds and Optima* 825
 - Lagrangian Duals* 827
 - Lagrangian versus Linear Programming Relaxation Bounds* 830
 - Lagrangian Dual Objective Functions* 832
 - Subgradient Search for Lagrangian Bounds* 833
 - Application of Subgradient Search to Numerical Example* 835
- 13.3 Dantzig–Wolfe Decomposition 836
 - Reformulation in Terms of Extreme Points and Extreme Directions* 838
 - Reformulation from GB Application 13.4 Subproblems* 839
 - Delayed Generation of Subproblem Extreme-Point and Extreme-Direction Columns* 840
 - Dantzig–Wolfe Solution of GB Application 13.4* 841
- 13.4 Benders Decomposition 842
 - Benders Decomposition Strategy* 844
 - Optimality in Benders Algorithm 13E* 845
 - Solution of Heart Guardian Application 13.5 with Benders Algorithm 13E* 846
- Exercises 849**
- References 854**

CHAPTER 14 COMPUTATIONAL COMPLEXITY THEORY 855

- 14.1 Problems, Instances, and the Challenge 855
 - The Challenge* 856
- 14.2 Measuring Algorithms and Instances 857
 - Computational Orders* 857
 - Instance Size as the Length of an Encoding* 859
 - Expressions for Encoding Length of All a Problem's Instances* 860
- 14.3 The Polynomial-Time Standard for Well-Solved Problems 861
- 14.4 Polynomial and Nondeterministic-Polynomial Solvability 862
 - Decision versus Optimization Problems* 862
 - Class P - Polynomially Solvable Decision Problems* 863
 - Class NP - Nondeterministic-Polynomially Solvable Decision Problems* 864
 - Polynomial versus Nondeterministic Polynomial Problem Classes* 865

- 14.5 Polynomial-Time Reductions and NP-Hard Problems 866
Polynomial Reductions between Problems 866
NP-Complete and NP-Hard Problems 868
- 14.6 P versus NP 869
The $P \neq NP$ Conjecture 870
- 14.7 Dealing with NP-Hard Problems 871
Special Cases 871
Pseudo-Polynomial Algorithms 871
Average Case Performance 872
Stronger Relaxations and Cuts for B&B and B&C 872
Specialized Heuristics with Provable Worst-Case Performance 872
General Purpose Approximate/Heuristic Algorithms 874
Exercises 875
References 878

CHAPTER 15 HEURISTIC METHODS FOR APPROXIMATE DISCRETE OPTIMIZATION 879

- 15.1 Constructive Heuristics 879
Rudimentary Constructive Search Algorithm 880
Greedy Choices of Variables to Fix 880
Greedy Rule for NASA Application 881
Constructive Heuristic Solution of NASA Application 882
Need for Constructive Search 884
Constructive Search of KI Truck Routing Application 885
- 15.2 Improving Search Heuristics for Discrete Optimization INLPs 886
Rudimentary Improving Search Algorithm 886
Discrete Neighborhoods and Move Sets 887
NCB Application Revisited 888
Choosing a Move Set 889
Rudimentary Improving Search of the NCB Application 891
Multistart Search 892
- 15.3 Tabu and Simulated Annealing Metaheuristics 893
Difficulty with Allowing Nonimproving Moves 894
Tabu Search 894
Tabu Search of the NCB Application 895
Simulated Annealing Search 897
Simulated Annealing Search of NCB Application 899

- 15.4 Evolutionary Metaheuristics and Genetic Algorithms 902
 - Crossover Operations in Genetic Algorithms* 902
 - Managing Genetic Algorithms with Elites, Immigrants, Mutations, and Crossovers* 903
 - Solution Encoding for Genetic Algorithm Search* 904
 - Genetic Algorithm Search of NCB Application* 905
 - Exercises** 906
 - References** 911

CHAPTER 16 UNCONSTRAINED NONLINEAR PROGRAMMING 913

- 16.1 Unconstrained Nonlinear Programming Models 913
 - USPS Single-Variable Application Model* 915
 - Neglecting Constraints to Use Unconstrained Methods* 915
 - Curve Fitting and Regression Problems* 916
 - Linear versus Nonlinear Regression* 917
 - Regression Objective Functions* 918
 - Custom Computer Curve Fitting Application Model* 918
 - Maximum Likelihood Estimation Problems* 919
 - PERT Maximum Likelihood Application Model* 921
 - Smooth versus Nonsmooth Functions and Derivatives* 922
 - Usable Derivatives* 923
- 16.2 One-Dimensional Search 924
 - Unimodal Objective Functions* 924
 - Golden Section Search* 925
 - Golden Section Solution of USPS Application* 927
 - Bracketing and 3-Point Patterns* 929
 - Finding a 3-Point Pattern* 930
 - Quadratic Fit Search* 932
 - Quadratic Fit Solution of USPS Application* 933
- 16.3 Derivatives, Taylor Series, and Conditions for Local Optima in Multiple Dimensions 935
 - Improving Search Paradigm* 935
 - Local Information and Neighborhoods* 936
 - First Derivatives and Gradients* 936
 - Second Derivatives and Hessian Matrices* 937
 - Taylor Series Approximations with One Variable* 939
 - Taylor Series Approximations with Multiple Variables* 940
 - Stationary Points and Local Optima* 941
 - Saddle Points* 943
 - Hessian Matrices and Local Optima* 943
- 16.4 Convex/Concave Functions and Global Optimality 947
 - Convex and Concave Functions Defined* 948
 - Sufficient Conditions for Unconstrained Global Optima* 950
 - Convex/Concave Functions and Stationary Points* 951

- Tests for Convex and Concave Functions* 951
Unimodal versus Convex/Concave Objectives 954
- 16.5 Gradient Search 955
Gradient Search Algorithm 955
Gradient Search of Custom Computer Application 956
Steepest Ascent/Descent Property 958
Zigzagging and Poor Convergence of Gradient Search 959
- 16.6 Newton's Method 959
Newton Step 960
Newton's Method 961
Newton's Method on the Custom Computer Application 962
Rapid Convergence Rate of Newton's Method 963
Computational Trade-offs between Gradient and Newton Search 963
Starting Close with Newton's Method 964
- 16.7 Quasi-Newton Methods and BFGS Search 964
Deflection Matrices 965
Quasi-Newton Approach 965
Guaranteeing Directions Improve 966
BFGS Formula 966
BFGS Search of Custom Computer Application 967
Verifying Quasi-Newton Requirements 971
Approximating the Hessian Inverse with BFGS 972
- 16.8 Optimization without Derivatives and Nelder–Mead 973
Nelder–Mead Strategy 973
Nelder–Mead Direction 976
Nelder–Mead Limited Step Sizes 977
Nelder–Mead Shrinking 979
Nelder–Mead Search of PERT Application 980
- Exercises 981**
References 986

CHAPTER 17 **CONSTRAINED NONLINEAR PROGRAMMING 987**

- 17.1 Constrained Nonlinear Programming Models 987
Beer Belge Location-Allocation Model 988
Linearly Constrained Nonlinear Programs 989
Texaco Gasoline Blending Model 990
Engineering Design Models 992
Oxygen System Engineering Design Model 993
- 17.2 Convex, Separable, Quadratic, and Posynomial Geometric Programming Special NLP Forms 995
Pfizer Optimal Lot Sizing Model 996
Convex Programs 998

	<i>Special Tractability of Convex Programs</i>	1000
	<i>Separable Programs</i>	1001
	<i>Special Tractability of Separable Programs</i>	1002
	<i>Quadratic Portfolio Management Model</i>	1004
	<i>Quadratic Programs Defined</i>	1005
	<i>Special Tractability of Quadratic Programs</i>	1006
	<i>Cofferdam Application Model</i>	1007
	<i>Posynomial Geometric Programs</i>	1008
	<i>Special Tractability of Posynomial Geometric Programs</i>	1010
17.3	Lagrange Multiplier Methods	1011
	<i>Reducing to Equality Form</i>	1011
	<i>Lagrangian Function and Lagrange Multipliers</i>	1012
	<i>Stationary Points of the Lagrangian Function</i>	1013
	<i>Lagrangian Stationary Points and the Original Model</i>	1014
	<i>Lagrange Multiplier Procedure</i>	1015
	<i>Interpretation of Lagrange Multipliers</i>	1017
	<i>Limitations of the Lagrangian Approach</i>	1018
17.4	Karush–Kuhn–Tucker Optimality Conditions	1019
	<i>Fully Differentiable NLP Model</i>	1019
	<i>Complementary Slackness Conditions</i>	1019
	<i>Lagrange Multiplier Sign Restrictions</i>	1020
	<i>KKT Conditions and KKT Points</i>	1020
	<i>Improving Feasible Directions and Local Optima Revisited</i>	1022
	<i>KKT Conditions and Existence of Improving Feasible Directions</i>	1024
	<i>Sufficiency of KKT Conditions for Optimality</i>	1027
	<i>Necessity of KKT Conditions for Optimality</i>	1027
17.5	Penalty and Barrier Methods	1028
	<i>Penalty Methods</i>	1028
	<i>Penalty Treatment of the Service Desk Application</i>	1030
	<i>Concluding Constrained Optimality with Penalties</i>	1031
	<i>Differentiability of Penalty Functions</i>	1031
	<i>Exact Penalty Functions</i>	1032
	<i>Managing the Penalty Multiplier</i>	1033
	<i>Sequential Unconstrained Penalty Technique (SUMT)</i>	1033
	<i>Barrier Methods</i>	1034
	<i>Barrier Treatment of Service Desk Application</i>	1035
	<i>Converging to Optimality with Barrier Methods</i>	1036
	<i>Managing the Barrier Multiplier</i>	1037
	<i>Sequential Unconstrained Barrier Technique</i>	1037
17.6	Reduced Gradient Algorithms	1038
	<i>Standard Form for NLPs with Linear Constraints</i>	1038
	<i>Conditions for Feasible Directions with Linear Constraints</i>	1040
	<i>Bases of the Main Linear Equalities</i>	1040

	<i>Basic, Nonbasic, and Superbasic Variables</i>	1041
	<i>Maintaining Equalities by Solving Main Constraints for Basic Variables</i>	1042
	<i>Active Nonnegativities and Degeneracy</i>	1042
	<i>Reduced Gradients</i>	1043
	<i>Reduced Gradient Move Direction</i>	1044
	<i>Line Search in Reduced Gradient Methods</i>	1046
	<i>Basis Changes in Reduced Gradient Methods</i>	1047
	<i>Reduced Gradient Algorithm</i>	1047
	<i>Reduced Gradient Search of Filter Tuning Application</i>	1048
	<i>Major and Minor Iterations in Reduced Gradient</i>	1049
	<i>Second-Order Extensions of Reduced Gradient</i>	1050
	<i>Generalized Reduced Gradient Procedures for Nonlinear Constraints</i>	1050
17.7	Quadratic Programming Methods	1051
	<i>General Symmetric Form of Quadratic Programs</i>	1051
	<i>Quadratic Program Form of the Filter Tuning Application</i>	1052
	<i>Equality-Constrained Quadratic Programs and KKT Conditions</i>	1053
	<i>Direct Solution of KKT Conditions for Quadratic Programs</i>	1054
	<i>Active Set Strategies for Quadratic Programming</i>	1055
	<i>Step Size with Active Set Methods</i>	1056
	<i>Stopping at a KKT Point with Active Set Methods</i>	1057
	<i>Dropping a Constraint from the Active Set</i>	1058
	<i>Active Set Solution of the Filter Tuning Application</i>	1059
17.8	Sequential Quadratic Programming	1061
	<i>Lagrangian and Newton Background</i>	1061
	<i>Sequential Quadratic Programming Strategy</i>	1062
	<i>Application of Algorithm 17E to Modified Pfizer Application 17.9</i>	1064
	<i>Approximations to Reduce Computation</i>	1065
17.9	Separable Programming Methods	1065
	<i>Pfizer Application 17.4 Revisited</i>	1066
	<i>Piecewise Linear Approximation to Separable Functions</i>	1067
	<i>Linear Program Representation of Separable Programs</i>	1069
	<i>Correctness of the LP Approximation to Separable Programs</i>	1070
	<i>Convex Separable Programs</i>	1071
	<i>Difficulties with Nonconvex Separable Programs</i>	1073
17.10	Posynomial Geometric Programming Methods	1073
	<i>Posynomial Geometric Program Form</i>	1073
	<i>Cofferdam Application Revisited</i>	1074
	<i>Logarithmic Change of Variables in GPs</i>	1075

Convex Transformed GP Model 1076
Direct Solution of the Transformed Primal GP 1077
Dual of a Geometric Program 1077
Degrees of Difficulty and Solving the GP Dual 1079
Recovering a Primal GP Solution 1080
Derivation of the GP Dual 1080
Signomial Extension of GPs 1082
Exercises 1082
References 1093

APPENDIX: GROUP PROJECTS 1095

SELECTED ANSWERS 1099

INDEX 1123

Preface

It is now nearly two decades since publication of the first edition of my textbook *Optimization in Operations Research*. Since that time thousands of students and hundreds of instructors, researchers, and practitioners have had the opportunity to benefit from its consistent content and accessible design. Of course, not all have seen benefit, but many have written kind reviews and letters expressing their high regard for the book. Also, the Institute of Industrial Engineers honored it with their Joint Publishers Book-of-the-Year Award in 1999.




In this second edition, I have tried to preserve what was best about the original while updating it with new and enhanced content. The goal remains the same—to make the tools of optimization modeling and analysis accessible to advanced undergraduate and beginning graduate students who follow the book in their studies, as well as researchers and working practitioners who use it as a reference for self-study. Emphasis is on the skills and intuitions they can carry away and apply in real settings or later coursework.

Although aimed at that same goal, much is new in the second edition:

- Stochastic optimization is covered for the first time with Stochastic Programming in Chapter 4, and Markov Decision Processes in Chapter 9.
- Coverage of linear programming techniques is expanded in Chapter 6 to encompass dual and primal-dual methods.
- New sections rigorously formalize optimality conditions for linear programming in Chapter 6, and cutting plane theory in Chapter 12.
- Treatment of the Hungarian Algorithm for assignment, and min/max spanning tree methods has been added to Chapter 10.
- A whole new Chapter 13 is devoted to large-scale optimization techniques including Delayed Column Generation, Lagrangian Relaxation, Dantzig–Wolfe Decomposition, and Benders’ Partitioning.
- A whole new Chapter 14 treats the theory of computational complexity to provide a rigorous foundation for comparing problems and algorithms.
- Nonlinear Chapter 17 now includes coverage of the popular Sequential Quadratic Programming method.
- More generally, additional mathematical rigor is added to justifications of methods throughout the book, including tracking computational orders for most.

New topics seek to cover even more completely the full breadth of optimization (or mathematical programming) that might be of interest to the book’s intended audience. Those span linear, integer, nonlinear, network, and dynamic programming models and algorithms, in both single and multi-objective context, and a rich sample of domains where they have been applied.

With content so inclusive, it is important to recognize that almost no reader or course will ever use it all, much less in the exact sequence presented in the book. For that reason, I have tried to make the organization of material as transparent and re-entrant as possible.

Dependencies between sections are minimized and clearly identified with explicit references. One- and two-page **Primers** concisely review prerequisite material where it is needed in the development to save diversions to other sources. To keep the focus on intuitions and strategies behind topics, **Definitions**, **Principles** and **Algorithms** are set out in easy-to-spot boxes, where high-level ideas can be located and absorbed quickly. When more detail is of interest, computations and discussions that may extend to several pages are recapped immediately in concise **Examples** (also marked for easy identification). For readers and instructors seeking more reinforcement with **Exercises** at the end of chapters, convenient icons clearly tag which of those require computer software () or advanced calculators (), and which have answers provided at the back of the book ()

The new edition also builds on my firm belief that making optimization materials accessible and exciting to readers of diverse backgrounds requires making the book a continuing discourse on optimization modeling. Every algorithm and analytic principle is developed in the context of a brief story set out as an **Application**. Also, computational exercises often begin with a formulation step. Many of those stories are derived from real OR applications footnoted in the development. Story settings—however contrived—provide a context for understanding both the needed decision variables, constraints and objectives of model forms, and steps in computation. For example, ideas like improving directions are more intuitive if some quantity in a story, not just a mathematical function, is clearly getting better as an algorithm is pursued. Likewise, binary decision variables become intuitive if the reader can see the either-or nature of some application decisions.

A related conviction is that students cannot really learn any mathematical topic without working with it in homework exercises. That is why the second edition continues the tradition of the first in providing a full range of exercises at the end of each chapter. Some continue from the first edition, but many are new or posed over modified parameter values. The range of exercises begins with verifications of algorithm details, definitions and properties, which are essential to building intuition about the methods. But a range of formulation exercises is also included extending from tiny examples subject to graphic or inspection solution to more complex applications drawn from real OR work that challenge formulation skills. In addition, a new Group Projects appendix details assignments I have used for years to engage student teams more deeply in published reports of actual optimization applications.

Early introductory books in optimization focused heavily on hand application of algorithms to compute solutions of tiny examples. With almost all real optimization now done with the help of large-scale computer software, more recent sources have sometimes limited attention to formulating data sets for submission to one of those algorithms—treating the computation largely as a black box.

I reject both these extremes. Graphic solution of small instances and hand implementation of algorithmic methods are essential if students are to internalize the principles on which the computation is based. The second edition continues my earlier pattern of moving quickly to such intuitive examples as each new concept is introduced. At the same time, no reader will ever grow excited about the power of optimization methods if he or she sees them applied only to tiny examples, much less abstract mathematical forms. That is why many of the examples and exercises in

both the first and second editions of the book ask students to apply available class software on models of greater size, where answers are not apparent until formal methods are shown to reveal them. Brief sections have also been added on coding models for software like AMPL.

Perhaps the greatest challenge in trying to bridge undergraduate and beginning graduate audiences in optimization is the question of mathematical rigor. Elementary treatments simply introduce algorithmic mechanics with little if any argument for their correctness. On the other hand, more advanced books on optimization methods often devolve quickly into rigorous mathematical propositions and formal proofs with almost no discussion of underlying strategies, intuitions, and tractability.

My effort in the first edition was to bridge that gap by focusing on the intuitions and strategies behind methods, and on their relative tractability, while offering only limited arguments for their correctness. In the interest of better serving the introductory graduate and self-study audiences, the second edition adds significantly more rigor to the arguments presented. They are still not stated in theorem or proof format, but most key elements of rationales are now justified.

I am proud of how the long overdue second edition has emerged, and I hope readers will agree that it is a significant advance over the first. I look forward to your comments as the new developments are absorbed.

I want to thank deeply the hundreds of students, friends, and colleagues at Georgia Tech, Purdue and the University of Arkansas for their advice and encouragement as the new edition has taken shape. This goes especially for a series of Graduate Assistants who have helped with exercises and solutions, and for the patience and support of department heads Marlin Thomas, Dennis Engi, John English, Kim Needy, and Ed Pohl. Finally, I need to thank my family—especially my wife Blanca and my son Rob—for their patience and encouragement in my long slog to finish the task.

About the Author



Dr. Ronald L. (Ron) Rardin retired as Distinguished Professor Emeritus in 2013 after a 40-year record of leadership as an educator and researcher in optimization methods and their application culminating after 2007 as John and Mary Lib White Distinguished Professor of Industrial Engineering on the faculty of the University of Arkansas-Fayetteville. He headed the University's Center on Innovation in Healthcare Logistics (CIHL) targeting supply chain and material flow aspects of healthcare operations in collaboration with a variety of healthcare industry organizations. He also took the lead with colleagues at Arkansas in founding the Health Systems Engineering Alliance (HSEA) of industrial engineering academic programs interested in healthcare.

Earlier, Professor Rardin retired in 2006 as Professor Emeritus of Industrial Engineering at Purdue University after completing 24 years there, including directing the Purdue Energy Modeling Research Groups, and playing a leading role in

Purdue's Regenstrief Center for Healthcare Engineering. Previously he had served on the Industrial and Systems Engineering faculty at the Georgia Institute of Technology for 9 years. He also served the profession in a rotation from 2000–2003 as Program Director for Operations Research and Service Enterprise Engineering at the National Science Foundation, including founding the latter program to foster research in service industries.

Dr. Rardin obtained his B.A. and M.P.A. degrees from the University of Kansas, and after working in city government, consulting and distribution for five years, a Ph.D. at Georgia Institute of Technology.

His teaching and research interests center on large-scale optimization modeling and algorithms, especially their application in healthcare and energy. He is an award winning teacher of those topics, and co-author of numerous research papers and two comprehensive textbooks: a graduate text *Discrete Optimization*, published in 1988, and a comprehensive undergraduate textbook on mathematical programming, *Optimization in Operations Research*, which was published in 1998 and received the Institute of Industrial Engineers (IIE) Book of the Year award. Among his many other honors, he is a Fellow of both IIE and the Institute for Operations Research and the Management Sciences (INFORMS), as well as 2012 winner of the IIE's David F. Baker award for career research achievement.

Optimization in Operations Research

This page intentionally left blank

Problem Solving with Mathematical Models

Any student with the most elementary scientific training has encountered the idea of solving problems by analyzing mathematical equations that approximate the physical realities of the universe we inhabit. Countless questions about objects falling, beams shearing, gases diffusing, currents flowing, and so on, are reduced to simple computations upon skillful application of one of the natural laws passed to us by Newton, Ohm, Einstein, and others.

The applicable laws may be less enduring, but “operations” problems such as planning work shifts for large organizations, choosing investments for available funds, or designing facilities for customer service can also be posed in mathematical form. A **mathematical model** is the collection of variables and relationships needed to describe pertinent features of such a problem.

Definition 1.1 | **Operations research (OR)** is the study of how to form mathematical models of complex engineering and management problems and how to analyze them to gain insight about possible solutions.

In this chapter some of the fundamental issues and vocabulary related to operations research are introduced.

1.1 OR APPLICATION STORIES

Operations research techniques have proved useful in an enormous variety of application settings. One of the goals of this book is to expose students to as broad a sample as possible. All application examples, many end-of-chapter exercises, several complete sections, and three full chapters present and analyze stories based on OR applications.

Whenever possible, these problems are drawn from reports of real operations research practice (identified in footnotes). Of course, they are necessarily reduced in size and complexity, and numerical details are almost always made up by the author.

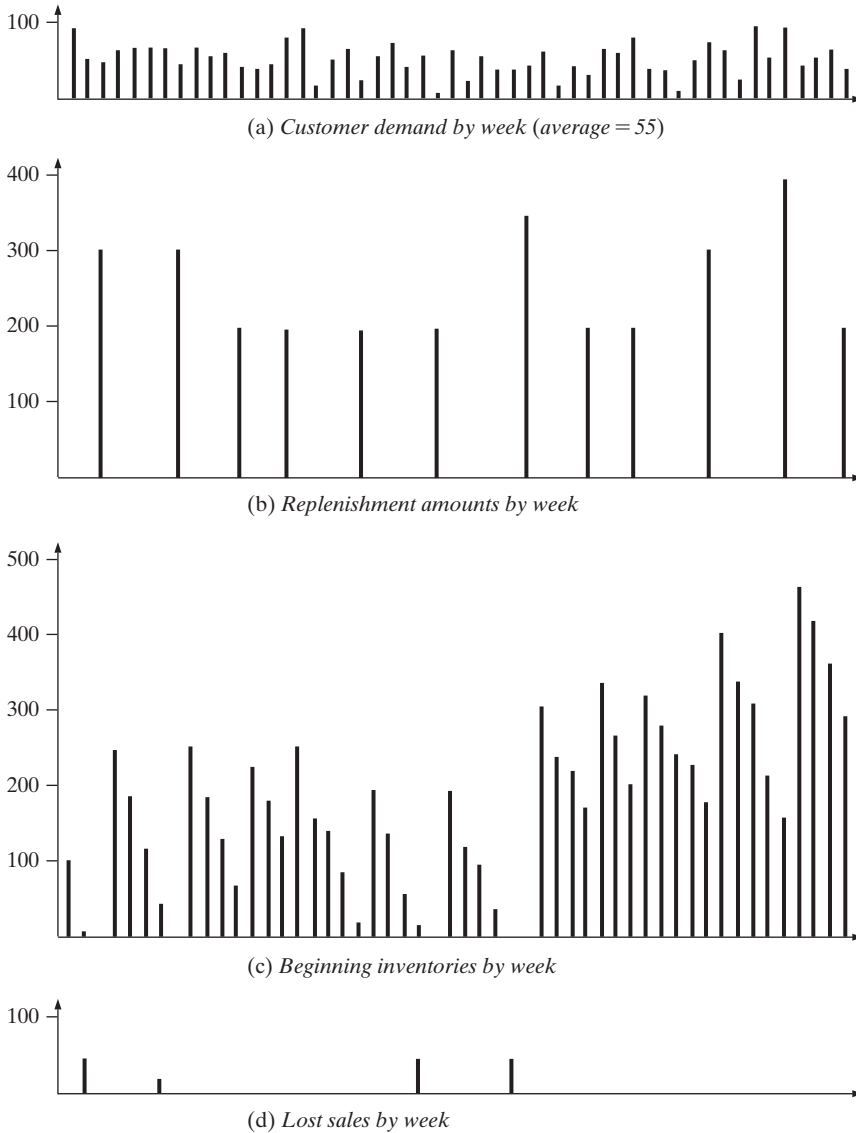


FIGURE 1.1 Mortimer Middleman Example History

Other stories illustrate key elements of standard applications but greatly oversimplify, to facilitate quick learning.

A handful of continuing examples are even smaller and more contrived. They still have a story, but convenience in illustrating methodological issues takes precedence over reality of application.

APPLICATION 1.1: MORTIMER MIDDLEMAN

Our first story is of the totally made-up variety. Mortimer Middleman—friends call him MM—operates a modest wholesale diamond business. Several times each year

MM travels to Antwerp, Belgium, to replenish his diamond supply on the international market. The wholesale price there averages approximately \$700 per carat, but Antwerp market rules require him to buy at least 100 carats each trip. Mortimer and his staff then resell the diamonds to jewelers at a profit of \$200 per carat. Each of the Antwerp trips requires 1 week, including the time for Mortimer to get ready, and costs approximately \$2000.

Customer demand values in Figure 1.1(a) show that business has been good. Over the past year, customers have come in to order an average of 55 carats per week.

Part (c) of Figure 1.1 illustrates Mortimer's problem. Weekly levels of on-hand diamond inventory have varied widely, depending on the ups and downs in sales and the pattern of MM's replenishment trips [Figure 1.1(b)].

Sometimes Mortimer believes that he is holding too much inventory. The hundreds of carats of diamonds on hand during some weeks add to his insurance costs and tie up capital that he could otherwise invest. MM has estimated that these holding costs total 0.5% of wholesale value per week (i.e., $0.005 \times \$700 = \3.50 per carat per week).

At other times, diamond sales—and Mortimer's \$200 per carat profit—have been lost because customer demand exceeded available stock [see Figure 1.1(d)]. When a customer calls, MM must either fill the order on the spot or lose the sale.

Adding this all up for the past year, MM estimates holding costs of \$38,409, unrealized profits from lost sales of \$31,600, and resupply travel costs of \$24,000, making the annual total \$94,009. Can he do better?

1.2 OPTIMIZATION AND THE OPERATIONS RESEARCH PROCESS

Operations research deals with **decision problems** like that of Mortimer Middleman by formulating and analyzing mathematical models—mathematical representations of pertinent problem features. Figure 1.2 illustrates this OR process.

The process begins with formulation or modeling. We define the variables and quantify the relationships needed to describe relevant system behavior.

Next comes analysis. We apply our mathematical skills and technology to see what conclusions the model suggests. Notice that these conclusions are drawn from

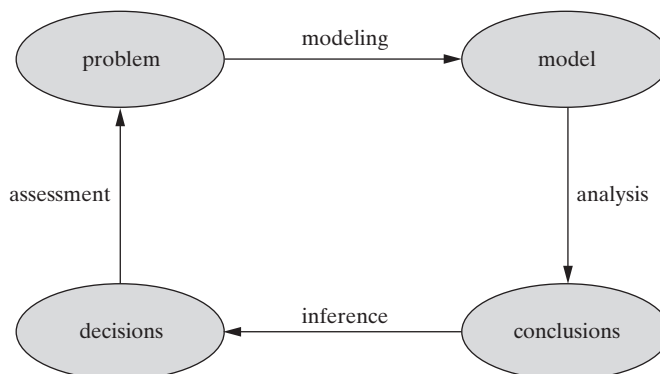


FIGURE 1.2 Operations Research Process

the model, not from the problem that it is intended to represent. To complete the process, we must engage in inference, that is, argue that conclusions drawn from the model are meaningful enough to infer decisions for the person or persons with the problem.

Often, an assessment of decisions inferred in this way shows them to be too inadequate or extreme for implementation. Further thought leads to revised modeling, and the loop continues.

Decisions, Constraints, and Objectives

We always begin modeling by focusing on three dimensions of the problem:

Definition 1.2 The three fundamental concerns in forming operations research models are (a) the **decisions** open to decision makers, (b) the **constraints** limiting decision choices, and (c) the **objectives** making some decisions preferred to others.

In dealing with virtually any decision problem—engineering, management, or even personal—explicitly defining the decisions, constraints, and objectives helps to clarify the issues. Mortimer is obviously the decision maker in our diamond inventory management example. What decisions does he get to make?

Actually, MM makes hundreds of decisions each year about when to replenish his stock and how much to buy. However, it is common in inventory management circumstances such as Mortimer’s to reduce the question to two policy decisions: What **reorder point** level of inventory should trigger a decision to buy new stock, and what **order quantity** should be purchased each time? These two variables constitute our decisions. We presume that each time on-hand inventory falls below the reorder point, Mortimer will head to Antwerp to buy a standard reorder quantity.

The next issue is constraints. What restrictions limit MM’s decision choices? In this example there aren’t very many. It is only necessary that both decisions be non-negative numbers and that the order quantity conform to the 100 carat minimum of the Antwerp market.

The third element is objectives. What makes one decision better than another? In MM’s case the objective is clearly to minimize cost. More precisely, we want to minimize the sum of holding, replenishment, and lost-sales costs.

Summarizing in a verbal model or word description, our goal is *to choose a non-negative reorder point and a nonnegative reorder quantity to minimize the sum of holding, replenishment, and lost-sales costs subject to the reorder quantity being at least 100.*

Optimization and Mathematical Programming

Verbal models can help organize an analyst’s thinking, but in this book we address a higher standard. We deal exclusively with optimization (also called mathematical programming).

Definition 1.3 **Optimization models** (also called **mathematical programs**) represent problem choices as decision variables and seek values that maximize or minimize objective functions of the decision variables subject to constraints on variable values expressing the limits on possible decision choices.

With our Mortimer Middleman example, the decision variables are

$q \triangleq$ reorder quantity purchased on each replenishment trip

$r \triangleq$ reorder point signaling the need for replenishment

(Here and throughout \triangleq means “is defined to be.”) Constraints require only that

$$q \geq 100$$

$$r \geq 0$$

The objective function,

$c(q, r) \triangleq$ total cost using a reorder quantity of q and a reorder point r

remains to be explicitly represented mathematically. We seek to minimize $c(q, r)$ over values of q and r satisfying all constraints.

Constant-Rate Demand Assumption

How we formulate constraints and objectives in terms of decision variables depends on what assumptions we are willing to make about the underlying system. We begin with a strong assumption regarding **constant-rate demand**: Assume that demand occurs at a constant rate of 55 carats per week. It is clear in Figure 1.1(a) that the demand rate is not exactly constant, but it does average 55 carats per week. Assuming that it is 55 carats in every week leads to some relatively simple analysis.

If the demand rate is constant, the pattern of on-hand inventory implied by a particular q and r will take one of the periodic “sawtooth” forms illustrated in Figure 1.3. Each time a shipment arrives, inventory will increase by order size q , then decline at the rate of 55 carats per week, producing regular cycles. Part (a) shows a case where inventory never runs out. A **safety stock** of (theoretically) untouched inventory protects against demand variability we have ignored. At the other extreme is part (c). Sales are lost because inventory runs out during the **lead time** between reaching the reorder point r and arrival of a new supply. Part (b) has neither safety stock nor lost sales. Stock runs out just as new supply arrives.

Back of Envelope Analysis

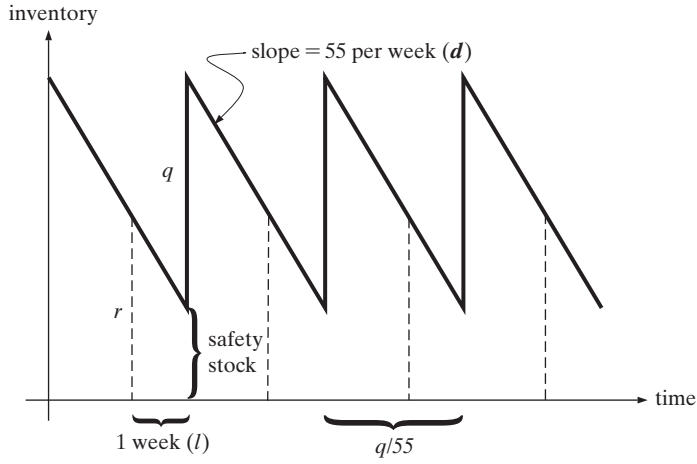
In cases where there are no lost sales [Figure 1.3(a) and (b)] it is easy to compute the length of each sawtooth cycle.

$$\frac{\text{order quantity}}{\text{demand rate}} = \frac{q}{55}$$

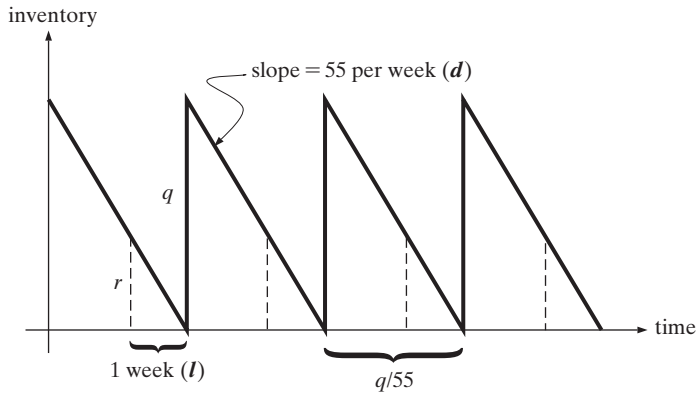
With lost sales [Figure 1.3(c)], each cycle is extended by a period when MM is out of stock that depends on both q and r .

Clearly, both modeling and analysis would be easier if we could ignore the lost-sales case. Can we afford to neglect lost sales? As in so many OR problems, a bit of crude “back of envelope” examination of the relevant costs will help us decide.

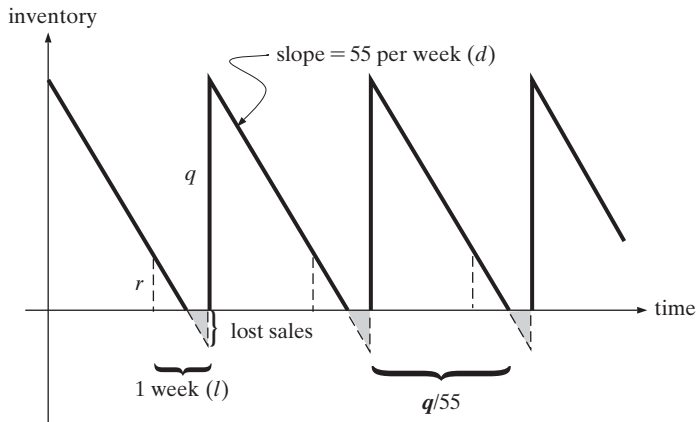
Lost sales may occur under the best of plans because of week-to-week variation in demand. Under our constant-rate demand assumption, however,



(a) With safety stock



(b) No safety stock or lost sales



(c) With lost sales

FIGURE 1.3 Inventories Under Constant-Rate Demand

there is no variation. Furthermore, MM can afford to add a unit to q and carry it for up to

$$\frac{\text{cost of lost sale}}{\text{weekly holding cost}} = \frac{\$200}{\$3.50} \approx 57.1 \text{ weeks}$$

rather than lose a carat of sales. Since the history in Figure 1.1 shows that inventory typically has been held no more than 4 to 6 weeks, it seems safe to make a second assumption regarding **no lost sales**: Assume that lost sales are not allowed.

Constant-Rate Demand Model

Since customers order a constant-rate 55 carats during the 1 week it takes Mortimer to carry out an Antwerp trip, both inventory at order arrival and lost sales can be computed by comparing 55 to r . If $r < 55$, we lose $(55 - r)$ carats of sales each cycle, something we have decided not to permit. Thus we may deduce the constraint

$$r \geq 55$$

With r restricted to be at least 55, $(r - 55)$ is the safety stock, and the cycle of rising and falling inventory repeats every $q/55$ weeks. Inventory on hand ranges from $(r - 55)$ at the low point of a cycle to $(r - 55) + q$ as a shipment arrives. The average will be the midpoint of these values, $(r - 55) + q/2$.

We are finally in a position to express all relevant costs. Holding cost per week is just the average inventory held times \$3.50. Replenishment cost per week is \$2000 divided by the cycle length or time between replenishments. Our first optimization model is

$$\begin{aligned} \text{minimize } c &= 3.50 \left[(r - 55) + \frac{q}{2} \right] + \frac{2000}{q/55} & (1.1) \\ \text{subject to } & q \geq 100, \quad r \geq 55 \end{aligned}$$

Feasible and Optimal Solutions

Remember that our goal is to help Mortimer make decisions. Since the decisions are the variables in our model, we would like to characterize good values for **decision variables** q and r .

Definition 1.4 | A **feasible solution** is a choice of values for the decision variables that satisfies all constraints. **Optimal solutions** are feasible solutions that achieve objective function value(s) as good as those of any other feasible solutions.

For example, $q = 200, r = 90$ is feasible in constant-rate demand model (1.1) because both constraints are satisfied: $200 \geq 100$ and $90 \geq 55$.

Here we can go farther and find an optimal solution. To begin, notice that if r deviates from demand 55, we incur extra holding cost and that no constraint prevents choosing r exactly 55. We conclude that

$$r^* = 55$$

will tell MM the perfect moment to start travel preparations. The asterisk (*) or **star** on a variable always denotes its optimal value.

Substituting this optimal choice of r of (1.1), the objective function reduces to

$$c(q, r) \triangleq 3.50 \left(\frac{q}{2} \right) + 2000 \left(\frac{55}{q} \right) \quad (1.2)$$

Elementary calculus will tell us how to finish (differentiate with respect to q and solve for a suitable point where the derivative is zero). To avoid being diverted by mathematical details in this introductory chapter, we leave the computation as an exercise for the reader.

The graphic presentation of cost function (1.2) in Figure 1.4 confirms the calculus result that the minimum average weekly cost occurs at

$$q^* = \pm \sqrt{\frac{2(2000)(55)}{3.50}} \approx 250.7$$

Since this value easily satisfies the $q \geq 100$ constraint, it is optimal.

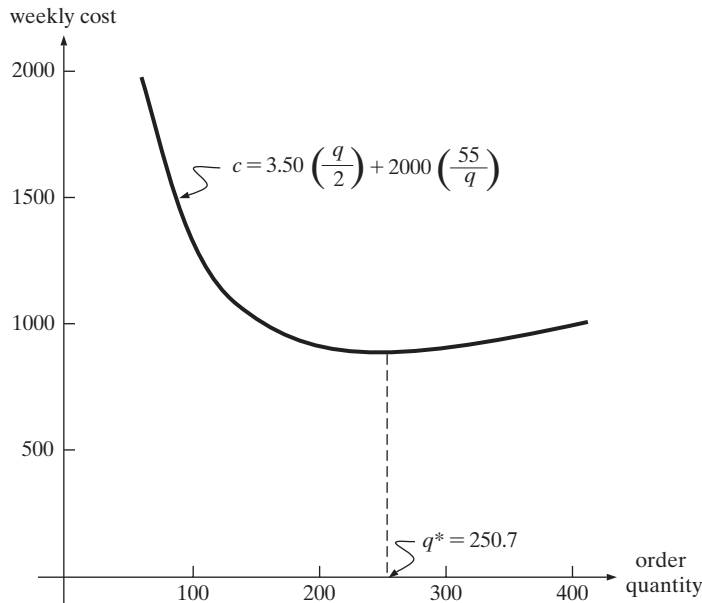


FIGURE 1.4 Optimal MM Order Quantity Under Constant-Rate Demand

To summarize, our assumptions of constant-rate demand and no lost sales have led us to advise Mortimer to go to Antwerp whenever inventory drops below $r^* = 55$ carats and to buy $q^* = 250.7$ carats of new diamonds each trip. Substituting these values in the objective function of (1.1), total cost should be about \$877.50 per week or \$45,630 per year—quite an improvement over Mortimer’s real experience of \$94,009.

1.3 SYSTEM BOUNDARIES, SENSITIVITY ANALYSIS, TRACTABILITY, AND VALIDITY

The modeling in Section 1.2 took as given many quantities, such as the demand per week and the cost per carat held, then computed optimal values for reorder point and reorder quantity. A line between those items taken as settled and those to be decided is called the **system boundary**. Figure 1.5 illustrates how **parameters**—quantities taken as given—define objective functions and constraints applicable to the decision model inside. Together, parameters and decision variables determine results measured as **output variables**.

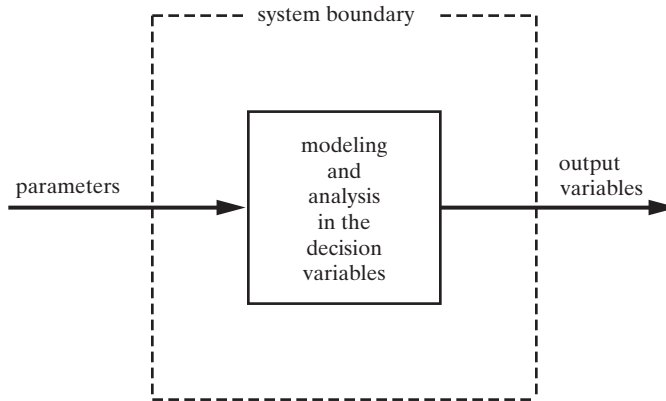


FIGURE 1.5 System Boundaries

EOQ Under Constant-Rate Demand

Only cost c is an output variable in our constant-rate demand model of Mortimer Middleman's problem. Enumerating the parameters, let

$d \triangleq$ weekly demand (55 carats)

$f \triangleq$ fixed cost of replenishment (\$2000)

$h \triangleq$ cost per carat per week for holding inventory (\$3.50)

$s \triangleq$ cost per carat of lost sales (\$200)

$\ell \triangleq$ lead time between reaching the reorder point and receiving a new supply (1 week)

$m \triangleq$ minimum order size (100 carats)

A great attraction of our constant-rate demand analysis is that it can be done just as well in terms of these symbols. If lost sales are not allowed, repetition of the analysis (calculus) in terms of symbolic parameters will cause us to conclude that

Principle 1.5

$$\begin{aligned} \text{optimal reorder quantity } q^* &= \sqrt{\frac{2fd}{h}} \\ \text{optimal reorder point } r^* &= \ell d \end{aligned}$$

These results hold as long as $q^* \geq m$ (to be general, we also need to specify $q^* \geq \ell d$).

The square root expression for q^* now exhibits one of the oldest results in operations research: the classic **economic order quantity (EOQ)** formula for inventory management. Although we will soon see problems with the r^* part of the solution in MM's case, EOQ order quantities yield reliable inventory policies in a wide variety of settings.

System Boundaries and Sensitivity Analysis

To see the power of symbolic results such as equations 1.5, we must recognize the inherent arbitrariness in system boundaries. If we took nothing as settled, models would mushroom in complexity and meaningful analysis would become impossible. Still, parameters we choose to regard as fixed at a system boundary often are known only vaguely. For example, MM's trips may cost approximately $f = \$2000$, but prices no doubt change with time, and more careful tabulation of past expenses might show that trips actually average \$1000 or \$3000. Figure 1.6 shows the significant implications for the optimal q^* . Our analysis with parameter $f = \$2000$ had $q^* = 250.7$. With $f = \$1000$, the optimal order quantity is 177.3; and with $f = \$3000$, it is 307.1.

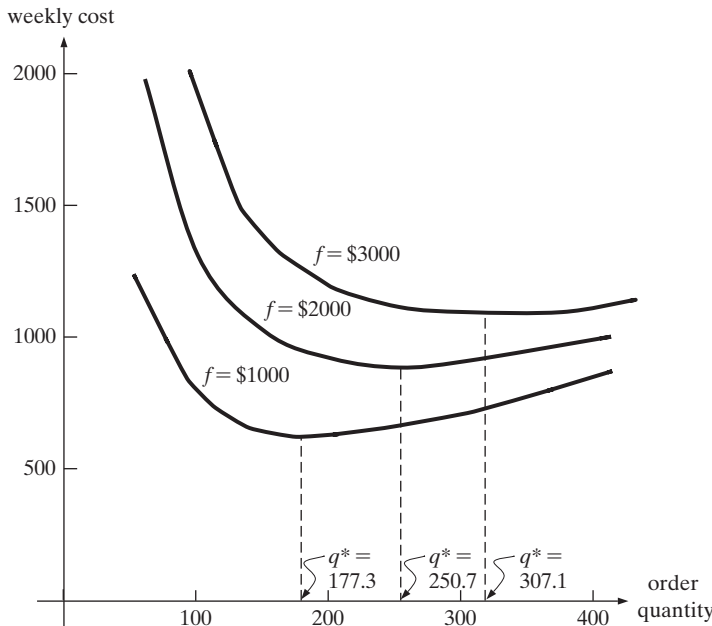


FIGURE 1.6 Sensitivity of MM Constant - Demand Results to Parameters

Such variations in input parameter values taken as fixed at a system boundary may dramatically affect the results of OR analysis.

Definition 1.6 | **Sensitivity analysis** is an exploration of results from mathematical models to evaluate how they depend on the values chosen for parameters.

Any really complete operations research study includes an investigation of the sensitivity of results to parameter values.

Closed-Form Solutions

Solutions prescribing choices for decision variables by simple formulas in the input variables are called **closed-form** solutions. Expressions [1.5](#) are an example.

The power of closed-form solutions lies with their providing results for many values of the parameters. Determining optimal q^* and r^* for the parameter values assumed in Section 1.2 was no small success. With closed-form solutions, however, we can determine sensitivities of optimal results to changes in input parameters. For example, we can see immediately that travel cost parameter f influences q^* in proportion to its square root and has no impact on r^* .

Principle 1.7 | Closed-form solutions represent the ultimate in analysis of mathematical models because they provide both immediate results and rich sensitivity analysis.

Tractability versus Validity

Definition 1.8 | **Tractability** in modeling means the degree to which the model admits convenient analysis—how much analysis is practical.

Our constant-rate demand model of Mortimer Middleman's problem has proved highly tractable because elementary calculus produced closed-form optimal solutions. Should we conclude that the goal of every OR study must be to formulate and analyze a model admitting closed-form solutions? Absolutely not!

Take another look at the operations research process depicted in Figure 1.2. Our purpose is not mathematical elegance but insights that will really help people like Mortimer Middleman deal with their problems. To assess the merit of operations research results, we must also consider another dimension.

Definition 1.9 | The **validity** of a model is the degree to which inferences drawn from the model hold for the real system.

The complete analysis possible with our constant-rate demand model derived directly from the strong assumptions that we made to achieve a simple mathematical form. As we will see in the following subsections, those same assumptions bring the applicability of our closed-form results into real doubt.

Our study of OR will return to this dilemma again and again.

Principle 1.10 | OR analysts almost always confront a tradeoff between validity of models and their tractability to analysis.

1.4 DESCRIPTIVE MODELS AND SIMULATION

Our constant-rate demand analysis of Mortimer Middleman's problem started by reducing all the information in Figure 1.1(a) to a single number, an average of 55 carats sold per week. Why not use more of what we know?

Simulation over MM's History

A **simulation model** is a computer program that simply steps through the behavior of a system of interest and reports experience. Of course, the behavior is tracked in computer variables and program logic rather than in a physical system.

To illustrate for Mortimer Middleman's inventory problem, suppose that we retain our verbal model (i.e., continue to focus on a reorder point r and a reorder quantity q). A straightforward computer program could then step through the 52 weeks of available history to try any given r and q . At each step, the program would:

1. Check whether MM is due to arrive with a new order of size q .
2. Determine if r has been reached so that a new trip is needed.
3. Reduce inventory by the demand actually experienced.

Holding cost for the week can be reported as \$3.50 times the average number of carats in inventory. Replenishment cost is \$2000 in each trip week. Lost-sales cost for the week is \$200 times any excess of demand over the available inventory.

Table 1.1 details such a simulation from MM's demand history. Using the optimal $q^* = 251$ and $r^* = 55$ computed in our constant-rate demand model, this simulation reports beginning inventory for each week, customer demand taken from Figure 1.1(a), inventory management actions taken, and their consequences for holding cost, replenishment cost, and lost sales.

Total simulated cost for all 52 weeks is \$108,621. Recall that Mortimer estimated actual cost at \$94,009, and the constant demand analysis said that costs under q^* and r^* should decline to \$45,630. The simulation model now is telling us that if MM adopts our supposedly optimal constant-rate demand policy, he will spend $\$108,621 - \$94,009 = \$14,621$ more than if he simply keeps operating as he always has. This is hardly a help.

Simulation Model Validity

Can these new results be trusted? Only a clairvoyant could be certain, but it seems safe to conclude that the simulation results, based on an entire year's actual experience, should be taken more seriously than values derived from the constant-rate demand model.

Definition and Principle 1.11	Simulation models often possess high validity because they track true system behavior fairly accurately.
-------------------------------	--

It should be noted, however, that the simulation was also based on some assumptions. For example, we have implicitly presumed that future demand will exactly mirror last year's experience.

TABLE 1.1 Deterministic Simulation of MM's Problem from Prior History

Week, t	Beginning Inventory	Customer Demand	Simulated Action	Holding Cost	Replenishment Cost	Lost Sales
1	100	94	Sell 94	\$185.5	0	0
2	6	54	Below $r = 55$, so trip; sell 6	1.2	\$2,000	\$ 9,600
3	0	52	$q = 251$ arrive; sell 52	787.5	0	0
4	199	64	Sell 64	584.5	0	0
5	135	69	Sell 69	353.5	0	0
6	66	69	Sell 66	110.5	0	600
7	0	68	Below $r = 55$, so trip; sell 0	0.0	2,000	13,600
8	0	47	$q = 251$ arrive; sell 47	798.0	0	0
9	204	68	Sell 68	595.0	0	0
10	136	56	Sell 56	378.0	0	0
11	80	62	Sell 62	171.5	0	0
12	18	44	Below $r = 55$, so trip; sell 18	12.9	2,000	5,200
13	0	41	$q = 251$ arrive; sell 41	808.5	0	0
14	210	46	Sell 46	654.5	0	0
15	164	84	Sell 84	427.0	0	0
16	80	94	Sell 80	119.1	0	2,800
17	0	18	Below $r = 55$, so trip; sell 0	0.0	2,000	3,600
18	0	52	$q = 251$ arrive; sell 52	787.5	0	0
19	199	67	Sell 67	581.0	0	0
20	132	26	Sell 26	416.5	0	0
21	106	59	Sell 59	269.5	0	0
22	47	77	Below $r = 55$, so trip; sell 47	50.2	2,000	6,000
23	0	42	$q = 251$ arrive; sell 42	805.0	0	0
24	209	59	Sell 59	630.0	0	0
25	150	11	Sell 11	507.5	0	0
26	139	67	Sell 67	371.0	0	0
27	72	25	Sell 25	210.0	0	0
28	47	60	Below $r = 55$, so trip; sell 47	64.4	2,000	2,600
29	0	41	$q = 251$ arrive; sell 41	808.5	0	0
30	210	42	Sell 42	661.5	0	0
31	168	47	Sell 47	507.5	0	0
32	121	66	Sell 66	308.0	0	0
33	55	20	Below $r = 55$, so trip; sell 20	157.5	2,000	0
34	35	46	$q = 251$ arrive; sell 46	920.5	0	0
35	240	36	Sell 36	777.0	0	0
36	204	69	Sell 69	595.0	0	0
37	135	64	Sell 64	360.5	0	0
38	71	83	Sell 71	106.3	0	2,400
39	0	42	Below $r = 55$, so trip; sell 0	0.0	2,000	8,400
40	0	38	$q = 251$ arrive; sell 38	812.0	0	0
41	213	13	Sell 13	724.5	0	0
42	200	50	Sell 50	612.5	0	0
43	150	77	Sell 77	392.0	0	0
44	73	64	Sell 64	143.5	0	0
45	9	27	Below $r = 55$, so trip; sell 9	5.2	2,000	3,600
46	0	96	$q = 251$ arrive; sell 96	710.5	0	0
47	155	57	Sell 57	444.5	0	0
48	98	95	Sell 95	178.5	0	0
49	3	46	Below $r = 55$, so trip; sell 3	0.3	2,000	8,600
50	0	56	$q = 251$ arrive; sell 56	780.5	0	0
51	195	68	Sell 68	563.5	0	0
52	127	42	Sell 42	371.0	0	0

Descriptive versus Prescriptive Models

Think now about tractability. How much did our simulation actually tell us? It estimated the holding, replenishment, and lost-sales costs of operating with $q = 251$ and $r = 55$ using fixed values for input parameters. Nothing more.

Models that evaluate fixed decision alternatives rather than indicating good choices may be termed **descriptive models**. Evaluating a few specific choices in this way sometimes tells a decision maker (here MM) as much as he or she needs to know. After all, many problems admit only a few practical solutions.

Still, results for a few cases are a very long way from what our **prescriptive** constant-rate demand optimization provided. Section 1.3's closed-form results both recommended optimal choices for q and r and offered insight about the sensitivity of results to changes in parameter values. The simulation did neither.

Principle 1.12 Descriptive models yield fewer analytic inferences than prescriptive, optimization models because they take both input parameters and decisions as fixed.

The tradeoff should now be clear. Resorting to a mathematically structureless form such as a simulation model can substantially improve validity. Still, that same lack of mathematical structure severely limits the possible analysis. More model validity almost always implies less tractability.

1.5 NUMERICAL SEARCH AND EXACT VERSUS HEURISTIC SOLUTIONS

The simulation in Section 1.4 provided a computer program for estimating the total inventory cost associated with a particular choice of reorder point and reorder quantity. Suppose we think of that computation as a function $c(q, r)$. That is, for any given q and r ,

$c(q, r) \triangleq$ total cost computed by the simulation with reorder point fixed at r and reorder quantity at q

Mortimer Middleman's problem then reduces to the mathematical model

$$\begin{aligned} & \text{minimize} && c(q, r) && (1.3) \\ & \text{subject to} && q \geq 100, \quad r \geq 0 \end{aligned}$$

Numerical Search

Since we know very little about the properties of the mathematical function $c(q, r)$, restating our problem in this abstract form lends no immediate insight. Still, it does suggest a way to proceed.

Numerical search is the process of systematically trying different choices for the decision variables, keeping track of the feasible one with the best objective function value found so far. We call such a search numerical because it deals with specific values of the variables rather than with symbolic quantities such as those we were able to manipulate in analyzing the constant-rate demand model.

Here we search numerically over q and r . It seems reasonable to begin with the q and r recommended by our analysis of the constant-rate demand model. Using superscripts (note that these are *not exponents*) to identify specific choices of the decision variables, $q^{(0)} = 251$, $r^{(0)} = 55$, and we have already seen that $c(q^{(0)}, r^{(0)}) = \$108,621$.

Next, we need a systematic process for thinking of new q 's and r 's to try. Much of this book centers on how search processes should be structured. For now, we will try something naively simple: increasing and decreasing one variable at a time in steps of 10.

Table 1.1 showed considerable lost sales, so we start by increasing r to introduce a safety stock. Continuing until the objective function deteriorates yields.

$$\begin{array}{lll} q^{(0)} = 251 & r^{(0)} = 55 & c(q^{(0)}, r^{(0)}) = 108,621 \\ q^{(1)} = 251 & r^{(1)} = 65 & c(q^{(1)}, r^{(1)}) = 108,421 \\ q^{(2)} = 251 & r^{(2)} = 75 & c(q^{(2)}, r^{(2)}) = 63,254 \\ q^{(3)} = 251 & r^{(3)} = 85 & c(q^{(3)}, r^{(3)}) = 63,054 \\ q^{(4)} = 251 & r^{(4)} = 95 & c(q^{(4)}, r^{(4)}) = 64,242 \end{array}$$

All these $(q^{(i)}, r^{(i)})$ are feasible. Proceeding from one of the best, $r = 85$, we now try changing q . Increasing gives

$$q^{(5)} = 261 \quad r^{(5)} = 85 \quad c(q^{(5)}, r^{(5)}) = 95,193$$

and decreasing yields

$$q^{(6)} = 241 \quad r^{(6)} = 85 \quad c(q^{(6)}, r^{(6)}) = 72,781$$

Both are worse than $(q^{(3)}, r^{(3)})$, so we terminate the search.

A Different Start

Our numerical search has discovered a choice of q and r with simulated cost \$63,054, far better than either the \$108,621 of the constant-rate demand solution or MM's \$94,009 actual cost. Lacking any information on how much MM's costs might be reduced, our only way of learning more with numerical search is to try a new search from a different initial point. Consider the search sequence

$$\begin{array}{lll} q^{(0)} = 251 & r^{(0)} = 145 & c(q^{(0)}, r^{(0)}) = 56,904 \\ q^{(1)} = 251 & r^{(1)} = 155 & c(q^{(1)}, r^{(1)}) = 59,539 \\ q^{(2)} = 251 & r^{(2)} = 135 & c(q^{(2)}, r^{(2)}) = 56,900 \\ q^{(3)} = 251 & r^{(3)} = 125 & c(q^{(3)}, r^{(3)}) = 59,732 \\ q^{(4)} = 261 & r^{(4)} = 135 & c(q^{(4)}, r^{(4)}) = 54,193 \\ q^{(5)} = 271 & r^{(5)} = 135 & c(q^{(5)}, r^{(5)}) = 58,467 \end{array}$$

This time, we have happened upon the better heuristic solution $q = 261$, $r = 135$ with simulated cost \$54,193. Certainly, the earlier best of $q = 251$, $r = 85$ was not optimal, but we still have no real reason to believe that our last result is the best achievable.

With only two variables we might try a more exhaustive search, forming a loop and evaluating $c(q, r)$ at an entire grid of points. The essential difficulty would remain, because results would depend on the size of the grid investigated.

Principle 1.13 Inferences from numerical search are limited to the specific points explored unless mathematical structure in the model supports further deduction.

Exact versus Heuristic Optimization

Definition 1.14 An **exact optimal solution** is a feasible solution to an optimization model that is provably as good as any other in objective function value. A **heuristic** or **approximate optimum** is a feasible solution derived from prescriptive analysis that is not guaranteed to yield an exact optimum.

Our numerical searches of MM's problem have produced only a heuristic optimum—a good feasible solution. Should we demand an exact optimum?

As usual, the answer is far from clear. The three “optimal” values of q depicted in Figure 1.6 are all mathematically exact, yet the decisions they recommend vary dramatically with the assumed value of input parameters. Furthermore, our simulation results have shown the true cost to be very different from any of the optimal objective function values computed assuming constant-rate demand.

Principle 1.15 Losses from settling for heuristic instead of exact optimal solutions are often dwarfed by variations associated with questionable model assumptions and doubtful data.

Still, it would clearly make us more certain of whether to recommend the $q = 261, r = 135$ solution, which is the best uncovered in our numerical searches, if we knew something about how far from optimal it might be. Exact optima add a satisfying degree of certainty.

Principle 1.16 The appeal of exact optimal solutions is that they provide both good feasible solutions and certainty about what can be achieved under a fixed set of model assumptions.

1.6 DETERMINISTIC VERSUS STOCHASTIC MODELS

All the searching in Section 1.5 was based on the assumption that Mortimer Middleman's future week-to-week demand will repeat identically the experience of the past year. This is almost certainly untrue. The best we can honestly claim regarding future events is that we know something about the probability of various outcomes.

Definition 1.17 A mathematical model is termed **deterministic** if all parameter values are assumed to be known with certainty, and **probabilistic** or **stochastic** if it involves quantities known only in probability.

Random Variables and Realizations

Random variables represent quantities known only in terms of a probability in stochastic models. We distinguish random variables from ones with single values by using uppercase (capital) letters.

If we do not accept the notion that next year's demands in Mortimer's problem will exactly duplicate last year's, weekly demands are random variable parameters to his decision problem. Each weekly demand will eventually be a specific number. At the time we have to choose a reorder point and reorder quantity; however, we may know only something about their probability distributions.

If the magnitude of each week's random demand—denoted by D_t —is independent of all others (an assumption), each of the values in Figure 1.1(a) is a **realization** of each D_t (i.e., a specific historical outcome). We can learn about the probability distribution of the D_t by counting how often different realizations appear in Figure 1.1(a). The solid line in Figure 1.7 presents such a **frequency histogram**. Demands ranged from 11 to 96, but Figure 1.7 clearly shows that demands in the range 40 to 70 are most common.

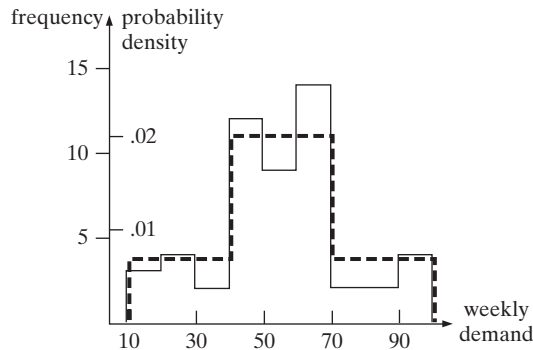


FIGURE 1.7 Weekly MM Demand Frequency Distribution

Eliminating some of the raggedness, the dashed lines in Figure 1.7 show how the true probability distribution for D probably looks. Outcomes in the range 40 to 70 are three times as probable as those in the regions 10–30 and 70–100, but demands are equally likely within regions.

Stochastic Simulation

The simulation model of Sections 1.4 and 1.5 is deterministic because all input parameters, including the week-by-week demands, are assumed known when computations such as Table 1.1 begin. We can develop a stochastic model of Mortimer's problem by improving on that simulation. Assuming that weekly demands D_1, \dots, D_{52} are independent random variables with known probability distribution such as the smoothed one (dashed lines) in Figure 1.7, we will investigate the distribution of the annual cost [now random variable $C(q, r)$] associated with any choice of reorder quantity q and reorder point r .

Stochastic simulation (sometimes called **Monte Carlo analysis**) provides the tool. It samples realizations from output variable distributions by:

1. Randomly generating a sequence of realizations for input parameters.
2. Simulating each realization against chosen values for the decision variables.

With a large enough (and random enough) sample, a frequency histogram of output realizations will approximate the true output distribution.

Figure 1.8 shows results of such stochastic simulation sampling in the Mortimer Middleman problem. A total of 200 different sets of realizations for demand variables D_1, \dots, D_{52} were randomly sampled from the distribution depicted in Figure 1.7. Then each was simulated with $q = 261, r = 135$ just as in Table 1.1, and the annual cost computed as one realization of $C(261, 135)$.

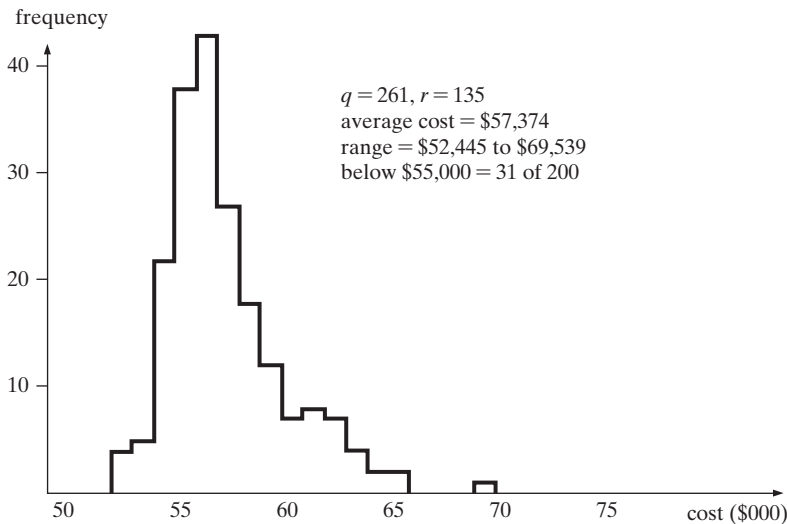


FIGURE 1.8 MM Annual Cost Frequency Distribution

We can see from Figure 1.8 that for this best known choice of decision variables, annual cost has a distribution ranging from \$52,445 to \$69,539, with an average of about \$57,374. Notice that this range of possible futures does include the single value \$54,193 obtained in Section 1.5's numerical searches. Still, many other costs might result, depending on what demand pattern is actually realized.

Stochastic simulation is an effective and widely used OR modeling technique because the enormous flexibility of simulation models makes it possible to formulate validly a very wide range of problem situations. Still, the ragged shape of the frequency histograms in Figure 1.8 highlights the extra analytic challenge associated with computing distributions by simulation. Even with decision variables fixed, and after 200 runs, we know the $C(q, r)$ distributions in Figure 1.8 only approximately. Statistical estimation techniques would be required to determine how much confidence we can have in any conclusions.

Principle 1.18 Besides providing only descriptive analysis, stochastic simulation models impose the extra analytic burden of having to estimate results statistically from a sample of system realizations.

Tradeoffs between Deterministic and Stochastic Models

Few, if any, parameters of real systems can actually be quantified with certainty, yet that is exactly what deterministic models assume. Distributional estimates such as Figure 1.8 usually provide a much more complete picture of what might actually happen with any chosen q and r .

Why not use stochastic models for every problem? It should be no surprise that the answer is tractability. We have already seen how much work is required to estimate the output distribution for a single choice of q and r via stochastic simulation. Other stochastic methods can sometimes produce results with less effort, and validity sometimes demands that stochastic variation be modeled, but a major tractability difference remains.

Principle 1.19 The power and generality of available mathematical tools for analysis of stochastic models does not nearly match that available for deterministic models.

When optimization is the goal, deterministic modeling is often the only practical choice.

Principle 1.20 Most optimization models are deterministic—not because OR analysts really believe that all problem parameters are known with certainty, but because useful prescriptive results can often be obtained only if stochastic variation is ignored.

Of course, deterministic optimization models are also used because they work. We will see in illustrations throughout this book that very satisfactory results have been obtained in a wide variety of applications, often in circumstances that involve many thousands of decision variables.

1.7 PERSPECTIVES

Mortimer Middleman's inventory problem is only one of hundreds of operations design, planning, and control questions that we will encounter in this book. The purpose of treating it in so many different ways here is to introduce the possible approaches and to reveal some of their strengths and weaknesses.

OR analysts must daily decide whether a back-of-the-envelope computation is sufficient or a formal model is required; whether a more detailed and thus more valid model is preferred to a more tractable approximation. One goal of this book is to provide the vocabulary and sensitivity to issues that will help us make those choices.

Other Issues

The chapter's emphasis on the tractability versus validity tradeoff in operations research is not intended to suggest that it is the only issue. Models also differ in how easily they can be understood by the consumer of the analysis (Mortimer here), how fast analysis must be completed, how much mathematical and computer power is required to do the analysis, how many data must be collected, and in a variety of other ways.

More fundamentally, many situations do not lend themselves to the OR approach. Even from this chapter's brief introduction, it should be clear that it is not easy to formulate valid models that are tractable to analysis. Operations research is founded on the conviction, buttressed by a long history of successful practice, that formulation and analysis of mathematical decision models is often worth the trouble. Still, time and resources are required.

Principle 1.21	The model-based OR approach to problem solving works best on problems important enough to warrant the time and resources for a careful study.
----------------	---

The Rest of This Book

In the remainder of this book, the main classes of deterministic optimization models, and the solution techniques available to deal with them, are developed. Chapter 2 begins with formulation and classification of a series of real-world problems in terms of the mathematical properties of the models required. Later chapters address each class in turn, emphasizing formulation of important business and engineering applications and highlighting common elements of the sometimes bewildering variety of analysis techniques used to address various classes.

EXERCISES

1-1 A segment of automatically controlled highway is being equipped with sensors spaced equally along its length. The maximum error in estimating traffic volume, which occurs halfway between any two sensors, can be expressed as $(d/s)^2$, where d is the length of the segment and s is the number of sensors. Each sensor costs p dollars, and designers want to reduce the maximum error as much as possible within a budget for sensors of b dollars. Identify each of the following for this design problem.

- ✓ (a) The decision variable
- ✓ (b) The input parameters
- ✓ (c) The objective function
- ✓ (d) The constraints

1-2 Return to the problem of Exercise 1-1 and assume that $d = 10$, $p = 3.5$, and $b = 14$. Determine (if necessary by trying all the possibilities) whether

each of the following is a feasible and/or optimal solution.

- ✓ (a) $s = 4$
- ✓ (b) $s = 6$
- ✓ (c) $s = 2$

1-3 A factory has two production lines available to make a product. The first can produce one lot of the product in t_1 hours at cost c_1 , and the second requires t_2 hours and cost c_2 . The plant manager wishes to find the least costly way to produce b lots in a total of at most T hours. An integer number x_1 will be produced on line 1, and integer number x_2 on line 2. Identify each of the following for this design problem.

- (a) The decision variables
- (b) The input parameters
- (c) The objective function
- (d) The constraints

1-4 Return to the problem of Exercise 1-3 and assume that $t_1 = 10$, $t_2 = 20$, $c_1 = 500$, $c_2 = 100$, $b = 3$, and $T = 40$. Determine (if necessary by trying all the possibilities) whether each of the following is a feasible and/or optimal solution.

- (a) $x_1 = 0, x_2 = 3$
- (b) $x_1 = 2, x_2 = 1$
- (c) $x_1 = 3, x_2 = 0$

1-5 A university wishes to purchase the maximum number of computer workstations that can be accommodated within the available laboratory floor space and budget. Determine whether each of the following outcomes is most likely the result of closed-form optimization, exact numerical optimization, heuristic optimization, or descriptive modeling.

- ✓ (a) The maximum number of stations that can be fit within 2000 square feet and a budget of \$500,000 is 110.
- (b) The usual arrangement of 80 workstations would require 1600 square feet of floor space and cost \$382,000.
- ✓ (c) The maximum number of stations feasible with an area of f thousand square feet and a budget of b thousand dollars is $\min\{50f, b/5\}$.
- (d) The best of the usual layouts for an area of 2000 square feet and a budget of \$500,000 can accommodate 85 workstations.

1-6 Explain why the first alternative of each pair constitutes more complete analysis of an optimization model than the second.

- (a) Closed—form versus numerical optimization
- (b) Exact optimization versus heuristic optimization
- (c) Heuristic optimization versus simulation of some specific solutions

1-7 Explain why a model that admitted one of the preferred choices in Exercise 1-6 would not necessarily be more appropriate than one allowing only the alternative.

1-8 An engineer is working on a design problem with n parameters to be chosen, and each has 3 possible values. A highly valid descriptive model is available, and he is thinking of choosing the best design simply by using the model to evaluate the effect of every combination of design parameter values.

- (a) Explain why this approach will require running the descriptive model on 3^n combinations of decision-variable values.
- ✓ (b) For $n = 10, 15, 20$, and 30 , compute the time a computer would require to check all the combinations, assuming that it runs 24 hours per day, 365 days per year, and that it requires 1 second to apply the model for any particular choice of design parameters.
- (c) Comment on the practical limits of this “try all combinations” analysis strategy in the light of your results from part (b).

1-9 Determine whether each of the following could probably be validly modeled only as a random variable or if a deterministic quantity would suffice.

- ✓ (a) The number of inches of rainfall in a city over the next 14 days
- (b) The average 14-day rainfall in a city
- ✓ (c) The market price of a common stock 1 week ago
- (d) The market price of a common stock 1 week from today
- ✓ (e) The seating capacity of a restaurant
- (f) The number of customers who will arrive at a restaurant this evening
- ✓ (g) The production rate of an industrial robot subject to frequent breakdowns
- (h) The production rate of a highly reliable industrial robot
- ✓ (i) Factory demand for a model of bulldozer over the next 7 days
- (j) Factory demand for a model of bulldozer over the next 7 years

This page intentionally left blank

Deterministic Optimization Models in Operations Research

With this chapter we begin our detailed study of deterministic models in operations research—models where it is reasonable to assume all problem data to be known with certainty. Few who have ever worked with real models can say the words *known with certainty* without breaking a smile. Input constants in OR models are almost always estimated, some rather crudely. We employ deterministic models because they often produce valid enough results to be useful and because deterministic models are almost always easier to analyze than are their stochastic counterparts.

The increased tractability of deterministic models permits us the luxury of dealing explicitly with optimization. Often, we achieve only an approximation, but the consistent goal is to prescribe the best possible solution.

Deterministic optimization models are also called **mathematical programs** because they decide how to plan or “program” activities. Our treatment is introduced in this chapter with a stream of examples. By formulating models for a variety of cases, we begin developing the modeling skills essential to every OR analyst. At the same time we illustrate the wide range of model forms available and introduce some terminology. Except for a tiny first example, all models presented are based on real applications by real organizations.

2.1 DECISION VARIABLES, CONSTRAINTS, AND OBJECTIVE FUNCTIONS

From the very early days of deterministic optimization, one of its heaviest users has been the petroleum refining industry. Refining operations are routinely planned by formal optimization, often on a daily or even hourly basis. We begin our survey of mathematical programming models with a made-up example from the refining setting, obviously much oversimplified.

APPLICATION 2.1: TWO CRUDE PETROLEUM

Two Crude Petroleum runs a small refinery on the Texas coast. The refinery distills crude petroleum from two sources, Saudi Arabia and Venezuela, into three main products: gasoline, jet fuel, and lubricants.

The two crudes differ in chemical composition and thus yield different product mixes. Each barrel of Saudi crude yields 0.3 barrel of gasoline, 0.4 barrel of jet fuel, and 0.2 barrel of lubricants. On the other hand, each barrel of Venezuelan crude yields 0.4 barrel of gasoline but only 0.2 barrel of jet fuel and 0.3 barrel of lubricants. The remaining 10% of each barrel is lost to refining.

The crudes also differ in cost and availability. Two Crude can purchase up to 9000 barrels per day from Saudi Arabia at \$100 per barrel. Up to 6000 barrels per day of Venezuelan petroleum are also available at the lower cost of \$75 per barrel because of the shorter transportation distance.

Two Crude's contracts with independent distributors require it to produce 2000 barrels per day of gasoline, 1500 barrels per day of jet fuel, and 500 barrels per day of lubricants. How can these requirements be fulfilled most efficiently?

Decision Variables

The first step in formulating any optimization model is to identify the **decision variables**.

Definition 2.1 Variables in optimization models represent the decisions to be taken.

Numerous quantities are floating around in even the very simple Two Crude problem statement. Which do we get to decide? Cost, availability, yield, and requirement are all **input parameters**—quantities that we will take as fixed (see Section 1.3). What must be decided is how much of each crude to refine. Thus we define decision variables

$$\begin{aligned} x_1 &\triangleq \text{barrels of Saudi crude refined per day (in thousands)} \\ x_2 &\triangleq \text{barrels of Venezuelan crude refined per day (in thousands)} \end{aligned} \quad (2.1)$$

Notice that like good modelers in all fields, we have specified both the meaning of each variable and the unit in which it is denominated.

Variable-Type Constraints

The next issue in formulating an optimization model is **constraints**. What limits decisions?

The most elementary constraints declare variable type.

Definition 2.2 **Variable-type constraints** specify the domain of definition for decision variables: the set of values for which the variables have meaning.

For example, variables may be limited to nonnegative values or to nonnegative integer values, or they may be totally unrestricted.

Decision variables x_1 and x_2 in the Two Crude application represent quantities of petroleum refined. Thus they are subject to the most common variable-type constraint form: **nonnegativity**. Every meaningful choice of these decision variables must satisfy

$$x_1, x_2 \geq 0 \quad (2.2)$$

That is, both quantities must be nonnegative real numbers.

It may seem a bit fastidious to specify constraints (2.2) when they are already implicit in definitions (2.1). Still, nothing can be taken for granted because our goal is to produce a form suitable for analysis by computer-based procedures. Formal methods for solving mathematical programs enforce only constraints explicitly expressed in the model formulation.

Main Constraints

The remainder of the limits on decision variable values constitute the main constraints.

Definition 2.3 | **Main constraints** of optimization models specify the restrictions and interactions, other than variable type, that limit decision variable values.

Even in as simple a case as Two Crude Petroleum, we have several main constraints. First consider output requirements. Petroleum volumes selected must meet contract requirements in the sense that

$$\underbrace{\sum (\text{yield per barrel}) (\text{barrels purchased})}_{\text{total output of a product}} \geq \text{product requirements}$$

Quantifying in terms of our decision variables produces one such constraint per product:

$$\begin{aligned} 0.3x_1 + 0.4x_2 &\geq 2.0 && \text{(gasoline)} \\ 0.4x_1 + 0.2x_2 &\geq 1.5 && \text{(jet fuel)} \\ 0.2x_1 + 0.3x_2 &\geq 0.5 && \text{(lubricants)} \end{aligned} \quad (2.3)$$

Each is denominated in thousands of barrels per day.

Availabilities yield the other class of main constraints. We can buy no more than 9000 barrels of Saudi crude per day, nor 6000 barrels of Venezuelan. Corresponding constraints are

$$\begin{aligned} x_1 &\leq 9 && \text{(Saudi)} \\ x_2 &\leq 6 && \text{(Venezuelan)} \end{aligned} \quad (2.4)$$

Objective Functions

Objective or **criterion functions** tell us how to rate decisions.

Definition 2.4 | **Objective functions** in optimization models quantify the decision consequences to be maximized or minimized.

What makes one choice of decision variable values preferable to another in the Two Crude case? Cost. The best solution will seek to minimize total cost:

$$\min \sum (\text{crude unit cost}) (\text{barrels purchased})$$

Quantifying this single objective in terms of the decision variables yields

$$\min 100x_1 + 75x_2 \tag{2.5}$$

(in thousands of dollars per day).

Standard Model

Once decision variables have been defined, constraints detailed, and objectives quantified, the mathematical programming model we require is complete. Still, it is customary to follow variable and parameter definitions with a summary of the model in a standard format.

Principle 2.5	The standard statement of an optimization model has the form
	$\begin{aligned} \min \text{ or } \max & \quad (\text{objective function(s)}) \\ \text{s.t.} & \quad (\text{main constraints}) \\ & \quad (\text{variable-type constraints}) \end{aligned}$
	where “s.t.” stands for “subject to.”

Combining (2.2)–(2.5), we may formalize our Two Crude model as follows:

$$\begin{aligned} \min & \quad 100x_1 + 75x_2 && \quad (\text{total cost}) \\ \text{s.t.} & \quad 0.3x_1 + 0.4x_2 \geq 2.0 && \quad (\text{gasoline requirement}) \\ & \quad 0.4x_1 + 0.2x_2 \geq 1.5 && \quad (\text{jet fuel requirement}) \\ & \quad 0.2x_1 + 0.3x_2 \geq 0.5 && \quad (\text{lubricant requirement}) \\ & \quad x_1 \leq 9 && \quad (\text{Saudi availability}) \\ & \quad x_2 \leq 6 && \quad (\text{Venezuelan availability}) \\ & \quad x_1, x_2 \geq 0 && \quad (\text{nonnegativity}) \end{aligned} \tag{2.6}$$

EXAMPLE 2.1: FORMULATING FORMAL OPTIMIZATION MODELS

Suppose that we wish to enclose a rectangular equipment yard by at most 80 meters of fencing. Formulate an optimization model to find the design of maximum area.

Solution: The decisions required are the dimensions of the rectangle. Thus define decision variables:

$$\begin{aligned} \ell & \triangleq \text{length of the equipment yard (in meters)} \\ w & \triangleq \text{width of the equipment yard (in meters)} \end{aligned}$$

Both variables are nonnegative quantities, which implies variable-type constraints:

$$\ell, w \geq 0$$

The only main constraint is that the perimeter of the equipment yard should not exceed 80 meters in length. In terms of decision variables, that limitation can be expressed as

$$2\ell + 2w \leq 80$$

Our objective is to maximize the enclosed area ℓw . Thus a complete model statement is as follows:

$$\begin{array}{lll} \max & \ell w & \text{(enclosed area)} \\ \text{s.t.} & 2\ell + 2w \leq 80 & \text{(fence length)} \\ & \ell, w \geq 0 & \text{(nonnegativity)} \end{array}$$

2.2 GRAPHIC SOLUTION AND OPTIMIZATION OUTCOMES

Methods for analyzing optimization models are the focus of many chapters to follow. However, very simple graphic techniques have enough power to deal with tiny models such as Two Crude formulation (2.6). They also provide “pictures” yielding helpful intuition about properties and solution methods for models of more realistic size.

In this section we develop the techniques of graphic solution. We also illustrate their intuitive power by exploring the unique optimal solution, alternative optimal solution, and infeasible and unbounded outcomes of optimization analysis.

Graphic Solution

Graphic solution solves 2- and 3-variable optimization models by plotting elements of the model in a coordinate system corresponding to the decision variables. For example, Two Crude model (2.6) involves decision variables x_1 and x_2 . Every choice of values for these variables corresponds to a point (x_1, x_2) in a 2-dimensional plot.

Feasible Sets

The first issue in graphic solution is the feasible set (also called the feasible region or the feasible space).

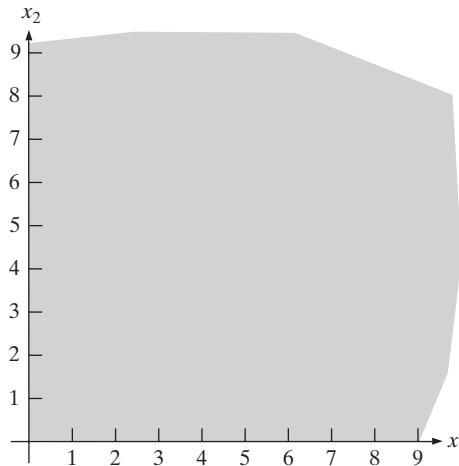
Definition 2.6 The **feasible set** (or **region**) of an optimization model is the collection of choices for decision variables satisfying all model constraints.

Graphing Constraints and Feasible Sets

We want to draw a picture of the feasible set in a coordinate system defined by the decision variables. The process begins with the variable-type constraints (definition [2.2](#)).

Principle 2.7 Graphic solution begins with a plot of the choices for decision variables that satisfy variable-type constraints.

In the Two Crude application, both variables are nonnegative. Thus every feasible solution corresponds to a point in the shaded, nonnegative part of the following plot:



Next we introduce the main constraints (definition [2.3](#)) one at a time. Begin with the Two Crude gasoline requirement

$$0.3x_1 + 0.4x_2 \geq 2 \quad (2.7)$$

If it were of the equality form

$$0.3x_1 + 0.4x_2 = 2$$

feasible points would lie along a corresponding line.

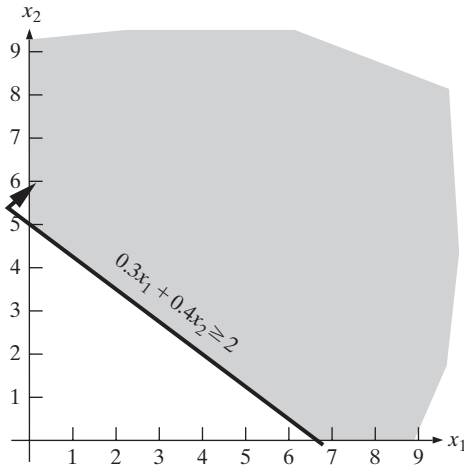
Principle 2.8 The set of points satisfying an equality constraint plots as a line or curve.

However, constraint (2.7) is an inequality.

Principle 2.9 The set of points satisfying an inequality constraint plots as a boundary line or curve, where the constraint holds with equality, together with all points on whichever side of the boundary satisfy the constraint as an inequality (we add a small arrow at the end of the equality line to indicate the feasible side).

To identify feasible points, we first plot the boundary and then include whichever side of the line applies.

Over Two Crude's nonnegative (x_1, x_2) , the result is



Points along the highlighted boundary satisfy the constraint with equality. To determine which side of the line depicts points satisfying the constraint as a strict inequality, we have only to substitute one point not on the line. For example, the origin $x_1 = x_2 = 0$ violates the constraint because

$$0.3x_1 + 0.4x_2 = 0.3(0) + 0.4(0) \neq 2$$

Feasible points must be on the other side of the equality line, just as the arrow on the constraint indicates.

Feasible (x_1, x_2) must satisfy all constraints simultaneously. To identify the feasible set fully, we merely repeat the foregoing process.

Principle 2.10 The feasible set (or region) for an optimization model is plotted by introducing constraints one by one, keeping track of the region satisfying all at the same time.

Figure 2.1 shows the result for our Two Crude application. Each of the 5 main inequality constraints yields a line, where it is satisfied as an equality, plus a side of the line where it holds as an inequality. Together with variable-type constraints $x_1, x_2 \geq 0$, these define the shaded feasible set.

EXAMPLE 2.2: GRAPHING CONSTRAINTS AND FEASIBLE SETS

Graph the feasible sets corresponding to each of the following systems of constraints.

- | | | |
|------------------------|------------------------|--------------------------------|
| (a) $x_1 + x_2 \leq 2$ | (b) $x_1 + x_2 \leq 2$ | (c) $(x_1)^2 + (x_2)^2 \leq 4$ |
| $3x_1 + x_2 \geq 3$ | $3x_1 + x_2 = 3$ | $ x_1 - x_2 \leq 0$ |
| $x_1, x_2 \geq 0$ | $x_1, x_2 \geq 0$ | |

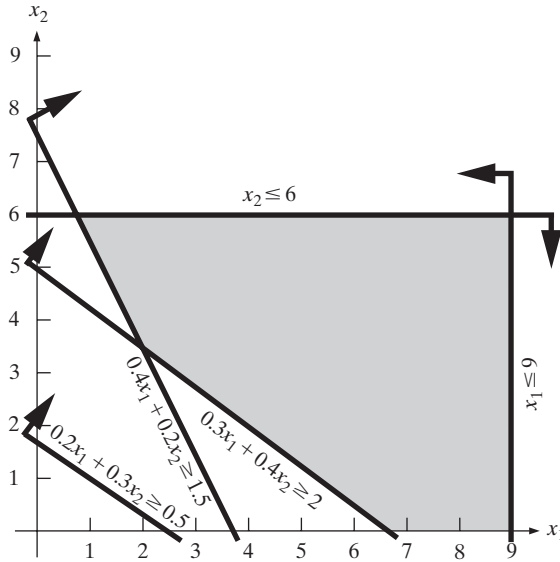
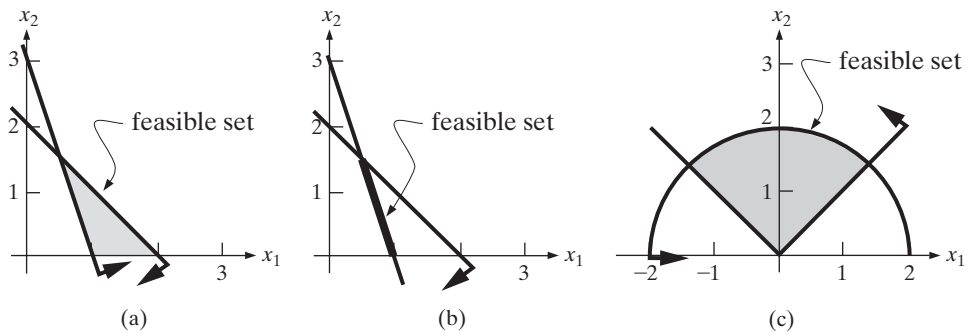


FIGURE 2.1 Feasible Set for the Crude Application

Solution: Applying the process of [2.10] produces the following plots:



Notice that the feasible set for system (b) is just the highlighted line segment, because one constraint is an equality. Also, system (c) allows negative variable values because the variable type is unrestricted.

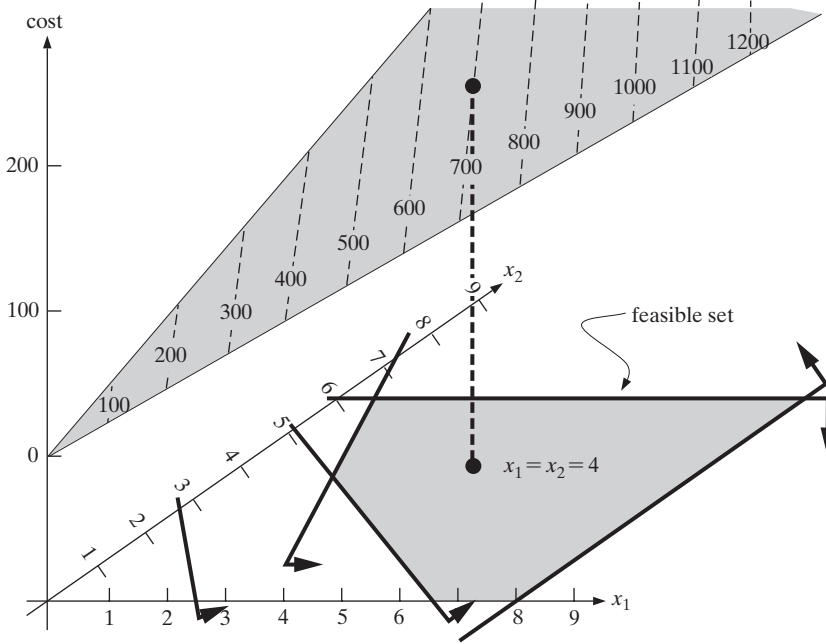
Graphing Objective Functions

To find the best feasible point, we must introduce the objective function into a plot like Figure 2.1. Observe that the objective in our Two Crude application, call it

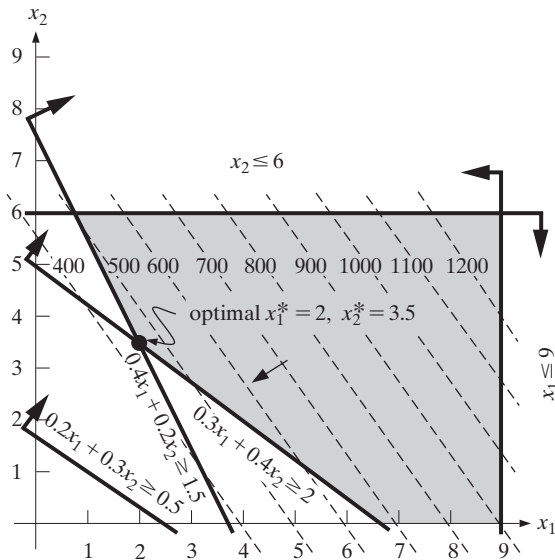
$$c(x_1, x_2) \triangleq 100x_1 + 75x_2 \tag{2.8}$$

is a function of decision variables x_1 and x_2 . Thus a plot requires a third dimension.

Figure 2.2(a) depicts surface (2.8) in an x_1 versus x_2 versus c coordinate system. For example, feasible solution $x_1 = x_2 = 4$ has value $c(4, 4) = 100(4) + 75(4) = 700$ on the objective function surface.



(a) Three-dimensional view



(b) Two-dimensions with contours

FIGURE 2.2 Graphic Solution of the Two Crude Model

Because most of us have trouble visualizing, much less drawing 3-dimensional plots, mathematical programmers customarily employ the 2-dimensional alternative of Figure 2.2(b). There, the third dimension, cost, is shown through contours.

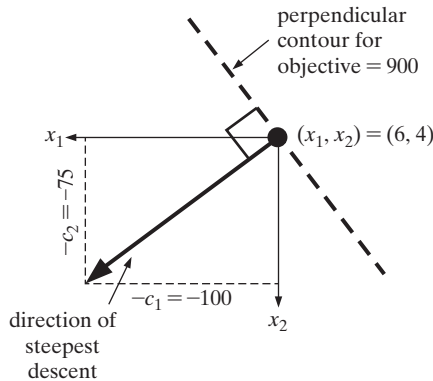
Principle 2.11 Objective functions are normally plotted in the same coordinate system as the feasible set of an optimization model by introducing **contours**, which are lines or curves (typically shown as dashed lines or curves) through choices of decision variables with the same objective function value and running perpendicular to the direction of steepest objective improvement at points along the line.

One way to introduce contours in graph solution plots is to begin at any convenient point visible in the plot, evaluate the objective there, and find the set of points with the same objective value. For example, we might pick $x_1 = 9$ and $x_2 = 0$ in the Two Crude model, and evaluate objective function (2.8) as

$$100x_1 + 75x_2 = 100(9) + 78(0) = 900$$

Then plotting the line $100x_1 + 75x_2 = 900$ yields the 900 contour of Figure 2.2(b).

An equivalent way is to pick an arbitrary feasible point, say $x_1 = 6$ and $x_2 = 4$ with objective value which happens also to = 900. There, we can construct the direction of steepest improvement (decrease) in our minimize objective function from the objective coefficients as shown below.



Combining a decrease of 100 for each unit step in x_1 and a decrease of 75 for each step in x_2 gives the steepest descent direction shown. The contour at point $(6, 4)$ will be perpendicular to that direction.

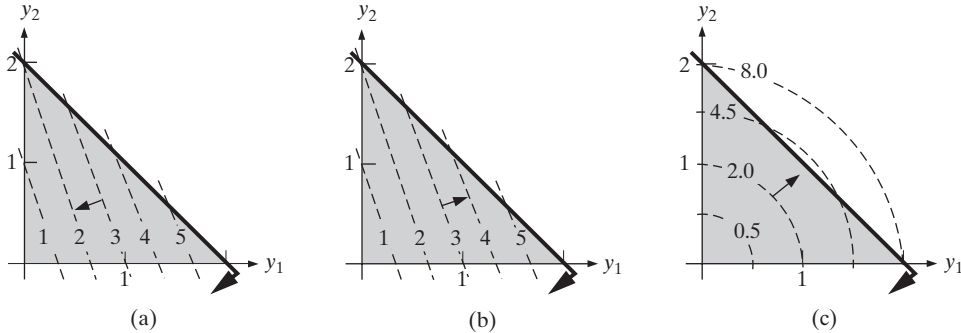
Other contours for this objective will have the same steepest descent direction and thus parallel to this one. There is a contour for every objective value, but we need only sketch a few to identify the pattern. Figure 2.2 illustrates with contours for each change of \$100 thousand in objective value.

EXAMPLE 2.3: PLOTTING OBJECTIVE FUNCTION CONTOURS

Show contours of each of the following objective functions over the feasible region defined by $y_1 + y_2 \leq 2, y_1, y_2 \geq 0$.

- (a) $\min 3y_1 + y_2$ (b) $\max 3y_1 + y_2$ (c) $\max 2(y_1)^2 + 2(y_2)^2$

Solution: Contours for the three cases are as follows:



(a) After identifying the feasible region by plotting the main and variable-type constraints, we arbitrarily pick $y_1 = 0, y_2 = 1$ to begin introducing the objective function as in principle 2.6. The corresponding contour is the line where

$$3y_1 + y_2 = 3(0) + (1) = 1$$

Then other contours are introduced by incrementing the level to 2, 3, ..., 5.

(b) Contours for this maximize model are identical to those minimizing the same objective function in part (a). A small arrow on one of the contours reminds us of the direction in which the objective function improves. Contours are perpendicular to those arrows.

(c) For this more complex objective function, contours will not plot as straight lines. We begin with $y_1 = 0, y_2 = 1$, and graph curve

$$2(y_1)^2 + 2(y_2)^2 = 2(0)^2 + 2(1)^2 = 2$$

Then other contours arise from trying nearby levels 0.5, 4.5, and 8.0.

Optimal Solutions

Our goal in graphic analysis of an optimization model is to identify an optimal solution if there is one.

Definition 2.12 An **optimal solution** is a feasible choice for decision variables with objective function value at least equal to that of any other solution satisfying all constraints.

We have already determined that the feasible points in Figure 2.2(b) are the (x_1, x_2) in the shaded area. To identify an optimal solution, we examine the objective function contours.

Principle 2.13 Optimal solutions show graphically as points lying on the best objective function contour that intersects the feasible region.

Our Two Crude model is of minimize form, so the best objective contour is the lowest. We can identify an optimal solution by finding a feasible point on the lowest possible contour.

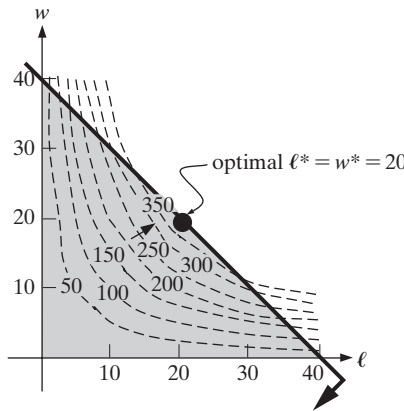
Only a few contour lines are explicitly displayed in Figure 2.2(b), but infinitely many exist. It is clear from the pattern of those shown that the unique point satisfying condition 2.13 is $(x_1^*, x_2^*) = (2, 3.5)$. The asterisk (*) or “star” denotes an optimal solution.

This completes graphic solution of the Two Crude model. The optimal operating plan uses 2 thousand barrels per day of Saudi petroleum and 3.5 thousand barrels per day of Venezuelan. Total daily cost will be $100(2) + 75(3.5) = 462.5$ thousand dollars.

EXAMPLE 2.4: SOLVING OPTIMIZATION MODELS GRAPHICALLY

Return to the equipment yard model of Example 2.1 and solve it graphically.

Solution: Identifying the feasible set and introducing objective function contours produces the following plot:



For this maximize problem we wish to be on the highest possible contour. Thus application of principle 2.7 yields optimal solution $l^* = w^* = 20$.

Optimal Values

Optimal solutions provide a best choice for decision variables and the optimal value is the corresponding objective function level.

Definition 2.14 The **optimal value** in an optimization model is the objective function value of any optimal solution.

For example, the Two Crude optimum of Figure 2.2(b) has optimal value \$462,500. Notice that two different objective values could not both be best.

Principle 2.15 An optimization model can have only one optimal value.

We see this graphically because there can only be one best contour level in rule 2.13.

Unique versus Alternative Optimal Solutions

The optimal solution $x_1^* = 2, x_2^* = 3.5$ in our Two Crude application is also unique because it is the only feasible solution in Figure 2.2(b) achieving the optimal value. This does not always happen. Many models have alternative optimal solutions.

Principle 2.16 An optimization model may have a **unique optimal solution** or several **alternative optimal solutions**.

Still, all must have the same optimal value.

We can illustrate principle 2.9 very easily with another graphic solution. Suppose that crude prices change in the Two Crude application to produce objective function

$$\min 100x_1 + 50x_2$$

Figure 2.3 shows the impact. There is still only one optimal value, \$375 thousand. But there are now an infinite number of alternative optimal solutions along the highlighted boundary of the feasible set; all lie on the optimal \$375 thousand contour.

Principle 2.17 Unique optimal solutions show graphically by the optimal-value contour intersecting the feasible set at exactly one point. If the optimal-value contour intersects at more than one point, the model has alternative optimal solutions.

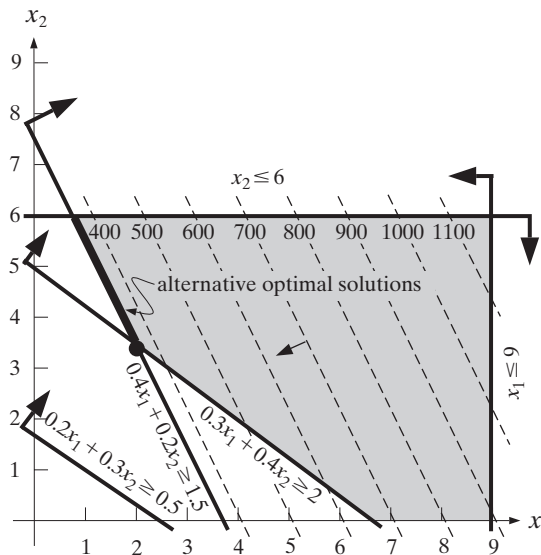


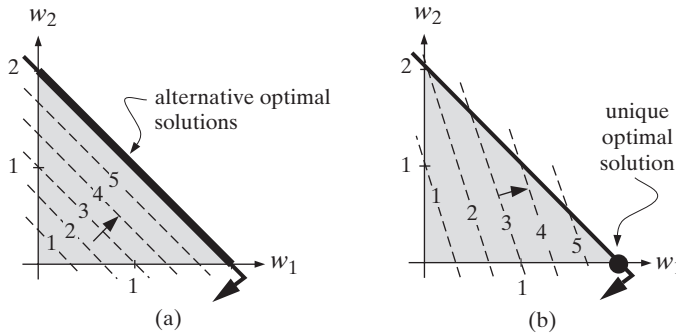
FIGURE 2.3 Variant of the Two Crude Model With Alternative Optima

EXAMPLE 2.5: IDENTIFYING UNIQUE AND ALTERNATIVE OPTIMAL SOLUTIONS

Determine graphically which of the following optimization models has a unique optimal solution and which has alternative optima.

<p>(a) $\max \quad 3w_1 + 3w_2$ s.t. $w_1 + w_2 \leq 2$ $w_1, w_2 \geq 0$</p>	<p>(b) $\max \quad 3w_1 + 3w_2$ s.t. $w_1 + w_2 \leq 2$ $w_1, w_2 \geq 0$</p>
--	--

Solution: Graphic solution of these models is as follows:



Model (a) has alternative optimal solutions along the highlighted boundary, including $(w_1, w_2) = (0, 2)$, $(w_1, w_2) = (1, 1)$, and $(w_1, w_2) = (2, 0)$. Model (b) has unique optimal solution $(w_1, w_2) = (2, 0)$ because the optimal-value contour intersects the feasible space at only one point.

Infeasible Models

Infeasible models have no optimal solutions.

Definition 2.18 | An optimization model is **infeasible** if no choice of decision variables satisfies all constraints.

Such models have no optimal solutions because they have no solutions at all.

Infeasibility is also easy to illustrate with models small enough to be analyzed graphically. For example, consider modifying our Two Crude case so that only 2 thousand barrels per day are available from each source. The resulting model is

\min	$100x_1 + 75x_2$	(total cost)
s.t.	$0.3x_1 + 0.4x_2 \geq 2.0$	(gasoline requirement)
	$0.4x_1 + 0.2x_2 \geq 1.5$	(jet fuel requirement)
	$0.2x_1 + 0.3x_2 \geq 0.5$	(lubricant requirement)
	$x_1 \leq 2$	(Saudi availability)
	$x_2 \leq 2$	(Venezuelan availability)
	$x_1, x_2 \geq 0$	(nonnegativity)

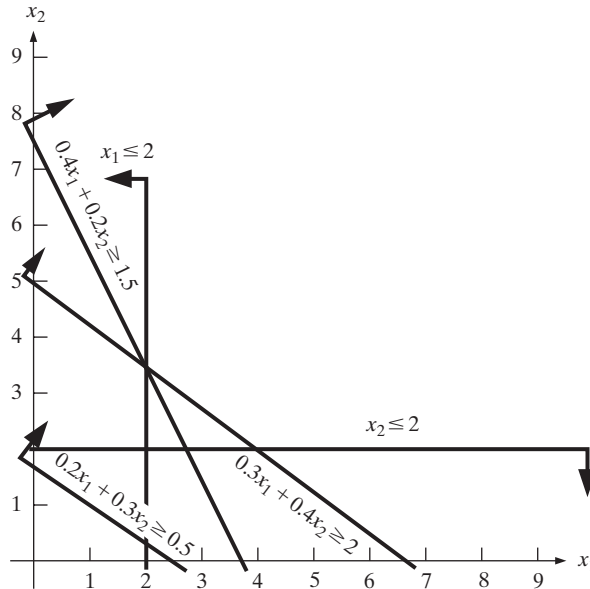


FIGURE 2.4 Infeasible of the Two Crude Model

An attempt to graph the feasible space produces Figure 2.4. As before, each constraint corresponds to a line where it holds as an equality and a side of the line satisfying it as an inequality. This time, however, there are no (x_1, x_2) satisfying all constraints simultaneously.

Principle 2.19 An infeasible model shows graphically by no point falling within the feasible region for all constraints.

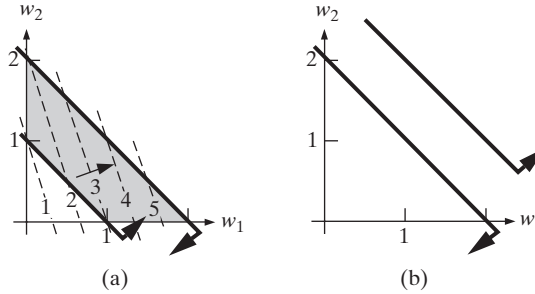
EXAMPLE 2.6: IDENTIFYING INFEASIBLE MODELS GRAPHICALLY

Determine graphically which of the following optimization models is feasible and which is infeasible.

(a) $\max \quad 3w_1 + w_2$
 s.t. $w_1 + w_2 \leq 2$
 $w_1 + w_2 \geq 1$
 $w_1, w_2 \geq 0$

(b) $\max \quad 3w_1 + w_2$
 s.t. $w_1 + w_2 \leq 2$
 $w_1 + w_2 \geq 3$
 $w_1, w_2 \geq 0$

Solution: Graphic solution of these models is as follows:



Model (a) has points in the shaded area that satisfy all constraints. Thus it is feasible. No points satisfy all constraints in model (b). That model is infeasible.

Unbounded Models

Another case where an optimization model has no feasible solution arises when it is unbounded.

Definition 2.20 An optimization model is **unbounded** when feasible choices of the decision variables can produce arbitrarily good objective function values.

Unbounded models have no optimal solutions because any possibility can be improved.

We can illustrate this outcome graphically with still another variant of the Two Crude case (although not a very realistic one). Suppose that Saudi Arabia decides to subsidize oil prices heavily so that Two Crude is paid \$10 for each barrel it consumes, and further that Saudi Arabia will supply unlimited amounts of petroleum at this negative price. The result is a revised model

$$\begin{aligned}
 \min \quad & -10x_1 + 75x_2 && \text{(total cost)} \\
 \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2.0 && \text{(gasoline requirement)} \\
 & 0.4x_1 + 0.2x_2 \geq 1.5 && \text{(jet fuel requirement)} \\
 & 0.2x_1 + 0.3x_2 \geq 0.5 && \text{(lubricant requirement)} \\
 & x_2 \leq 6 && \text{(Venezuelan availability)} \\
 & x_1, x_2 \geq 0 && \text{(nonnegativity)}
 \end{aligned}$$

Figure 2.5 shows its graphic solution. Notice that as x_1 (Saudi purchases) increases, we encounter feasible solutions with ever better objective function values. No solution is optimal because a better one can always be found.

This is exactly what it means to be unbounded.

Principle 2.21 Unbounded models show graphically by there being points in the feasible set lying on ever-better objective function contours.

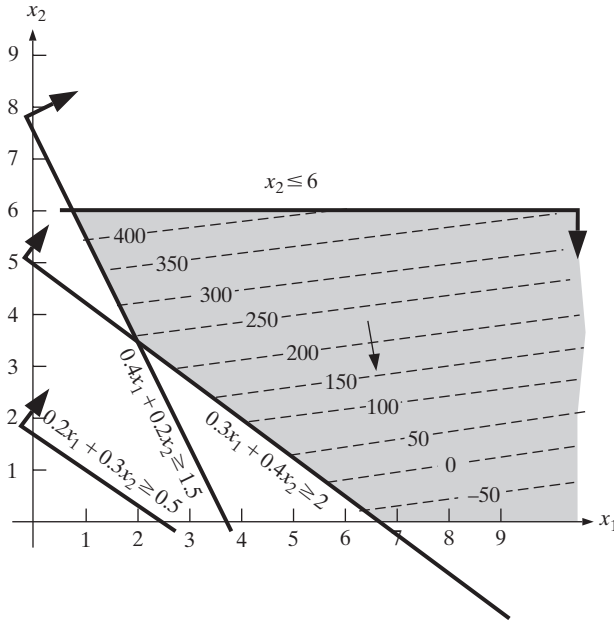


FIGURE 2.5 Unbounded Variant of the Two Crude Example

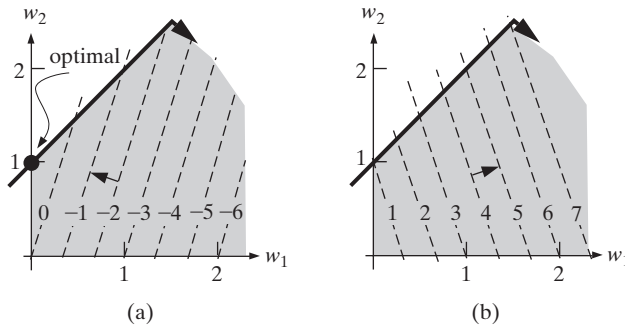
EXAMPLE 2.7: IDENTIFYING UNBOUNDED MODELS GRAPHICALLY

Determine graphically which of the following optimization models has an optimal solution and which is unbounded.

(a) $\max \quad -3w_1 + w_2$
 s.t. $-w_1 + w_2 \leq 1$
 $w_1, w_2 \geq 0$

(b) $\max \quad 3w_1 + w_2$
 s.t. $-w_1 + w_2 \leq 1$
 $w_1, w_2 \geq 0$

Solution: Graphic solution of these models is as follows:



Model (a) has a unique optimal solution at $(w_1, w_2) = (0, 1)$. Model (b) is unbounded because we can find feasible solutions on arbitrarily good contours of the objective function.

2.3 LARGE-SCALE OPTIMIZATION MODELS AND INDEXING

Because it could be solved graphically, our tiny Two Crude Petroleum application facilitated introduction of a variety of mathematical programming concepts. Still, it seriously misrepresents problem size. In real applications optimization models quickly grow to thousands, even millions, of variables and constraints. We begin to see how in this section and introduce the indexed notational schemes that keep large models manageable.

APPLICATION 2.2: PI HYBRIDS

To illustrate, we introduce our first real application of operations research.¹ A large manufacturer of corn seed, which we will call Pi Hybrids, operates ℓ facilities producing seeds of m hybrid corn varieties and distributes them to customers in n sales regions. They want to know how to carry out these production and distribution operations at minimum cost.

After some effort, a variety of parameters that we will take as constant have been estimated:

- The cost per bag of producing each hybrid at each facility
- The corn processing capacity of each facility in bushels
- The number of bushels of corn that must be processed to make a bag of each hybrid
- The number of bags of each hybrid demanded in each customer region
- The cost per bag of shipping each hybrid from each facility to each customer region

Our task is to produce a suitable optimization model.

Indexing

Indexes or **subscripts** permit representing collections of similar quantities with a single symbol. For example,

$$\{z_i : i = 1, \dots, 100\}$$

represents 100 similar values with the same z name, distinguishing them with the index i .

Indexing is a powerful aid in keeping large-scale optimization models easy to think about and concise to write down. In fact, it often provides the initial model organization.

Principle 2.22 The first step in formulating a large optimization model is to choose appropriate indexes for the different dimensions of the problem.

¹Based on M. Zuo, W. Kuo, and K. L. McRoberts (1991), "Application of Mathematical Programming to a Large-Scale Agricultural Production and Distribution System," *Journal of the Operational Research Society*, 42, 639–648.

Our Pi Hybrids application has three main dimensions: facilities, hybrids, and sales regions. Thus we begin formulation of a model by assigning an index to each:

$f \triangleq$ production facility number ($f = 1, \dots, \ell$)

$h \triangleq$ hybrid variety number ($h = 1, \dots, m$)

$r \triangleq$ sales region number ($r = 1, \dots, n$)

Indexed Decision Variables

We are now ready to think about what decisions Pi Hybrids' problem requires. Clearly, they fall into two categories: how much to produce and how much to ship.

There is a production decision for each plant and each hybrid. We could sequentially number the corresponding decision variables $1, \dots, km$, but it would then be difficult to keep plant and hybrid issues separate. It is much more convenient to employ decision variables with distinct subscripts for plant and hybrids:

$x_{f,h} \triangleq$ number of bags of hybrid h produced at facility f
 $(f = 1, \dots, \ell; h = 1, \dots, m)$

Multiple subscripts are extremely common in large OR models.

Principle 2.23 It is usually appropriate to use separate indexes for each problem dimension over which a decision variable or input parameter is defined.

Shipping at Pi Hybrid is distinguished by the production facility shipped from, the hybrid variety shipped, and the sales region shipped to. Following principle [2.14](#), we complete the list of decision variables with a 3-subscript family:

$y_{f,h,r} \triangleq$ number of bags of hybrid h shipped from facility f to sales region r
 $(f = 1, \dots, \ell; h = 1, \dots, m; r = 1, \dots, n)$

Notice that indexing has made it possible to define quite a large number of variables with just two main symbols. Taking $l = 20$, $m = 25$, and $n = 30$, there are $\ell m = 20(25) = 500$ x -variables and $\ell mn = 20(25)(30) = 15,000$ y variables. The 15,500 total dwarfs the tiny examples of preceding sections, but it represents only a rather average size for applied models.

EXAMPLE 2.8: COUNTING INDEXED DECISION VARIABLES

Suppose that an optimization model employs decision variables $w_{i,j,k,\ell}$, where i and k range over $1, \dots, 100$, while j and ℓ index through $1, \dots, 50$. Compute the total number of decision variables.

Solution: The number of variables is

$$\begin{aligned} & (\text{number of } i) (\text{number of } j) (\text{number of } k) (\text{number of } \ell) \\ &= 100(50)(100)(50) \\ &= 25,000,000 \end{aligned}$$

Indexed Symbolic Parameters

If Pi Hybrids made only 2 products at 2 facilities, we could easily express total production cost as

$$\begin{pmatrix} \text{cost of} \\ \text{hybrid} \\ 1 \text{ at} \\ \text{facility} \\ 1 \end{pmatrix} x_{1,1} + \begin{pmatrix} \text{cost of} \\ \text{hybrid} \\ 2 \text{ at} \\ \text{facility} \\ 1 \end{pmatrix} x_{1,2} + \begin{pmatrix} \text{cost of} \\ \text{hybrid} \\ 1 \text{ at} \\ \text{facility} \\ 2 \end{pmatrix} x_{2,1} + \begin{pmatrix} \text{cost of} \\ \text{hybrid} \\ 2 \text{ at} \\ \text{facility} \\ 2 \end{pmatrix} x_{2,2} \quad (2.9)$$

using the actual cost values. But for $\ell = 20, m = 25$ writing out the $\ell m = 20(25) = 500$ production cost terms actually required in our model would be bulky and almost impossible to read or explain.

The answer to this dilemma is more indexing, this time on input parameters.

Principle 2.24 To describe large-scale optimization models compactly it is usually necessary to assign indexed symbolic names to most input parameters, even though they are being treated as constant.

For example, after defining

$$p_{f,h} \triangleq \text{cost per bag of producing hybrid } h \text{ at facility } f$$

we may employ **summation notation** to express form (2.9) for any number of facilities and hybrids as

$$\sum_{f=1}^{\ell} \sum_{h=1}^m p_{f,h} x_{f,h}$$

Moving indexes in the double sum capture terms for all combinations of facilities $f = 1, 2, \dots, \ell$ and hybrids $h = 1, 2, \dots, m$.

EXAMPLE 2.9: USING SUMMATION NOTATION

(a) Write the following sum more compactly with summation notation:

$$2w_{1,5} + 2w_{2,5} + 2w_{3,5} + 2w_{4,5} + 2w_{5,5}$$

(b) Write out terms separately of the sum

$$\sum_{i=1}^4 iw_i$$

Solution:

(a) Introducing moving index i for the first subscript, the sum is

$$\sum_{i=1}^5 2w_{i,5} = 2 \sum_{i=1}^5 w_{i,5}$$

(b) The 4 terms of the sum are

$$1w_1 + 2w_2 + 3w_3 + 4w_4$$

Objective Functions

For similar reasons of convenience, define the following symbolic names for other Pi Hybrids input parameters:

$u_f \triangleq$ corn processing capacity of facility f in bushels

$a_h \triangleq$ number of bushels of corn that must be processed to obtain a bag of hybrid h

$d_{h,r} \triangleq$ number of bags of hybrid h demanded in sales region r

$s_{f,h,r} \triangleq$ cost per bag of shipping hybrid h from facility f to sales region r

With the aid of our indexed decision variables and these indexed parameters, we are now ready to formulate the objective function of a Pi Hybrids model. It should minimize

$$\text{total cost} = \text{total production cost} + \text{total shipping cost}$$

or

$$\min \sum_{f=1}^{\ell} \sum_{h=1}^m p_{f,h} x_{f,h} + \sum_{f=1}^{\ell} \sum_{h=1}^m \sum_{r=1}^n s_{f,h,r} y_{f,h,r}$$

Indexed Families of Constraints

Turn now to constraints for the Pi Hybrids model. What restrictions must decision variables satisfy?

One family of constraints must enforce production capacities at the various facilities. Capacities u_f are measured in bushels processed, with a_h needed for each bag of hybrid h . Thus we may express the capacity requirement at each facility 1 by

$$\sum_{h=1}^m \left(\begin{array}{l} \text{bushels} \\ \text{per bag of} \\ h \end{array} \right) \left(\begin{array}{l} \text{bags of } h \\ \text{produced} \\ \text{at 1} \end{array} \right) \leq \text{capacity at 1}$$

or

$$\sum_{h=1}^m a_h x_{1,h} \leq u_1$$

With $\ell = 20$ there are 20 such capacity constraints for different facilities. We could write each one down explicitly, but again, the model would become very bulky and hard to comprehend.

Standard mathematical programming notation deals with this difficulty by listing indexed families of constraints.

Principle 2.25 Families of similar constraints distinguished by indexes may be expressed in a single-line format

$$(\text{constraint for fixed indexes}) (\text{ranges of indexes})$$

which implies one constraint for each combination of indexes in the ranges specified.

Written in this style, all capacity constraints of the Pi Hybrids model can be expressed in the single line

$$\sum_{h=1}^m a_h x_{f,h} \leq u_f \quad f = 1, \dots, \ell$$

Separate constraints are implied for $f = 1, 2, \dots, \ell$. Equivalent forms are

$$\sum_{h=1}^m a_h x_{f,h} \leq u_f \quad \text{for all } f$$

and

$$\sum_{h=1}^m a_h x_{f,h} \leq u_f \quad \forall f$$

because we know that f ranges from 1 through ℓ and the mathematical symbol \forall means “for all”.

EXAMPLE 2.10: USING INDEXED FAMILIES OF CONSTRAINTS

An optimization model must decide how to allocate available supplies s_i at sources $i = 1, \dots, p$ to meet requirements r_j at customers $j = 1, \dots, q$. Using decision at variables

$$w_{i,j} \triangleq \text{amount allocated from source } i \text{ to customer } j$$

formulate each of the following requirements in a single line.

- (a) The amount allocated from source 32 cannot exceed the supply available at 32.
- (b) The amount allocated from each source i cannot exceed the supply available at i .
- (c) The amount allocated to customer n should equal the requirement at n .
- (d) The amount allocated to each customer j should equal the requirement at j .

Solution:

- (a) Only one constraint is required:

$$\sum_{j=1}^n w_{32,j} \leq s_{32}$$

- (b) Here we require constraints for all sources i . Using notation [2.25], all m can be expressed:

$$\sum_{j=1}^n w_{i,j} \leq s_i \quad i = 1, \dots, m$$

- (c) As with part (a) there is only one constraint:

$$\sum_{i=1}^m w_{i,n} = r_n$$

(d) This requirement implies constraints for each demand j . Using notation [2.25](#), all n can be expressed

$$\sum_{i=1}^m w_{i,j} = r_j \quad j = 1, \dots, n$$

EXAMPLE 2.11: COUNTING INDEXED CONSTRAINTS

Determine the number of constraints in the following systems.

(a) $\sum_{i=1}^{22} z_{i,3} \geq b_3$

(b) $\sum_{i=1}^{22} z_{i,p} \geq b_p, \quad p = 1, \dots, 45$

(c) $\sum_{k=1}^{10} z_{i,j,k} \leq g_j, \quad i = 1, \dots, 14; \quad j = 1, \dots, 30$

Solution:

(a) This expression represents only 1 constraint, associated with fixed index 3.

(b) This expression represents 45 constraints, one for each p .

(c) This expression represents $14(30) = 420$ constraints, one for each i and each j .

Pi Hybrids Application Model

In terms of the notation we have defined, a full model for Pi Hybrids' production–distribution problem is

$$\begin{aligned} \min \quad & \sum_{f=1}^{\ell} \sum_{h=1}^m p_{f,h} x_{f,h} + \sum_{f=1}^{\ell} \sum_{h=1}^m \sum_{r=1}^n s_{f,h,r} y_{f,h,r} && \text{(total cost)} \\ \text{s.t.} \quad & \sum_{h=1}^m a_h x_{f,h} \leq u_f \quad f = 1, \dots, \ell && \text{(capacity)} \\ & \sum_{f=1}^{\ell} y_{f,h,r} = d_{f,h} \quad h = 1, \dots, m; \quad r = 1, \dots, n && \text{(demands)} \quad (2.10) \\ & \sum_{r=1}^n y_{f,h,r} = x_{f,h} \quad f = 1, \dots, \ell; \quad h = 1, \dots, m && \text{(balance)} \\ & x_{f,h} \geq 0 \quad f = 1, \dots, \ell; \quad h = 1, \dots, m && \text{(nonnegativity)} \\ & y_{f,h,r} \geq 0 \quad f = 1, \dots, \ell; \quad h = 1, \dots, m; \quad r = 1, \dots, n \end{aligned}$$

Besides the objective function and capacity constraints derived earlier, model (2.10) includes four new systems of constraints. The first of these enforces demand for each hybrid in each sales region. Amounts shipped from various facilities are summed to compute the total applicable to each demand. A second system of constraints balances production and distribution. It requires that the total of any hybrid shipped from any facility should match the amount of that hybrid produced there. Finally, there are the variable-type constraints. The last two systems require all production and distribution quantities to be nonnegative.

How Models Become Large

Assuming $\ell = 20$, $m = 25$ and $n = 30$, the 500 x variables and 15,000 y variables of model (2.10) are subject to a total of $(20 + 750 + 500 + 15,500) = 16,770$ constraints:

$$\begin{aligned} \ell &= 20 && \text{capacity constraints} \\ mn &= 25(30) = 750 && \text{demand constraints} \\ \ell m &= 20(25) = 500 && \text{balance constraints} \\ \ell m + \ell mn &= 20(25) + 20(25)(30) = 15,500 && \text{nonnegativity constraints} \end{aligned}$$

Still, we have managed to write the model compactly in only a few lines.

In part this compactness derives from the power of indexed notation. But there is another reason. Model (2.10) actually involves only a few simple notions: production cost, shipping cost, capacity, demand, balance, and nonnegativity. What makes it big is repetition of these notions over many combinations of facilities, hybrids, and sales regions.

This is typical of the way that OR models grow.

Principle 2.26 Optimization models become large mainly by a relatively small number of objective function and constraint elements being repeated many times for different periods, locations, products, and so on.

With suitable indexing over such problem dimensions, we can express very large models in just a few lines.

2.4 LINEAR AND NONLINEAR PROGRAMS

Different classes of optimization models have enormously different tractability. This makes recognizing the major categories an important modeling skill. In this section we begin developing that ability by illustrating the fundamental distinction between linear programs and nonlinear programs.

General Mathematical Programming Format

The distinction begins with thinking of mathematical programs in terms of functions of the decision variables.

Principle 2.27 The general form of a **mathematical program** or (single objective) optimization model is

$$\begin{aligned} & \min \text{ or } \max && f(x_1, \dots, x_n) \\ & \text{s.t.} && g_i(x_1, \dots, x_n) \begin{cases} \leq \\ = \\ \geq \end{cases} b_i \quad i = 1, \dots, m \end{aligned}$$

where f, g_1, \dots, g_m are given functions of decision variables x_1, \dots, x_n , and b_1, \dots, b_m are specified constant parameters.

Individual constraints may be of \leq , $=$, or \geq form.

To illustrate, return to the Two Crude petroleum model of Section 2.1:

$$\begin{aligned} \min & && 100x_1 + 75x_2 \\ \text{s.t.} & && 0.3x_1 + 0.4x_2 \geq 2.0 \\ & && 0.4x_1 + 0.2x_2 \geq 1.5 \\ & && 0.2x_1 + 0.3x_2 \geq 0.5 \\ & && x_1 \leq 9 \\ & && x_2 \leq 6 \\ & && x_1, x_2 \geq 0 \end{aligned}$$

Decision variables are x_1 and x_2 , so that $n = 2$ and there are $m = 7$ constraints. In format [2.27](#) the implied functions are

$$\begin{aligned} f(x_1, x_2) &\triangleq 100x_1 + 75x_2 \\ g_1(x_1, x_2) &\triangleq 0.3x_1 + 0.4x_2 \\ g_2(x_1, x_2) &\triangleq 0.4x_1 + 0.2x_2 \\ g_3(x_1, x_2) &\triangleq 0.2x_1 + 0.3x_2 \\ g_4(x_1, x_2) &\triangleq x_1 \\ g_5(x_1, x_2) &\triangleq x_2 \\ g_6(x_1, x_2) &\triangleq x_1 \\ g_7(x_1, x_2) &\triangleq x_2 \end{aligned} \tag{2.11}$$

Notice that there are g functions for both main and variable-type constraints.

Right-Hand Sides

Format [2.27](#) collects everything involving the decision variables in the functions f, g_1, \dots, g_m . Constraint limits b_1, \dots, b_m must be constants.

For obvious reasons these constraint constants b_i are called the **right-hand sides** (or **RHSs**) of the model. In our Two Crude application the right-hand sides are

$$b_1 = 2.0, \quad b_2 = 1.5, \quad b_3 = 0.5, \quad b_4 = 9, \quad b_5 = 6, \quad b_6 = 0, \quad \text{and} \quad b_7 = 0$$

EXAMPLE 2.12: EXPRESSING MODELS IN FUNCTIONAL FORM

Assuming that the decision variables are $w_1, w_2,$ and $w_3,$ express the following optimization model in general functional format [2.27] and identify all required functions and right-hand sides:

$$\begin{aligned} \max \quad & (w_1)^2 + 8w_2 + (w_3)^2 \\ \text{s.t.} \quad & w_1 + 6w_2 \leq 10 + w_2 \\ & (w_2)^2 = 7 \\ & w_1 \geq w_3 \\ & w_1, w_2 \geq 0 \end{aligned}$$

Solution: In format [2.27] the objective function is

$$f(w_1, w_2, w_3) \triangleq (w_1)^2 + 8w_2 + (w_3)^2$$

After collecting all terms involving the decision variables on the left-hand side, constraints have the form

$$\begin{aligned} g_1(w_1, w_2, w_3) &\leq 10 \\ g_2(w_1, w_2, w_3) &= 7 \\ g_3(w_1, w_2, w_3) &\geq 0 \\ g_4(w_1, w_2, w_3) &\geq 0 \\ g_5(w_1, w_2, w_3) &\geq 0 \end{aligned}$$

where

$$\begin{aligned} g_1(w_1, w_2, w_3) &\triangleq w_1 + 6w_2 - w_2 = w_1 + 5w_2 \\ g_2(w_1, w_2, w_3) &\triangleq (w_2)^2 \\ g_3(w_1, w_2, w_3) &\triangleq w_1 - w_3 \\ g_4(w_1, w_2, w_3) &\triangleq w_1 \\ g_5(w_1, w_2, w_3) &\triangleq w_2 \end{aligned}$$

Associated right-hand-side constants are

$$b_1 = 10, \quad b_2 = 7, \quad b_3 = 0, \quad b_4 = 0, \quad \text{and} \quad b_5 = 0$$

Linear Functions

We distinguish classes of mathematical programs according to whether functions f, g_1, \dots, g_m of format [2.27] are linear or nonlinear in the decision variables.

Definition 2.28 | A function is **linear** if it is a constant-weighted sum of decision variables. Otherwise, it is **nonlinear**.

Linear functions may involve only constants and terms with variables in the first power. For example, the objective function

$$f(x_1, x_2) \triangleq 100x_1 + 75x_2$$

of the Two Crude model is linear because it simply applies weights 100 and 75 in summing decision variables x_1 and x_2 . On the other hand, the objective function in Example 2.12,

$$f(w_1, w_2, w_3) \triangleq (w_1)^2 + 8w_2 + (w_3)^2$$

is nonlinear. It includes second powers of some decision variables.

EXAMPLE 2.13: RECOGNIZING LINEAR FUNCTIONS

Assuming that x 's are decision variables and all other symbols are constant, determine whether each of the following functions is linear or nonlinear.

(a) $f(x_1, x_2, x_3) \triangleq 9x_1 - 17x_3$

(b) $f(x_1, x_2, x_3) \triangleq \sum_{j=1}^3 c_j x_j$

(c) $f(x_1, x_2, x_3) \triangleq \frac{5}{x_1} + 3x_2 - 6x_3$

(d) $f(x_1, x_2, x_3) \triangleq x_1 x_2 + (x_2)^3 + \ln(x_3)$

(e) $f(x_1, x_2, x_3) \triangleq e^\alpha x_1 + \ln(\beta) x_3$

(f) $f(x_1, x_2, x_3) \triangleq \frac{x_1 + x_2}{x_2 - x_3}$

Solution:

(a) This function is linear because it merely sums the 3 decision variables with weights 9, 0, and -17 , respectively.

(b) This function is also linear because the c_j are constants.

(c) This function is nonlinear because it involves negative powers of decision variable x_1 .

(d) This function is nonlinear because it involves products, powers not 1, and logarithms of decision variables.

(e) This function is linear. With α and β constant, it is just a weighted sum of the decision variables.

(f) This function is nonlinear because it involves a quotient, even though both numerator and denominator are linear functions.

Linear and Nonlinear Programs Defined

The functional forms in format [2.27](#) characterize linear programs and nonlinear programs.

Definition 2.29 An optimization model in functional form [2.27](#) is a **linear program (LP)** if its (single) objective function f and all constraint functions g_1, \dots, g_m are linear in the decision variables. Also, decision variables should be able to take on whole-number or fractional values.

Definition 2.30 An optimization model in functional form [2.27](#) is a **nonlinear program (NLP)** if its (single) objective function f or any of the constraint functions g_1, \dots, g_m is nonlinear in the decision variables. Also, decision variables should be able to take on whole-number or fractional values.

EXAMPLE 2.14: RECOGNIZING LINEAR AND NONLINEAR PROGRAMS

Assuming that y 's are decision variables and all other symbols are constant, determine whether each of the following mathematical programs is a linear program or a nonlinear program.

- (a)
$$\begin{aligned} \min \quad & \alpha(3y_1 + 11y_4) \\ \text{s.t.} \quad & \sum_{j=1}^5 d_j y_j \leq \beta \\ & y_j \geq 1 \quad j = 1, \dots, 9 \end{aligned}$$
- (b)
$$\begin{aligned} \min \quad & \alpha(3y_1 + 11y_4)^2 \\ \text{s.t.} \quad & \sum_{j=1}^5 d_j y_j \leq \beta \\ & y_j \geq 1 \quad j = 1, \dots, 9 \end{aligned}$$
- (c)
$$\begin{aligned} \max \quad & \sum_{j=1}^9 y_j \\ \text{s.t.} \quad & y_1 y_2 \leq 100 \\ & y_j \geq 1 \quad j = 1, \dots, 9 \end{aligned}$$

Solution: We apply definitions [2.11](#) and [2.12](#).

(a) This model is a linear program because the objective function and all constraints involve just weighted sums of the decision variables.

(b) This model has the same linear constraints as the model of part (a). However, it is a nonlinear program because its objective function is nonlinear.

(c) This model is a nonlinear program. Its objective function and last 9 constraints are linear, but the single nonlinear constraint

$$y_1 y_2 \leq 100$$

renders the entire model nonlinear.

Two Crude and Pi Hybrids Models Are LPs

Both the Two Crude Petroleum model of Section 2.1 and the Pi Hybrids model of Section 2.3 are linear programs. In the Two Crude case we exhibited the required functions in (2.11). Each obviously satisfies definition [2.10]. It is easier to be confused about the much larger Pi Hybrids model (2.10) because we assigned so many symbolic names to constants. However, a careful look will show that its objective function and every one of its constraints involve just a weighted sum of decision variables $x_{f,h}$ and $y_{f,h,r}$. It, too, is a linear program.

APPLICATION 2.3: E-MART

For an example of a nonlinear program or NLP, consider the problem of budgeting advertising expenditures faced by a large European variety store chain we will call E-mart.² E-mart sells products in m major merchandise groups, such as children's wear, candy, music, toys, and electric. Advertising is organized into n campaign formats promoting specific merchandise groups through a particular medium (catalog, press, or television). For example, one variety of campaign advertises children's wear in catalogs, another promotes the same product line in newspapers and magazines, while a third sells toys with television. The profit margin (fraction) for each merchandise group is known, and E-mart wishes to maximize the profit gained from allocating its limited advertising budget across the campaign alternatives.

Indexing, Parameters, and Decision Variables for E-mart

We begin a model by introducing indexes for the two main dimensions of the problem:

$$\begin{aligned} g &\triangleq \text{merchandise group number } (g = 1, \dots, m) \\ c &\triangleq \text{campaign type number } (c = 1, \dots, n) \end{aligned}$$

Then we may denote major input parameters by

$$\begin{aligned} p_g &\triangleq \text{profit, as a fraction of sales, realized from merchandise group } g \\ b &\triangleq \text{available advertising budget} \end{aligned}$$

Decisions to be made concern how to spend E-mart's advertising budget. Thus we will employ decision variables

$$x_c \triangleq \text{amount spent on campaign type } c$$

Nonlinear Response

To complete a model, we must quantify how sales in each group g are affected by advertising expenditures on each campaign c . If the relationship is linear

$$\begin{pmatrix} \text{sales increase} \\ \text{in group } g \text{ due} \\ \text{to campaign } c \end{pmatrix} = s_{g,c} x_c$$

²Based on P. Doyle and J. Saunders (1990), "Multiproduct Advertising Budgeting," *Marketing Science*, 9, 97–113.

where

$s_{g,c} \triangleq$ parameter relating advertising expenditures in campaign c to sales growth in merchandise group g

A linear form is attractive because it leads to easier analysis.

Principle 2.31 | When there is an option, linear constraint and objective functions are preferred to nonlinear ones in optimization models because each nonlinearity of an optimization model usually reduces its tractability as compared to linear forms.

Unfortunately, marketing researchers often find the linear alternative inappropriate. The main difficulty is that linear functions produce what economists call **equal returns to scale**.

Principle 2.32 | Linear functions implicitly assume that each unit increase in a decision variable has the same effect as the preceding increase: equal returns to scale.

The E-mart experience shows something different. Their advertising history exhibits **decreasing returns to scale**; that is, each dollar of advertising in a particular campaign yields less than did the preceding dollar.

Such unequal returns to scale imply nonlinearity. E-mart analysts chose the nonlinear form

$$\left(\begin{array}{l} \text{sales increase} \\ \text{in group } g \text{ due} \\ \text{to campaign } c \end{array} \right) = s_{g,c} \log(x_c + 1) \quad (2.12)$$

This sales response function has the required decreasing returns because logarithms grow at a declining rate as x_c becomes large. Adding $+ 1$ keeps the function nonnegative over $x_c \geq 0$.

E-mart Application Model

After estimating constants $s_{g,c}$ from history, we are ready to state the complete E-mart model:

$$\begin{array}{ll} \max & \sum_{g=1}^m p_g \sum_{c=1}^n s_{g,c} \log(x_c + 1) \quad (\text{total profit}) \\ \text{s.t.} & \sum_{c=1}^n x_c \leq b \quad (\text{budget limit}) \\ & x_c \geq 0 \quad c = 1, \dots, n \quad (\text{nonnegative expenditures}) \end{array} \quad (2.13)$$

The objective function maximizes total profit by summing sales increase expressions (2.12) times corresponding profit factors. A single main constraint enforces the budget limit, and variable-type constraints keep all expenditures nonnegative. The model is a nonlinear program because its objective function is nonlinear.

2.5 DISCRETE OR INTEGER PROGRAMS

Variables in mathematical programs always encode decisions, but there are many types of decisions. In this section we introduce **discrete optimization** models, which include decisions of a logical character qualitatively different from those of linear and nonlinear programs. Discrete optimization models are also called integer (linear or nonlinear) programs, mixed-integer (linear or nonlinear) programs, and combinatorial optimization problems.

APPLICATION 2.4: BETHLEHEM INGOT MOLD

For an example, we turn to the problem confronted by Bethlehem Steel Corporation in choosing ingot sizes and molds.³ In their process for making steel products, molten output from main furnace is poured into large molds to produce rectangular blocks called *ingots*. After the molds have been removed, the ingots are reheated and rolled into product shapes such as I-beams and flat sheets.

In our fictional version, Bethlehem's mills using this process make approximately $n = 130$ different products. The dimensions of ingots directly affect efficiency. For example, ingots of one dimension may be easiest to roll into I-beams, but another dimension produces sheet steel with less waste. Some ingot sizes cannot be used at all in making certain products.

A careful examination of the best mold dimensions for different products yielded $m = 600$ candidate designs. However, it is impractical to use more than a few because of the cost of handling and storage. We wish to select at most $p = 6$ and to minimize the waste associated with using them to produce all n products.

Indexes and Parameters of the Bethlehem Application

Our Bethlehem problem has two major index dimensions:

$$\begin{aligned} i &\triangleq \text{mold design number } (i = 1, \dots, m) \\ j &\triangleq \text{product number } (j = 1, \dots, n) \end{aligned}$$

One set of input parameters are the

$$c_{j,i} \triangleq \text{amount of waste caused by using ingot mold } i \text{ on product } j$$

The other input required for a model is some indication of which products can use which molds. For this purpose we define index sets

$$I_j \triangleq \text{collection of indexes } i \text{ corresponding to molds that} \\ \text{could be used for product } j$$

If $i \in I_j$, mold i is feasible for product j .

Discrete versus Continuous Decision Variables

As usual, actual modeling begins with decision variables. Notice, however, that the decisions to be taken in our Bethlehem application are qualitatively different from those of earlier models. A mold design will either be selected or not. Once selected, it will either be used for a given product, or it will not.

³Based on F. J. Vasko, F. E. Wolf, K. S. Stott, and J. W. Scheirer (1989), "Selecting Optimal Ingot Sizes for Bethlehem Steel," *Interfaces*, 19:1, 68–84.

Such decisions require a new, logical, discrete variable type.

Definition 2.33 A variable is **discrete** if it is limited to a fixed or countable set of values. Often, the choices are only 0 and 1.

Frequently occurring discrete variable types include **binary** or **0–1 variables** limited to values 0 and 1, and nonnegative **integer variables** that may take any nonnegative integer value.

We employ discrete variables mainly to model decisions of an all-or-nothing, either–or character such as those involved in our Bethlehem ingot mold application. In particular, we will use

$$y_i \triangleq \begin{cases} 1 & \text{if ingot mold } i \text{ is selected} \\ 0 & \text{if not} \end{cases}$$

Such y_i are discrete because they are limited to two values. We use value 1 to mean an event occurs, and value 0 when it does not. A value of 0.3 or $\frac{4}{5}$ has no physical meaning.

In a similar manner, we model decisions about which mold to use in making each product with decision variables

$$x_{j,i} \triangleq \begin{cases} 1 & \text{if ingot mold } i \text{ is used for product } j \\ 0 & \text{if not} \end{cases}$$

Again the variables are allowed only a countable set of values to reflect the logical character of the decisions.

Contrast these decision variables with the continuous variables we encountered in earlier models.

Definition 2.34 A variable is **continuous** if it can take on any value in a specified interval.

For example, a nonnegative variable is continuous because it can assume any value in the interval $[0, +\infty)$.

Variables in the Two Crude model of Section 2.1 certainly have this continuous character. They are denominated in thousands of barrels of petroleum, and any nonnegative real value has a physical interpretation. Technically speaking, variables of the Pi Hybrids model in Section 2.3 should be limited to the integers (a discrete set) because they count bags of corn seed produced and shipped. Still, the numbers involved are likely to be so large that there is no loss of validity in allowing all nonnegative values. We will see in later chapters that there is a considerable gain in tractability.

Principle 2.35 When there is an option, such as when optimal variable magnitudes are likely to be large enough that fractions have no practical importance, modeling with continuous variables is preferred to discrete because optimizations over continuous variables are generally more tractable than are ones over discrete variables.

EXAMPLE 2.15: CHOOSING DISCRETE VERSUS CONTINUOUS VARIABLES

Decide whether a discrete or a continuous variable would be best employed to model each of the following quantities.

- (a) The operating temperature of a chemical process
- (b) The warehouse slot assigned a particular product
- (c) Whether a capital project is selected for investment
- (d) The amount of money converted from yen to dollars
- (e) The number of aircraft produced on a defense contract

Solution:

- (a) A temperature can assume any value in a physical range. It is naturally continuous.
- (b) The slot will be selected from a finite, and thus countable, list. It will probably require a discrete variable.
- (c) Assuming that one cannot invest in just part of a project, there are only two possibilities: Take the project or reject it. A 0–1 discrete variable is appropriate.
- (d) Amounts of money can assume any nonnegative value. They should be modeled with continuous variables.
- (e) If the number of aircraft is likely to be large, principle [2.21] argues for a continuous variable even though the number of aircraft is clearly countable. If only a few expensive aircraft are planned, it may be necessary to model discreteness explicitly. Then a nonnegative integer variable is required.

Constraints with Discrete Variables

We are now ready to begin forming the constraints of our Bethlehem ingot mold application. One requirement is that at most p molds be selected:

$$\sum_{i=1}^m y_i \leq p$$

Notice that our convention of making the variable = 1 when something happens, and = 0 otherwise makes such requirements easy to express.

A similar requirement is that each product be assigned exactly one mold design. We may sum over the possible choice to express those constraints as

$$\sum_{i \in I_j} x_{j,i} = 1 \quad j = 1, \dots, n$$

Notice again how easy it is to model “at least 1,” “at most 1,” and “exactly 1” restrictions with 0–1 discrete variables.

A last system of main constraints for the Bethlehem model must encode the dependency between x and y variables. Mold i cannot be assigned to product j unless it is one of the p selected. That is,

$$x_{j,i} \leq y_i \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

EXAMPLE 2.16: EXPRESSING CONSTRAINTS IN 0–1 VARIABLES

In choosing among a collection of 16 investment projects, variables

$$w_j \triangleq \begin{cases} 1 & \text{if project } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Express each of the following constraints in terms of these variables.

- (a) At least one of the first eight projects must be selected.
- (b) At most three of the last eight projects can be selected.
- (c) Either project 4 or project 9 must be selected, but not both.
- (d) Project 11 can be selected only if project 2 is also.

Solution

$$\begin{array}{ll} \text{(a)} \sum_{j=1}^8 w_j \geq 1 & \text{(b)} \sum_{j=9}^{16} w_j \leq 3 \\ \text{(c)} w_4 + w_9 = 1 & \text{(d)} w_{11} \leq w_2 \end{array}$$

Bethlehem Ingot Mold Application Model

We may complete a discrete optimization model of our Bethlehem ingot mold application as

$$\begin{array}{ll} \min & \sum_{j=1}^n \sum_{i \in I_j} c_{j,i} x_{j,i} & \text{(total waste)} \\ \text{s.t.} & \sum_{i=1}^m y_i \leq p & \text{(select at most } p) \\ & \sum_{i \in I_j} x_{j,i} = 1 & j = 1, \dots, n \quad \text{(one each product)} \\ & x_{j,i} \leq y_i & j = 1, \dots, n; i \in I_j \quad \text{(use only if selected)} \\ & y_i = 0 \text{ or } 1 & i = 1, \dots, m \quad \text{(binary variables)} \\ & x_{j,i} = 0 \text{ or } 1; & j = 1, \dots, n; i \in I_j \end{array} \tag{2.14}$$

The objective function of this model merely totals the scrap waste associated with assigning molds to products. In addition to the main constraints formulated above, we have included m variable-type constraints on the y_j and mn such constraints on the $x_{j,i}$. The specification “= 0 or 1” signals that these variables are discrete and may take on only the values 0 and 1.

Integer and Mixed-Integer Programs

A mathematical program is a **discrete optimization model** if it includes any discrete variables at all. Otherwise, it is a **continuous optimization model**.

Discrete models are often called integer programs because we may think of discrete variables as being restricted to integers within an interval. For example,

$$y_j = 0 \text{ or } 1$$

is equivalent to

$$\begin{aligned} 0 &\leq y_j \leq 1 \\ y_j &\text{ integer} \end{aligned}$$

Whenever the allowed values of a variable are countable, they can be aligned with the integers in a similar way.

Definition 2.36 An optimization model is an **integer program (IP)** if any of its decision variables is discrete. If all variables are discrete, the model is a **pure integer program**; otherwise, it is a **mixed-integer program**.

EXAMPLE 2.17: RECOGNIZING INTEGER PROGRAMS

Determine whether an optimization model over each of the following systems of variables is an integer program, and if so, state whether it is pure or mixed.

- (a) $w_j \geq 0, j = 1, \dots, q$
- (b) $w_j = 0 \text{ or } 1, j = 1, \dots, p$
 $w_{p+1} \geq 0$ and integer
- (c) $w_j \geq 0, j = 1, \dots, p$
 $w_{p+1} \geq 0$ and integer

Solution: We apply definition [2.15](#).

- (a) Here all variables are continuous, so the model is continuous, not discrete.
- (b) The first p variables are limited to 0 and 1, and the last to any nonnegative integer. Thus the model is a pure integer program.
- (c) The one integer variable makes this model discrete, and so an IP. It is a mixed-integer program because it also has continuous decision variables.

Integer Linear versus Integer Nonlinear Programs

Bethlehem ingot model (2.14) would fulfill definition [2.11](#) of a linear program except for the binary type of its variables. Thus it is natural to classify it an integer linear program.

Definition 2.37 A discrete or integer programming model is an **integer linear program (ILP)** if its (single) objective function and all main constraints are linear.

The alternative is an integer nonlinear program.

Definition 2.38 | A discrete or integer programming model is an **integer nonlinear program (INLP)** if its (single) objective function or any of its main constraints is nonlinear.

EXAMPLE 2.18: RECOGNIZING ILPS AND INLPS

Assuming that all w_j are decision variables, determine whether each of the following mathematical programs is best described as a LP, a NLP, an (ILP), or an INLP.

$$\begin{aligned} \text{(a) max} \quad & 3w_1 + 14w_2 - w_3 \\ \text{s.t.} \quad & w_1 \leq w_2 \\ & w_1 + w_2 + w_3 = 10 \\ & w_j = 0 \text{ or } 1 \quad j = 1, \dots, 3 \end{aligned}$$

$$\begin{aligned} \text{(b) min} \quad & 3w_1 + 14w_2 - w_3 \\ \text{s.t.} \quad & w_1, w_2 \leq 1 \\ & w_1 + w_2 + w_3 = 10 \\ & w_j \geq 0 \quad j = 1, \dots, 3 \\ & w_1 \text{ integer} \end{aligned}$$

$$\begin{aligned} \text{(c) min} \quad & 3w_1 + 9\frac{\ln(w_2)}{w_3} \\ \text{s.t.} \quad & w_1 \leq w_2 \\ & w_1 + w_2 + w_3 = 10 \\ & w_2, w_3 \geq 1 \\ & w_1 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(d) max} \quad & 19w_1 \\ \text{s.t.} \quad & w_1 \leq w_2 \\ & w_1 + w_2 + w_3 = 10 \\ & w_2, w_3 \geq 1 \\ & w_1 \geq 0 \end{aligned}$$

Solution:

(a) Except for its discrete variable-type constraints, this model would be a linear program because the objective function and both main constraints are linear. Thus the model is an ILP.

(b) The product in its first main constraint makes this model nonlinear. However, it would not usually be called a nonlinear program because w_1 is discrete. The model is best classified as an INLP.

- (c) The logarithm and quotient terms in its objective function make this model nonlinear. Since all variables are continuous, it should be classified as a NLP.
- (d) The objective function and all main constraints of this model are linear and variable types are all continuous. Thus the model is a LP.

APPLICATION 2.5: PURDUE FINAL EXAM SCHEDULING

We may illustrate integer nonlinear programming applications with a problem familiar to every college student: final exam scheduling. In a typical term Purdue University⁴ picks one of $n = 30$ final exam time periods for each of over $m = 2000$ class units on its main campus. Most exams involve just one class section, but there are a substantial number of “unit exams” held at a single time for multiple sections.

The main issue in this exam scheduling is “conflicts,” instances where a student has more than one exam scheduled during the same time period. Conflicts burden both students and instructors because a makeup exam will be required in at least one of the conflicting courses. Purdue’s exam scheduling procedure begins by processing enrollment records to determine how many students are jointly enrolled in each pair of course units. Then an optimization scheme seeks to minimize total conflicts as it selects time periods for all class units.

Indexing, Parameters, and Decision Variables for Purdue Finals Application

As usual, we begin a model of this problem by introducing indexes for its main dimensions:

$$\begin{aligned} i &\triangleq \text{class unit number } (i = 1, \dots, m) \\ t &\triangleq \text{exam time period number } (t = 1, \dots, n) \end{aligned}$$

Then discrete decision variables encode the schedule options:

$$x_{i,t} \triangleq \begin{cases} 1 & \text{if class } i \text{ is assigned to time period } t \\ 0 & \text{otherwise} \end{cases}$$

Also, we define joint enrollment input parameters:

$$e_{i,i'} \triangleq \text{number of students taking an exam in both class unit } i \text{ and class unit } i'$$

Nonlinear Objective Function

The main challenge in solving Purdue’s final exam scheduling problem with the foregoing notation is to represent total conflicts in an objective function. To begin, focus on any pair of courses i and i' . The product

$$x_{i,t}x_{i',t} = \begin{cases} 1 & \text{if } i \text{ and } i' \text{ are both scheduled at time period } t \\ 0 & \text{if not} \end{cases}$$

⁴Based on C. J. Horan and W. D. Coates (1990), “Using More Than ESP to Schedule Final Exams: Purdue’s Examination Scheduling Procedure II (ESP II),” *College and University Computer Users Conference Proceedings*, 35, 133–142.

Thus we may sum over time periods and multiply by joint enrollment to express the conflict for any pair:

$$\text{conflicts between } i \text{ and } i' = e_{i,i'} \sum_{t=1}^n x_{i,t} x_{i',t}$$

It remains only to total such expressions over all course pairs i, i' . The result is the Purdue final exam scheduling objective function:

$$\min \sum_{i=1}^{m-1} \sum_{i'=i+1}^m e_{i,i'} \sum_{t=1}^n x_{i,t} x_{i',t} \quad (\text{total conflicts}) \quad (2.15)$$

Notice that summations have been indexed so that $i' > i$, to avoid counting any pair more than once. The first sum considers all i except the last, which has no higher index, and the second adds in all pairs with $i' > i$.

Purdue Final Exam Scheduling Application Model

Beginning from objective function (2.15), we may model Purdue's problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m-1} \sum_{i'=i+1}^m e_{i,i'} \sum_{t=1}^n x_{i,t} x_{i',t} && (\text{total conflicts}) \\ \text{s.t.} \quad & \sum_{t=1}^n x_{i,t} = 1 && i = 1, \dots, m \quad (\text{class } i \text{ scheduled}) \\ & x_{i,t} = 0 \text{ or } 1 && i = 1, \dots, m; \quad t = 1, \dots, n \end{aligned} \quad (2.16)$$

Main constraints simply assure that each class unit i is assigned exactly one exam time period t .

Model (2.16) is an INLP. Its main constraints are linear, but the objective function has nonlinear product terms and variable types discrete.

2.6 MULTIOBJECTIVE OPTIMIZATION MODELS

All the models considered so far have a clear, quantitative way to compare feasible solutions. That is, they have single objective functions. In many business and industrial applications, single objectives realistically model the true decision process. Such organizations often are really satisfied to maximize some measure of profit or minimize some approximation to cost, although other objectives may also be relevant.

Matters become much more confused when the problem arises in the government sector, or in a complex engineering design, or in circumstances where uncertainty cannot be ignored. In such applications, solutions may be evaluated quite differently by different participants in the decision process, or against different performance criteria. None can be discounted. A **multiobjective optimization** model is required to capture all the perspectives—one that maximizes or minimizes more than one objective function at the same time.

APPLICATION 2.6: DUPAGE LAND USE PLANNING

Perhaps no public-sector problem involves more conflict between different interests and perspectives than land use planning. That is why a multiobjective approach was adopted when government officials in DuPage County, Illinois, which is a rapidly growing suburban area near Chicago, sought to construct a plan controlling use of its undeveloped land.⁵

Table 2.1 shows a simplified classification with $m = 7$ land use types. The problem was to decide how to allocate among these uses the undeveloped land in the county's $n = 147$ planning regions.

TABLE 2.1 Land Use Types
in DuPage Application

i	Land Use Type
1	Single-family residential
2	Multiple-family residential
3	Commercial
4	Offices
5	Manufacturing
6	Schools and other institutions
7	Open space

Multiple Objectives

No single criterion fully captures the appropriateness of assigning undeveloped acres to a given use. Our version of the problem will use 5:

1. **Compatibility:** an index of the compatibility between each possible use in a region and the existing uses in and around the region.
2. **Transportation:** the time incurred in making trips generated by the land use to/from major transit and auto links.
3. **Tax load:** the ratio of added annual operating cost for government services associated with the use versus increase in the property tax assessment base.
4. **Environmental impact:** the relative degradation of the environment resulting from the land use.
5. **Facilities:** the capital costs of schools and other community facilities to support the land use.

A good plan should make the first of these objectives large and the others small. Assigning the indexes

$i \triangleq$ land use type ($i = 1, \dots, m$)

$j \triangleq$ planning region ($j = 1, \dots, n$)

⁵Based on D. Bammi and D. Bammi (1979), "Development of a Comprehensive Land Use Plan by Means of a Multiple Objective Mathematical Programming Model," *Interfaces*, 9:2, part 2, 50–63.

the following symbolic constants parameterize the five objectives:

$c_{i,j} \triangleq$ compatibility index per acre of land use i in planning region j

$t_{i,j} \triangleq$ transportation trip time generated per acre of land use i in planning region j

$r_{i,j} \triangleq$ property tax load ratio per acre of land use i in planning region j

$e_{i,j} \triangleq$ relative environmental degradation per acre of land use i in planning region j

$f_{i,j} \triangleq$ capital costs for community facilities per acre of land use i in planning region j

Then with nonnegative decision variables

$x_{i,j} \triangleq$ number of undeveloped acres assigned to land use i in planning region j

we have the following multiple objectives:

$$\begin{aligned} \max \quad & \sum_{i=1}^m \sum_{j=1}^n c_{i,j} x_{i,j} \\ \min \quad & \sum_{i=1}^m \sum_{j=1}^n t_{i,j} x_{i,j} \\ \min \quad & \sum_{i=1}^m \sum_{j=1}^n r_{i,j} x_{i,j} \\ \min \quad & \sum_{i=1}^m \sum_{j=1}^n e_{i,j} x_{i,j} \\ \min \quad & \sum_{i=1}^m \sum_{j=1}^n f_{i,j} x_{i,j} \end{aligned}$$

Constraints of the DuPage Land Use Application

Some of the main constraints enforced in the DuPage application are straightforward. Define symbolic parameters

$b_j \triangleq$ number of undeveloped acres in planning region j

$\ell_i \triangleq$ county-wide minimum number of acres allocated to land use type i

$u_i \triangleq$ county-wide maximum number of acres allocated to land use type i

$o_j \triangleq$ number of acres in planning region j consisting of undevelopable floodplains, rocky areas, etc.

Then a first system of main constraints assures that all undeveloped land in each planning region is allocated:

$$\sum_{i=1}^m x_{i,j} = b_j \quad j = 1, \dots, n$$

Two others enforce county-wide lower and upper limits on various uses:

$$\begin{aligned} \sum_{j=1}^n x_{i,j} &\geq \ell_i & i = 1, \dots, m \\ \sum_{j=1}^n x_{i,j} &\leq u_i & i = 1, \dots, m \end{aligned}$$

Finally, we need to assure all undevelopable land is assigned to parks and other open space:

$$x_{7,j} \geq o_j \quad j = 1, \dots, n$$

The more complex constraints describe how land uses interact. In particular, single- and multiple-family residential development creates a demand for nearby commercial centers and open space, as well as new schools and other institutions. Using the symbolic parameters

$s_i \triangleq$ new acres of land use i implied by allocation of an acre of undeveloped land to single-family residential

$d_i \triangleq$ new acres of land use i implied by allocation of an acre of undeveloped land to multiple-family residential

we obtain

$$x_{i,j} \geq s_i x_{1,j} + d_i x_{2,j} \quad i = 3, 6, 7; \quad j = 1, \dots, n$$

Acres in $i = 3$, commercial, $i = 6$, institutions, and $i = 7$, open space, must meet the implied demands.

DuPage Land Use Application Model

Collecting all the above produces the following multiobjective optimization model of our DuPage application:

$$\begin{aligned}
 \max \quad & \sum_{i=1}^m \sum_{j=1}^n c_{i,j} x_{i,j} && \text{(compatibility)} \\
 \min \quad & \sum_{i=1}^m \sum_{j=1}^n t_{i,j} x_{i,j} && \text{(transportation)} \\
 \min \quad & \sum_{i=1}^m \sum_{j=1}^n r_{i,j} x_{i,j} && \text{(tax load ratio)} \\
 \min \quad & \sum_{i=1}^m \sum_{j=1}^n e_{i,j} x_{i,j} && \text{(environmental)} \\
 \min \quad & \sum_{i=1}^m \sum_{j=1}^n f_{i,j} x_{i,j} && \text{(facilities)} \\
 \text{s.t.} \quad & \sum_{i=1}^m x_{i,j} = b_j && j = 1, \dots, n \quad \text{(all used)} \\
 & \sum_{j=1}^n x_{i,j} \geq l_i && i = 1, \dots, m \quad \text{(use minimums)} \\
 & \sum_{j=1}^n x_{i,j} \leq u_i && i = 1, \dots, m \quad \text{(use maximums)} \\
 & x_{7,j} \geq o_j && j = 1, \dots, n \quad \text{(undevelopable)} \\
 & x_{i,j} \geq s_i x_{1,j} + d_i x_{2,j} && i = 3, 6, 7; \quad j = 1, \dots, n \quad \text{(implied needs)} \\
 & x_{i,j} \geq 0 && i = 1, \dots, m; \quad j = 1, \dots, n \quad \text{(nonnegativity)}
 \end{aligned} \tag{2.17}$$

The only new element is nonnegativity type-constraints on the decision variables.

Conflict among Objectives

Like so many other applications, DuPage’s land use planning could be modeled validly only with multiple objective functions. Still, we will see in later chapters that there is a price in tractability. It is almost certain that the objective functions will conflict about the best allocation of land uses. For example, a solution placing trip-generating manufacturing land uses near traffic arteries will score well on the transportation objective, but it may not be at all compatible with existing land use, and it may severely degrade sensitive environments.

Single objective models are easier to deal with because they avoid such conflicts. With multiple objectives it is not even clear how to define an “optimal” solution.

Principle 2.39 When there is an option, single-objective optimization models are preferred to multiobjective ones because conflicts among objectives usually make multiobjective models less tractable.

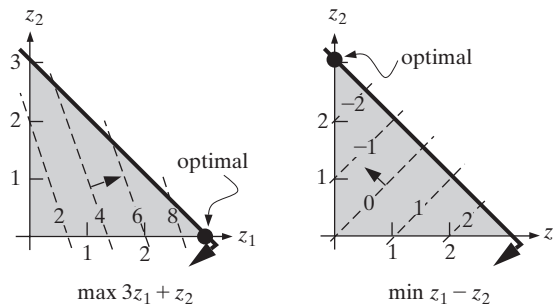
EXAMPLE 2.19: UNDERSTANDING MULTIPLE-OBJECTIVE CONFLICT

Consider the multiobjective mathematical program

$$\begin{aligned} \max \quad & 3z_1 + z_2 \\ \min \quad & z_1 - z_2 \\ \text{s.t.} \quad & z_1 + z_2 \leq 3 \\ & z_1, z_2 \geq 0 \end{aligned}$$

Graph the feasible region and show that the best solutions for the two objective functions conflict.

Solution: The feasible region and contours of the two objective functions are as follows:



Feasible solution $z_1 = 3, z_2 = 0$ is optimal for the first objective function, but $z_1 = 0, z_2 = 3$ is optimal for the second. To find a solution that is overall “best,” the analysis must somehow balance these conflicting objectives.

2.7 CLASSIFICATION SUMMARY

Practitioners of optimization methods are actually confronted, not by generic model forms like those introduced in the various sections of this chapter, but instead by **instances** with specific decision variables, specific constraint functions, and specific objectives, all of which have model parameters explicit as numeric constants rather than symbols. Still, over and over in the rest of this book we will see that the **problem** form—whether a model is linear or nonlinear, continuous or discrete, single or multiobjective—has by far the greatest effect on whether a given instance is tractable and on what methods are most important to try in seeking a solution. Thus, one of the most important skills for students or practitioners struggling with instances of interest is to recognize these distinctions and classify the underlying problem form accordingly.

To clarify the classification, Figure 2.6 provides a summary in a single display. The most tractable LP case has continuous variables, linear constraints, and a single linear objective function. If either constraints or the objective are nonlinear, it becomes a nonlinear NLP. Any discrete variables turn LPs into ILPs and NLPs into INLPs. All forms become more complex if there is more than one objective.

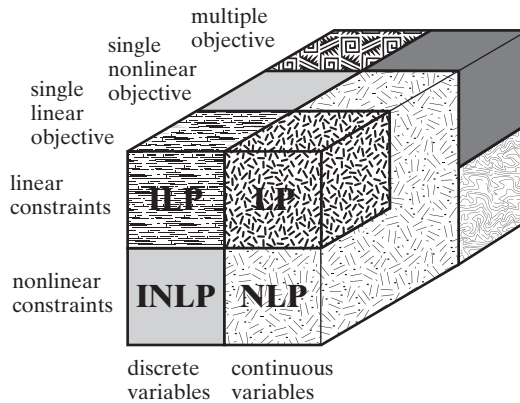


FIGURE 2.6 Classification of Optimization Models

2.8 COMPUTER SOLUTION AND AMPL

This chapter has introduced many of the key issues in formulating and classifying optimization models. Section 2.2 also showed how tiny examples in 2 or 3 decision variables can be solved graphically to gain intuition about issues arising in more realistic cases. Still, real applications of mathematical programming are almost always addressed with computer solution techniques implementing the processes and theory that make up most of the rest of this book.

In this section we introduce some of the main ideas behind the wide variety of available software for conducting computer-based optimization. Special emphasis will be focused on one of the most popular—AMPL.⁶

⁶See R. Fourer, D. Gay and B. Kernighan (2003), *AMPL: A Modeling Language for Mathematical Programming*, 2nd edition, Brooks/Cole, Pacific Grove, California.

Solvers versus Modeling Languages

Designers and users of software for mathematical programming need to address two distinct challenges:

Principle 2.40 | **Solver** software inputs instances of mathematical programs in whatever mathematical format is convenient for the optimization methods being applied, then computes and returns optimal solutions and related analytic results.

Principle 2.41 | **Modeling languages** accept formulations of models and parameters in formats similar to the standard statements of Sections 2.2-2.6, then create a corresponding input set for a chosen solver. Once the solver has completed, results are then translated back to the format of the formulation and reported.

Table 2.3 illustrates for the familiar Two Crude model of (2.6), assuming it is now to be solved with computer software. This is the simplest form, with all indices and parameters hard-coded.

Part (a) of the table shows an AMPL input coding for this elementary case, and part (b) illustrates the output received. Notice:

- AMPL statements always end with a semicolon. Any line or part of a line after “#” is only a comment. Blank lines have no impact.
- var statements define decision variables and declare variable type constraints.

TABLE 2.3 Elementary AMPL Modeling of Two Crude Application 2.1

var	x1 >= 0;	# decision variables and types
var	x2 >= 0;	
minimize tcost:	100*x1+75*x2;	# objective function
subject to		
gas:	0.3*x1+0.4*x2 >= 2.0;	
jet:	0.4*x1+0.2*x2 >= 1.5;	
lubr:	0.2*x1+0.3*x2 >= 0.5;	
saudi:	x1 <= 9;	
venez:	x2 <= 6;	
option	solver cplex;	# choose and call solver
	solve;	
display	cost, x1,x2;	# report solver outputs

(a) AMPL Input

CPLEX	optimal solution	# solver outcome
	462.5	
	tcost = 462.5	# optimal values
	x1 = 2.0	
	x2 = 3.5	

(b) AMPL Output

- minimize or maximize statements specify objective functions.
- Main constraints follow `subject to`, with each labeled by a name at its start.
- Mathematical expressions in both objective functions and constraints use the typical style of scientific computer programming, with “+”, “-”, “*”, “/” operations, and “> =” and “< =” inequalities.
- After the formulation is complete, an option `solver` command chooses the desired solver and a `solve` command invokes it. Here the very popular **CPLEX**⁷ `solver` is employed.
- `display` commands after the `solve` show what outcome variable values should be reported (part (b)).

Indexing, Summations, and Symbolic Parameters

Section 2.2 explained how almost all mathematical programs of realistic size and validity use index sets for various dimensions of the formulation ([2.22]), summations to combine multiple terms, and symbolic parameters to represent constants ([2.24]). Useful modeling languages like AMPL must accommodate such large-scale features in a way that mirrors the standard mathematical formulation.

Table 2.4 illustrates how this can be done for the Pi Hybrids Application 2.2 formulation (2.10). New elements include:

- `param` statements define symbolic parameters.
- `set` statements declare index sets.
- `sum` operators implement summations.
- In all cases where objects are defined over index ranges, the range is shown within “{ }” brackets.
- Subscripts are called out within square “[]” brackets.

Perhaps most importantly, Table 2.4 presents only the symbolic AMPL **model** formulation of Pi Hybrids cases. Constants will follow in an accompanying **data** section.

Principle 2.42 | Large-scale AMPL input sets distinguish between an abstract **model** section, and an accompanying **data** section with details a of specific instance to be solved.

In many settings this distinction proves convenient because different instances of the same model form can be addressed simply by modifying the data section.

Table 2.5 shows how a data section might be structured for a small Pi Hybrids instance with just $l = 2$ facilities, $m = 4$ varieties, and $n = 3$ regions. The full input

⁷See for example IBM ILOG AMPL (2010), *IBM ILOG AMPL Version 12.2 User's Guide*, ampl.com/booklets.

TABLE 2.4 AMPL Model for Pi Hybrids Application 2.2

```

# identify this as the abstract formulation model section
model;

# index sets for facilities, hybrids, and regions
param l;
param m;
param n;
set facils := 1 .. l;
set hybrs := 1 .. m;
set regns := 1 .. n;

# symbolic parameters for facility capacity, buschels per bag,
# production costs, regional demands, and shipping costs
param u{f in facils};
param a{h in vars};
param p{f in facils, h in hybrs};
param d{h in hybrs, r in regns};
param s{f in facils, h in hybrs, r in regns};

# decision variables and types constraints for production and sales
var x{f in facils, h in hybrs} >=0;
var y{f in facils, h in hybrs, r in regns} >= 0;

# total cost objective
minimize tcost: sum{f in facils, h in hybrs} p[f, h]*x[f, h]
+ sum{f in facils, h in hybrs, r in regns} s[f, h, r]*y[f, h, r];

# main constraint sets for facility capacity, regional demands,
# and production-shipping balance
subject to
fcap{f in facils}: sum{h in hybrs} a[h]*x[f, h] <= u[f];
rdems{h in hybrs, r in regns}: sum{f in facils} y[f, h, r] = d[h, r];
psbal{f in facils, h in hybrs}: sum{r in regns} y[f, h, r] = x[f, h];

```

for a given run would be the model section of Table 2.4, the data section of Table 2.5, suitable `solve` commands, and needed `display` choices. Besides rules encountered above, notice that in assigning values to parameters

- The symbol “: = ” is used to indicate assignment of a constant’s value.
- No commas or other delimiters separate data values.
- Scalar parameters are merely assigned with “: = .”
- Single-subscript parameters are assigned in a list of subscript-value pairs.
- Multiple-subscript parameters are assigned in a table with columns corresponding to the last of the subscripts and rows for each combination of the other indices.

TABLE 2.5 AMPL Data Section for a Small Instance of Pi Hybrids Application 2.2

```

# identify this as the data section for a specific instance
data;

# set scalar parameter values
param l := 2;
param m := 4;
param n := 3;

# detail single-subscript parameters in lists
param u := 1 2200 2 2555;
param a := 1 7 2 11 3 6 4 18;

# detail multiple-subscript parameters in tables
param p:  1    2    3    4    :=
1         1.10 0.89 2.05 1.45
2         1.55 1.13 2.15 1.56
3         0.95 0.83 1.80 1.22 ;
param d:  1    2    3    :=
1         123 119 500
2         311 281 333
3         212 188 424
4         189 201 440 ;
param s:  1    2    3    :=
1  1     0.89 0.91 0.77
1  2     1.00 0.84 0.89
1  3     0.77 0.76 0.78
1  4     0.99 1.03 0.85
2  1     0.92 0.89 0.92
2  2     0.87 0.95 0.90
2  3     0.91 0.83 0.77
2  4     0.89 0.79 0.86 ;

```

EXAMPLE 2.20: CODING AMPL FOR LPs WITH INDEXING AND SUMS

Consider the following model and data sections of an AMPL linear program input set.

```

model;
param m;
param n;
set rows := 1 .. m;
set cols := 1 .. n;
param a{i in rows, j in cols};
param r{i in rows};
param d{j in cols};
var x{j in cols} >= 0;

```

```

maximize totl: sum{j in cols} d[j]*x[j];
subject to
lims{ in in rows}: sum{j in cols} a[i,j]*x[j] <= r[i];
data;
param m:= 2;
param n:= 3;
param d:= 1 210 2 333 3 40;
param r:= 1 1100 2 2019;
param a: 1 2 3 :=
1 14 23 41
2 29 19 50 ;

```

Determine the corresponding LP in standard mathematical format with all constant values shown explicitly.

Solution: Using the 3 declared x variables, and given values for symbolic parameters, the formulation is

$$\begin{aligned}
 \max \quad & 210x_1 + 333x_2 + 40x_3 \\
 \text{s.t.} \quad & 14x_1 + 23x_2 + 41x_3 \leq 1100 \\
 & 29x_1 + 19x_2 + 50x_3 \leq 2019 \\
 & x_1, x_2, x_3 \geq 0
 \end{aligned}$$

Nonlinear and Integer Models

Both the models depicted so far are linear programs. Still, AMPL models of nonlinear and integer or mixed-integer programs are also easily accommodated, albeit sometimes with different solvers.

Table 2.6 illustrates the nonlinear case with the model (2.13) of E-mart Application 2.3.

The only new element is use of the `log()` function, which is one of many available in AMPL. Of course, a different solver like MINOS⁸ would be needed for this nonlinear program.

For the integer and mixed-integer programming case, we can illustrate with the AMPL coding of Bethlehem Ingot Mold Application 2.4 shown in Table 2.7.

New elements with the Bethlehem Steel Application of Table 2.7 include the following:

- Most important is indication that both x and y variables are binary, that is, limited to integer values 0 or 1. This simple change in the `var` commands makes the model an integer program.
- When nonnegative integer variables, say z_k for $k \in KK$, are present that need not be just 0 or 1, AMPL coding would include `z{k in KK} integer >=0;`
- This application also illustrates how subsets of index sets can be coded for AMPL. Bethlehem application sets $I_j \subset I$, which are the subsets of mold set I usable for products j , are represented as `amold [j]`.

⁸B. Murtagh and M. Saunders (1998-), *MINOS Optimization Software*, Stanford Business Software, Inc.

TABLE 2.6 AMPL Model for E-Mart Application 2.3

```

model;

# index sets for groups and sales campaigns
param m;
param n;
set groups := 1 .. m;
set cpaigns := 1 .. n;

# symbolic parameters for budget, profit and sales rate increase
param b;
param p{g in groups};
param s{g in groups, c in cpaigns};

# decision variables and types constraints for sales campaigns
var x{c in cpaigns} >=0;

# total profit objective
maximize tprof: sum{g in groups}p[g]
    * sum{c in cpaigns} s[g,c]*log(x[c]+1);

# main budget constraint
subject to
budgt: sum{c in cpaigns} x[c] <= b;

```

TABLE 2.7 AMPL Model for Bethlehem Application 2.4

```

model;

# index sets for molds, products, and applicable molds
param m;
param n;
set molds := 1 .. m;
set prods := 1 .. n;
set amolds {j in prods} within molds;
# symbolic parameters for mold limit and waste
param p;
param c{j in prods, i in amolds[j]};

# decision variables for mold assignments and molds used
var x {j in prods, i in amolds[j]} binary;
var y {i in molds} binary;

# total waste objective
minimize twaste: sum {j in prods} sum {i in amolds [j]} c[j,i]*x[j,i];

# mains constraints for molds selected, assignment for products,
# and mold-assignment correspondence
subject to
mcnt: sum{i in molds} y[i] <= p;
pasmt{j in prods}: sum{i in amolds [j]} x [j ,i) = 1;
mamatch{j in prods,i in amolds[j]}: x[j,i] <= y[i];

```

EXAMPLE 2.21: CODING AMPL FOR MIPs WITH INDEXING AND SUMS

Consider the following model and data sections of an AMPL input set for a mixed-integer linear program.

```

model;
param m;
param n;
set rows := 1 .. m;
set cols := 1 .. n;
param a{i in rows, j in cols};
param r{i in rows};
param d{j in cols};
param f{i in rows};
var x{j in cols} >= 0;
var y{i in rows} integer >= 0;
minimize tcost: sum{j in cols} d[j]*x[j]+sum{i in rows}
f[i]*y[i];
subject to
lims{ i in rows}: sum{j in cols} a[i, j]*x[j] +400*y[i] >= r[i];
data;
param m:= 2;
param n:= 3;
param d:= 1 210 2 333 3 40;
param r:= 1 1100 2 2019;
param f:= 1 300 2 222;
param a: 1 2 3 :=
1 14 23 41
2 29 19 50 ;

```

Determine the corresponding MILP in standard mathematical format with all constant values shown explicitly.

Solution: Using the 3 declared x variables, 2 declared y variables, and given values for symbolic parameters, the formulation is

$$\begin{array}{ll}
 \min & 210x_1 + 333x_2 + 40x_3 + 300y_1 + 222y_2 \\
 \text{s.t.} & 14x_1 + 23x_2 + 41x_3 + 400y_1 \geq 1100 \\
 & 29x_1 + 19x_2 + 50x_3 + 400y_2 \geq 2019 \\
 & x_1, x_2, x_3 \geq 0 \\
 & y_1, y_2 \geq 0 \text{ and integer}
 \end{array}$$

EXERCISES^{9,10}

2-1 The Notip Table Company sells two models of its patented five-leg tables. The basic version uses a wood top, requires 0.6 hour to assemble, and sells for a profit of \$200. The deluxe model takes 1.5 hours to assemble because of its glass top, and sells for a profit of \$350. Over the next week the company has 300 legs, 50 wood tops, 35 glass tops, and 63 hours of assembly available. Notip wishes to determine a maximum profit production plan assuming that everything produced can be sold.

- ✓ (a) Formulate a mathematical programming model with 4 main constraints to select an optimal production plan using decision variables $x_1 \triangleq$ number of basic models produced and $x_2 \triangleq$ number of deluxe models.
- ✓ (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, solve your model graphically for an optimal product mix, and explain why it is unique.
- ✓ (d) On a separate 2-dimensional plot, show that the model has alternative optimal solutions if profits are \$120 and \$300, respectively.

2-2 Wiley Wiz is a mutual fund manager trying to decide how to divide up to \$12 million between domestic and foreign stocks. Domestic stocks have been returning 11% per year and foreign 17%. Naturally, Wiley would like to maximize the annual return from his investments. Still, he wants to exercise some caution. No more that \$10 million of the fund should go into domestic stocks and no more than \$7 million into foreign. Also, at least half as much should be invested in foreign as domestic, and at least half as much in domestic as foreign to maintain some balance.

- (a) Formulate a mathematical programming model with 5 main constraints to decide Wiley's optimal investment plan

using decision variables $x_1 \triangleq$ millions of dollars invested in domestic stocks and $x_2 \triangleq$ millions of dollars invested in foreign stocks.

- ✓ (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, solve your model graphically for an optimal investment plan.
- (d) On a separate 2-dimensional plot, show graphically that the problem has alternative optimal solutions if rates of return are equal for domestic and foreign.

2-3 The Tall Tree lumber company owns 95,000 acres of forestland in the Pacific northwest, at least 50,000 of which must be aerially sprayed for insects this year. Up to 40,000 acres could be handled by planes based at Squawking Eagle, and up to 30,000 acres could be handled from a more distant airstrip at Crooked Creek. Flying time, pilots, and materials together cost \$3 per acre when spraying from Squawking Eagle and \$5 per acre when handled from Crooked Creek. Tall Tree seeks a minimum cost spraying plan.

- (a) Formulate a mathematical programming model to select an optimal spraying plan using decision variables $x_1 \triangleq$ thousands of acres sprayed from Squawking Eagle and $x_2 \triangleq$ thousands of acres sprayed from Crooked Creek.
- ✓ (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, solve your model graphically for an optimal spraying plan, and explain why it is unique.
- (d) On a separate 2-dimensional plot, show graphically that the problem is unbounded if the Squawking Eagle capacity and both nonnegativity constraints are omitted.

⁹Whenever an exercise calls for a **formulation**, be sure to define all decision variables and any symbolic parameters, state all constraints and sets of constraints (both main and variable type), annotating each with its meaning, and state the objective function(s), annotating with its/their meaning(s).

¹⁰Whenever an exercise calls for a **graphic solution** of an optimization model, be sure to label all axes, plot and label each constraint, show their common feasible set (if any), depict contours of the objective function(s) including showing the direction of steepest improvement, then identify and justify an optimal solution, or explain why no optimal solution exists.

- (e) On a separate 2-dimensional plot, show graphically that the problem becomes infeasible if the Crooked Creek facility is destroyed by fire.

2-4 The Fast Food Fantasy (Triple-F) hamburger chain is attempting to respond to customer demand for more healthy food by introducing a new birdburger made from a combination of beef and chicken. The new burger should weight at least 125 grams and have at most 350 calories, 15 grams of fat, and 360 milligrams of sodium. Each gram of beef used has 2.5 calories, 0.2 gram of fat, and 3.5 milligrams of sodium. Corresponding values for chicken are 1.8 calories, 0.1 gram, and 2.5 milligrams. Triple-F wants to find the mix that will meet all requirements and maximize beef content.

- (a) Formulate a mathematical programming model to decide an optimal birdburger blend using decision variables $x_1 \triangleq$ grams of beef per burger and $x_2 \triangleq$ grams of chicken per burger.
- (b) Enter and solve your model with the class optimization software.
- (c) Using a two-dimensional plot, solve your model graphically for an optimal ingredient mix, and explain why it is unique.
- (d) On a separate 2-dimensional plot, show graphically that the model becomes infeasible if weight requirement is raised from 125 grams to 200 grams.
- (e) On a separate 2-dimensional plot, show graphically that the model is unbounded if the minimum weight and nonnegativity constraints are omitted.

2-5 Sun Agriculture (SunAg) operates a farm of 10,000 acres in the dry southwestern part of the United States. In the next season SunAg can plant acres in either vegetables, which return a profit of approximate \$450 per acre, or cotton, which returns \$200 per acre. As a precaution against bad weather, insects, and other factors, SunAg will plant no more than 70% of its total holdings in any one of these options. Also, irrigation water is limited. To grow vegetables requires 10 units (of water) per acre, and cotton requires 7, out of a government allocation of 70,000 units per season. SunAg wishes to develop a planting plan that maximizes profit.

- (a) Formulate a mathematical programming model to solve SunAg's problem using

2 main constraints, 2 upper bound constraints, 2 variable-type constraints, and decision variables v = acres in vegetables and c = acres in cotton.

- (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, find an optimal plan graphically.
- (d) Show graphically that if we accidentally omitted the upper and lower bound constraints on the two variables, the resulting model would be unbounded.
- (e) Now return to the correct model and imagine that SunAg is required by government regulation to plant all its acres. Show graphically that this case is infeasible.

2-6 The Kazak Film company needs to cut 15 long rolls and 10 short rolls of film from stock pieces. Each stock piece can be cut in one of two patterns. The first produces 5 long and 2 short rolls; the second yields 3 long and 5 short. Once any part of a piece of stock is cut, anything that remains is scrap. Also, neither pattern should be used more than 4 times because the jig used to cut it will become too inaccurate. Kazak wants to find the allowable combination of patterns that will minimize the number of stock pieces required.

- (a) Formulate a mathematical model to decide what patterns to use. Use decision variables $x_1 \triangleq$ number of times pattern 1 is used and $x_2 \triangleq$ number of times pattern 2 is used.
- (b) Both variables in your model should be restricted to take on only integer (whole number) values. Explain why.
- (c) Enter and solve your model with the class optimization software.
- (d) Using a 2-dimensional plot, solve your model graphically for an optimal cutting plan. Remember to consider only integer points.
- (e) Explain how your plot shows that the model has alternative optimal solutions.

2-7 A factory is building a 500-square feet open rectangular cooling pool for water exhausted from its main process. The pool will be 8 feet deep, its length should be at least twice its width, and there is room for a width of at most 15 feet. We wish to

choose the feasible design that minimizes cost by minimizing the concrete area of the pool walls.

- ✓ (a) Formulate a mathematical programming model with 3 main constraints to choose an optimal design using decision variables $x_1 \triangleq$ length of the pool and $x_2 \triangleq$ width of the pool.
- ✓ (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, solve your model graphically for an optimal design.
- ✓ (d) On a separate 2-dimensional plot, show graphically that the problem becomes infeasible if the pool can be at most 25 feet in length.

2-8 An architect is designing a cylindrical hotel that will have 150,000 square feet of floor space. She wishes to make the hotel have as many 10-foot-high floors as possible, but the height of the building should not exceed 4 times its diameter or it might be unstable. (Fractional numbers of floors are acceptable in this rough analysis.)

- (a) Formulate a mathematical programming model with 2 main constraints to choose an optimal design using decision variables $x_1 \triangleq$ diameter of the hotel in feet and $x_2 \triangleq$ number of floors.
- ✓ (b) Enter and solve your model with the class optimization software.
- (c) Using a 2-dimensional plot, solve your model graphically for an optimal design.
- (d) On a separate 2-dimensional plot, show graphically that the problem becomes infeasible if the diameter is limited to 50 feet.

2-9 Consider a linear program over constraint set

$$\begin{aligned} x_1 + x_2 &\geq 2 \\ -x_1 + x_2 &\geq 0 \\ x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

- (a) Identify the feasible set in a 2-dimensional plot.
- ✓ (b) Devise a linear objective function for which the LP has a unique optimal solution, and illustrate graphically.
- ✓ (c) Devise a linear objective function for which the LP has alternative optimal solutions, and illustrate graphically.

- ✓ (d) Devise a linear objective function for which the LP is unbounded, and illustrate graphically.
- ✓ (e) Propose a constraint which, if added to the above constraints, would make the problem infeasible, and illustrate graphically.

2-10 Do Exercise 2-9 for a linear program over constraints

$$\begin{aligned} 2x_1 + 3x_2 &\geq 6 \\ x_1 &\geq 0 \end{aligned}$$

2-11 Write each of the following as compactly as possible using summation and “for all” indexed notation.

- ✓ (a) $\min 3y_{3,1} + 3y_{3,2} + 4y_{4,1} + 4y_{4,2}$
- (b) $\max 1y_{1,3} + 2y_{2,3} + 3y_{3,3} + 4y_{4,3}$
- ✓ (c) $\max \alpha_1 y_{1,4} + \alpha_2 y_{2,4} + \dots + \alpha_p y_{p,4}$
- (d) $\min \delta_1 y_1 + \delta_2 y_2 + \dots + \delta_t y_t$
- ✓ (e) $\begin{aligned} y_{1,1} + y_{1,2} + y_{1,3} + y_{1,4} &= s_1 \\ y_{2,1} + y_{2,2} + y_{2,3} + y_{2,4} &= s_2 \\ y_{3,1} + y_{3,2} + y_{3,3} + y_{3,4} &= s_3 \end{aligned}$
- (f) $\begin{aligned} a_{1,1}y_1 + a_{2,1}y_2 + a_{3,1}y_3 + a_{4,1}y_4 &= c_1 \\ a_{1,2}y_1 + a_{2,2}y_2 + a_{3,2}y_3 + a_{4,2}y_4 &= c_2 \\ a_{1,3}y_1 + a_{2,3}y_2 + a_{3,3}y_3 + a_{4,3}y_4 &= c_3 \end{aligned}$

2-12 Suppose that the decision variables of a mathematical programming model are

$x_{i,j,t} \triangleq$ amount of product i produced on manufacturing line j during week t

where $i = 1, \dots, 17; j = 1, \dots, 5; t = 1, \dots, 7$. Use summation and “for all” indexed notation to write expressions for each of the following systems of constraints in terms of these decision variables, and determine how many constraints belong to each system.

- ✓ (a) Total production on any line in any week should not exceed 200.
- ✓ (b) The total 7-week production of product 5 should not exceed 4000.
- ✓ (c) At least 100 units of each product should be produced each week.

2-13 Repeat Exercise 2-12, this time coding the variables and constraints in AMPL [Tables 2.4 and 2.5].

2-14 Suppose that the decision variables of a mathematical programming model are

$$x_{i,j,t} \triangleq \text{acres of land plot } i \text{ allocated to crop } j \text{ in year } t$$

where $i = 1, \dots, 47; j = 1, \dots, 9; t = 1, \dots, 10$. Use summation and “for all” indexed notation to write expressions for each of the following systems of constraints in terms of these decision variables, and determine how many constraints belong to each system.

- (a) The total acres allocated in each year to each plot i cannot exceed the available acres there (call it p_i).
- (b) At least 25% of the total acreage allocated in each of the first 5 years should be devoted to corn (crop $j = 4$).
- (c) More acres should be devoted to beans (crop $j = 1$) in each year and each plot than to any other crop.

□ **2-15** Repeat Exercise 2-14, this time coding the variables and constraints in AMPL as in Table 2.4.

2-16 Assuming that decision variables are y_1, \dots, y_3 , identify the objective function f , constraint functions g_i , and right-hand sides b_i of the general mathematical programming format 2.27 corresponding to each of the following optimization models.

- ✓ (a) $\max (y_1)^2 y_2 / y_3$
 s.t. $y_1 + y_2 + y_3 + 7 = 20$
 $2y_1 \geq y_2 - 9y_3$
 $y_1, y_3 \geq 0$
- (b) $\min 13y_1 + 22y_2 + 10y_2 y_3 + 100$
 s.t. $y_1 + 5 \geq y_2 - 9y_3$
 $8y_2 \geq 4y_3$
 $y_1, y_2 \geq 0, y_3 \leq 0$

2-17 Taking the x_j as variables, and all other symbols as given constants, determine whether each of the following is a linear or a nonlinear constraint, and briefly explain why.

- ✓ (a) $3x_1 + 2x_2 - x_{17} = 9$
- (b) $x_1 + x_3 = 4x_6 + 9x_7$
- ✓ (c) $\alpha/x_9 + 10x_{13} \leq 100$
- (d) $x_4/\alpha + \beta x_{13} \geq 29$

- ✓ (e) $\sum_{j=1}^7 \beta_j (x_j)^2 \leq 10$
- (f) $\log(x_1) \geq 28x_2 x_3$
- ✓ (g) $\max \{x_1, 3x_1 + x_2\} \geq 111$
- (h) $\sum_{j=1}^{15} \sin(\gamma_j) x_j \leq 33$

2-18 Assuming that the w_j are decision variables and all other symbols are constant, determine whether each of the following is a linear program (LP) or a nonlinear program (NLP), and briefly explain why.

- ✓ (a) $\min 3w_1 + 8w_2 - 4w_3$
 s.t. $\sum_{j=1}^3 h_j w_j = 9$
 $0 \leq w_j \leq 10, j = 1, \dots, 3$
- (b) $\min 5w_1 + 23/w_2$
 s.t. $9w_1 - 15w_2 \leq w_3$
 $w_1, w_2 \geq 0$
- ✓ (c) $\max \sum_{j=1}^{10} \alpha_j w_j$
 s.t. $(w_1)^2 + w_2 w_3 \geq 14$
 $w_1, w_2, w_3 \leq 1$
- (d) $\max \sum_{j=1}^{40} w_j / \log(\beta_j)$
 $\sum_{j=1}^{27} \sigma_j w_j \geq e^b$
 $w_j \geq 0 \quad j = 1, \dots, 40$

2-19 Determine whether a discrete or a continuous variable would be most appropriate to model each of the following quantities.

- ✓ (a) Amount of electricity consumed
- (b) Whether a plant should be closed.
- ✓ (c) Process used to manufacture
- (d) Number of storms in the coming hurricane season.

2-20 The Forest Service can build a firewatch tower on any of 8 mountains. Using decision variables $x_j = 1$ if a tower is built on mountain j and $= 0$ otherwise, write constraint(s) enforcing each of the following requirements.

- ✓ (a) In all 3 sites will be selected.
- (b) At least 2 of mountains 1, 2, 4, and 5 must be selected.
- ✓ (c) A tower should not be built on both mountains 3 and 8.
- (d) A tower can be built on mountain 1 only if one is built on mountain 4.

2-21 The National Science Foundation (NSF) has received 4 proposals from professors to undertake new research in OR methods. Each proposal can be accepted for funding next year at the level (in thousands of dollars) shown in the following table or rejected. A total of \$1 million is available for the year.

Proposal	1	2	3	4
Funding	700	400	300	600
Score	85	70	62	93

Scores represent the estimated value of doing each body of research that was assigned by NSF's advisory panel.

- ✓ (a) Formulate a discrete optimization model to decide what projects to accept to maximize total score within the available budget using decision variables $x_j = 1$ if proposal j is selected and $= 0$ otherwise.
- ✓ (b) Enter and solve your model with the class optimization software.

2-22 The state Department of Labor is considering the establishment of area job training centers at up to 4 sites. The following table shows the land cost (in thousands of dollars) of the 4 sites and indicates with an \times the sites that could provide adequate service to each of the five regions of the state.

Region	Site			
	1	2	3	4
Northwest	—	\times	—	\times
Southwest	\times	\times	—	\times
Capital	—	\times	\times	—
Northeast	\times	—	—	\times
Southeast	\times	\times	\times	—
Cost	43	175	60	35

The Department seeks a minimum total cost collection of sites that together could service all five regions.

- (a) Formulate a discrete optimization model to decide what sites to build using decision variables $y_j = 1$ if site j is selected and $= 0$ otherwise.
- ✓ (b) Enter and solve your model with the class optimization software.

2-23 Assuming that the z_j are decision variables, determine whether each of the following mathematical programs is best described as a linear program (LP), a nonlinear program (NLP), an integer linear program (ILP), or an integer nonlinear program (INLP), and briefly explain why.

✓ (a) $\max \quad 3z_1 + 14z_2 + 7z_3$
 s.t. $10z_1 + 5z_2 + 18z_3 \leq 25$
 $z_j = 0$ or $1, \quad j = 1, \dots, 3$

(b) $\max \quad 7z_1 + 12/z_2 + z_2z_3$
 s.t. $15z_1 - 11z_2 \geq z_3$
 $0 \leq z_j \leq 1 \quad j = 1, \dots, 3$

✓ (c) $\min \quad 7z_1z_2 + 17z_2z_3 + 27z_1z_3$
 s.t. $\sum_{j=1}^3 z_j = 2$
 $z_j = 0$ or $1, \quad j = 1, \dots, 3$

(d) $\min \quad z_4$
 s.t. $27z_1 + 33z_2 + 15z_3 \leq z_4$
 $z_1, z_2, z_3 \geq 0$

✓ (e) $\max \quad 12z_1 + 4z_2$
 s.t. $z_1z_2z_3 = 1$
 $z_1, z_2 \geq 0$
 $z_3 = 0$ or 1

(f) $\max \quad 12z_1 + 18z_2 + 15z_3$
 s.t. $z_1 + z_2 + z_3 \leq 14$
 $z_j \geq 0 \quad j = 1, \dots, 3$
 z_1, z_3 integer

✓ (g) $\max \quad (5z_1 + 19z_2)/27$
 s.t. $z_1 + 20z_2 \leq 60$
 $z_1, z_2 \geq 0$

(h) $\max \quad 3z_1z_2 + 15z_3$
 s.t. $z_1 + z_2 \leq 18z_3$
 $z_1, z_2 \geq 0, z_3$ binary

2-24 Return to the mathematical programs of Exercise 2-23. Determine which of each of the following pairs of those models would normally be most tractable, and briefly explain why.

- (a) Model (a) versus (d)
- (b) Model (b) versus (d)
- ✓ (c) Model (c) versus (d)
- (d) Model (e) versus (f)
- (e) Model (a) versus (g)

2-25 Consider the multiobjective optimization model

$$\begin{aligned} \max \quad & x_1 \\ \max \quad & -3x_1 + x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 4 \\ & x_1 \leq 8 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- ✔ (a) Compute graphically a solution that is optimal if only the first objective is considered.
- ✔ (b) Compute graphically a solution that is optimal if only the second objective is considered.
- ✔ (c) Discuss the conflict inherent in trying to maximize both objectives at once.

2-26 Do Exercise 2-25 for the multiobjective optimization model

$$\begin{aligned} \min \quad & x_1 + 10x_2 \\ \min \quad & 12x_1 + 5x_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

2-27 Mexican Communications¹¹ is choosing cable for a new 16,000-meter telephone line. The following table shows the diameters available, along with the associated cost, resistance, and attenuation of each per meter.

Diameter (0.1 mm)	Cost (\$/m)	Resistance (ohms/m)	Attenuation (db/m)
4	0.092	0.279	0.00175
5	0.112	0.160	0.00130
6	0.141	0.120	0.00161
9	0.420	0.065	0.00095
12	0.719	0.039	0.00048

The company wishes to choose the least cost combination of wires that will provide a new line with at most 1600 ohms resistance and 8.5 decibels attenuation.

- (a) Formulate a mathematical programming model with three main constraints to choose an optimal combination of wires using decision variables ($d = 4, 5, 6, 9, 12$)

$x_d \triangleq$ meters of diameter d wire used

Assume that resistance and attenuation grow linearly with the length of the wire used.

- ✔ (b) Enter and solve your model with class optimization software.

2-28 The city of Lancaster’s water distribution system¹² has 3 wells for water supply. There are 10 pumps at these 3 wells. It is estimated that a pumping rate of 10,000 gallons per minute is needed to satisfy the city’s total water demand. There are limits on how much water can be pumped from each well: 3000 gal/min from well 1; 2500 gal/min from well 2; 7000 gal/min from well 3. There are also different costs of operating each pump and limits on the rate of each pump:

Pump	Maximum (gal/min)	Cost (\$/gal/min)	From Well
1	1100	0.05	1
2	1100	0.05	2
3	1100	0.05	3
4	1500	0.07	1
5	1500	0.07	2
6	1500	0.07	3
7	2500	0.13	1
8	2500	0.13	2
9	2500	0.13	3
10	2500	0.13	3

Lancaster wishes to determine the least cost way to meet its pumping needs.

- (a) Explain why appropriate decision variables for a model of this problem are ($j = 1, \dots, 10$)

$x_j \triangleq$ pump rate per minute of pump j

- (b) Assign suitable symbolic names to the constants of the cost and maximum rate values in the table above.
- (c) Formulate an objective function to minimize the cost of the pumping plan selected.
- (d) Formulate a system of 3 constraints enforcing well capacities.

¹¹Based on L. F. Hernandez and B. Khoshnevis (1992), “Optimization of Telephone Wire Gauges for Transmission Standards,” *European Journal of Operational Research*, 58, 389–392.

¹²Based on S. C. Sarin and W. El Benni (1982), “Determination of Optimal Pumping Policy of a Municipal Water Plant,” *Interfaces*, 12:2, 43–48.

- (e) Formulate a system of 10 constraints enforcing pump capacities.
 - (f) Formulate a single constraint enforcing the overall pumping requirement.
 - (g) Complete your model with an appropriate system of variable-type constraints.
 - (h) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (i) Enter and solve your model with class optimization software.

2-29 A small engineering consulting firm¹³ is establishing its plan for the next year. The director and the three partners are to meet to decide which projects to pursue.

Preliminary research has been done on eight projects. The expected profit for each project is given in the following table together with the number of person-days of background preparation each will require and the computer processing unit (CPU) time (in hours) each will use.

Project	Profit	Person-Days	CPU
1	2.1	550	200
2	0.5	400	150
3	3.0	300	400
4	2.0	350	450
5	1.0	450	300
6	1.5	500	150
7	0.6	350	200
8	1.8	200	600

Excluding downtime, it is estimated that 1000 CPU hours will be available through the year. Presently there are 10 engineers (including the director and the partners); each works 240 days per year. At most three engineers could be let go, and management does not want to hire any new engineers for next year, due to market uncertainties. A minimum of 3 projects need to be selected, so each partner will be in charge of at least one project for the year. The director has four favorite projects (3, 4, 5, and 8), and the company needs to select at least one of these.

The firm wishes to formulate an optimization model to determine which projects to undertake,

assuming that projects must be selected on an all-or-nothing basis.

- (a) Justify why appropriate decision variables for the model are $(j = 1, \dots, 8)$.

$$x_j \triangleq \begin{cases} 1 & \text{if project } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

- (b) Assign suitable symbolic names to the constants in the foregoing table.
 - (c) Formulate an objective function to maximize total profit from projects selected.
 - (d) Formulate a pair of constraints to enforce the minimum and maximum engineer person-days available with different numbers laid off.
 - (e) Formulate 3 constraints to enforce the limit on computer time, meet the requirement to select at least three projects, and include at least one of the director's favorites.
 - (f) Complete your model with an appropriate system of variable-type constraints.
 - (g) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (h) Enter and solve your model with class optimization software.

2-30 A major expansion of the Brisbane airport¹⁴ will require moving substantial quantities of earth from 4 sites where it is surplus to 7 locations where it is needed. The following table shows the haul distances (hundreds of meters) between points, as well as the quantity available (m^3) at each surplus site.

Need Site	Surplus Site			
	Apron	Term.	Cargo	Access
Extension	26	28	20	26
Dry pond	12	14	26	10
Roads	10	12	20	4
Parking	18	20	2	16
Fire station	11	13	6	24
Industrial park	8	10	22	14
Perimeter road	20	22	18	21
Quantity available	660	301	271	99

¹³Based on R. B. Gerdding and D. D. Morrison (1980), "Selecting Business Targets in a Competitive Environment," *Interfaces*, 10:4, 34–40.

¹⁴Based on C. Perry and M. Iliff (1983), "From the Shadows: Earthmoving on Construction Projects," *Interfaces*, 13:1, 79–84.

Quantities needed are 247 cubic meters at the extension, 394 at the dry pond, 265 along roads, 105 in the parking area, 90 at the fire station, 85 in the industrial park, and 145 along the perimeter road. The site engineer wishes to compute a minimum total distance times volume plan for accomplishing the required earth moving.

- (a) Explain why appropriate decision variables for a model of this problem are ($i = 1, \dots, 4, j = 1, \dots, 7$)
 $x_{i,j} \triangleq$ cubic meters moved from surplus i to need j
- (b) Assign suitable symbolic names to the constants of the problem.
- (c) Formulate an objective function to minimize total distance times volume movement.
- (d) Formulate a system of 4 main constraints, assuring that the full available amount is moved from each surplus site.
- (e) Formulate a system of 7 main constraints, assuring that the required amount is moved to each needing location.
- (f) Complete your model with an appropriate system of variable-type constraints.
- (g) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (h) Enter and solve your model with class optimization software.

2-31 The state highway department would like to have a formula for estimating the snow removal cost of each snow event as a function of the number of inches of snow to fall. A sample of n falls f_j and corresponding removal costs $c_j, j = 1, \dots, n$, has been collected from history. Now the department would like to fit these data to an S-shaped curve of the form

$$c = \frac{k}{1 + e^{a+bf}}$$

in a way that minimizes the sum of squared errors. Here k, a , and b are empirical parameters of arbitrary sign.

- (a) Explain why the decision variables in this optimization problem are k, a , and b .
- (b) Formulate an unconstrained optimization model to perform the desired curve fit.
- (c) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective?

2-32 The Blue Hills Home Corporation (BHHC)¹⁵ employs 22 remedial education teachers to service special needs students at 22 schools in the St. Louis area. BHHC assigns its teachers to schools for an entire year and is presently making decisions for the current year. One assignment consideration is cost; BHHC reimburses its teachers the cost $c_{i,j}$ of teacher i traveling to school j . However, the assignment must also consider 3 sets of preferences. Teachers express preferences scores $t_{i,j}$ of teacher i being assigned to school j , BHHC supervisors express preferences $s_{i,j}$, and school principals provide scores $p_{i,j}$. In all three cases a higher score indicates a greater preference.

- (a) Explain why appropriate decision variables for a mathematical programming model of this problem are ($i, j = 1, \dots, 22$)

$$x_{i,j} \begin{cases} 1 & \text{if teacher } i \text{ is assigned to school } j \\ 0 & \text{otherwise} \end{cases}$$

- (b) Using these decision variables and the symbolic input constants above, express all BHHC assignment goals as separate objective functions.
- (c) Write a system of 22 constraints expressing the requirement that each teacher be assigned to exactly one school.
- (d) Write a system of 22 constraints expressing the requirement that each school be assigned exactly one teacher.
- (e) Complete your model with an appropriate system of variable-type constraints.
- (f) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.

¹⁵Based on S. Lee and M. J. Schniederjans (1983), "A Multicriteria Assignment Problem: A Goal Programming Approach," *Interfaces*, 13:4, 75–81.

2-33 Professor Proof is trying to decide which of 6 needed teaching assistant tasks he will assign to each of his 2 graduate assistants. Naturally, one assistant would probably be better at some tasks and the other assistant better at others. The following table shows his scoring of their potentials (high is good).

Assistant	Task					
	1	2	3	4	5	6
0	100	85	40	45	70	82
1	80	70	90	85	80	65

Professor Proof wants to assign three tasks to each assistant. However, tasks 5 and 6 are related and should be assigned to the same assistant.

- (a) Explain why appropriate decision variables for an optimization model of this problem are $(j = 1, \dots, 6)$

$$x_j \triangleq \begin{cases} 0 & \text{if task } j \text{ is assigned to 0} \\ 1 & \text{if task } j \text{ is assigned to 1} \end{cases}$$

- (b) Formulate an objective function to maximize the potential of the assignment chosen. (*Hint:* $1 - x_j = 1$ when $x_j = 0$.)
- (c) Formulate a single main constraint to enforce requirements that at each assistant be assigned three tasks.
- (d) Formulate a single main constraint to enforce the requirement that tasks 5 and 6 go to the same assistant.
- (e) Complete your model with an appropriate system of variable-type constraints.
- (f) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.

- (g) Enter and solve your model with class optimization software.

2-34 Fast Food Fantasy (Triple-F) cooks different types of hamburgers $j = 1, \dots, 4$ in batches. A batch of burger j consists of at most u_j units and requires the entire cooking grill for t_j minutes. Assuming that the hourly demand for each burger is a known quantity d_j , Triple-F would like to decide the best batch size for each product. All required batches (and fractions of batches) must fit within the available grill time each hour, and the time required to sell out each burger should be minimized so that none will get too cold waiting to be sold.

- (a) Explain why appropriate decision variables for a mathematical programming model of this problem are $(j = 1, \dots, 4)$

$$x_j \triangleq \text{batch size of burger } j$$

- (b) Formulate a system of 4 objective functions, minimizing the time to sell out batches of each burger assuming that demand is smooth over time.
- (c) Formulate a single constraint assuring that all batches needed to meet demand each hour can be cooked.
- (d) Complete your model with a suitable system of upper-bound and variable-type constraints.
- (e) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.

2-35 The Kitty Railroad is in the process of planning relocations of freight cars among the 5 regions of the country to get ready for the fall harvest. The following table shows the cost of moving a car between each pair of regions, along with the number of cars in each at present and the number needed for harvest shipping.

From	Region				
	1	2	3	4	5
1	—	10	12	17	35
2	10	—	18	8	46
3	12	18	—	9	27
4	17	8	9	—	20
5	35	46	27	20	—
Present	115	385	410	480	610
Need	200	500	800	200	300

We want to choose a reallocation plan to get the required number of cars in each region at minimum total moving cost.

- (a) Briefly justify why appropriate decision variables for this problem are $(i, j = 1, \dots, 5, i \neq j)$,

$$x_{i,j} \triangleq \text{number of cars moved from region } i \text{ to region } j$$

- (b) The numbers of cars $x_{i,j}$ must physically be integer (whole numbers), but it is probably better to model them as continuous. Explain why.

- (c) Assign symbolic names for the constants in the foregoing table.
- (d) Write an objective function minimizing total movement cost.
- (e) Write a system of 5 main constraints, assuring that the net number of cars in each region after the move will meet the need.
- (f) Complete your model with an appropriate system of variable-type constraints.
- (g) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (h) Enter and solve your model with class optimization software.

2-36 A large copper company¹⁶ has 23 plants, each of which can burn 4 different kinds of fuels to produce the energy needed in smelting. Energy requirements at each plant p are known quantities r_p . We also know the energy output e_f of each ton of fuel f burned and the quantity of sulfur pollution s_f released per ton of fuel f burned. Costs vary by location, but estimates $c_{f,p}$ are available of the cost per ton for fuel f at plant p . We want to choose mixes of fuels at plants to fulfill energy needs while minimizing both cost and pollution.

- (a) Briefly justify why appropriate decision variables for this problem are ($f = 1, \dots, 4; p = 1, \dots, 23$)
 $x_{f,p} \triangleq$ amount of fuel f burned at plant p
- (b) Write an objective function minimizing total energy cost.
- (c) Write an objective function minimizing total sulfur pollution.
- (d) Write a system of main constraints requiring that sufficient energy be produced at each plant. Also indicate how many constraints there are in this system.
- (e) Complete your model with an appropriate system of variable-type constraints. Also indicate how many constraints there are in this system.
- (f) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.

2-37 Alabama Cabinet¹⁷ runs a sawmill producing wood panels called “blanks” for cabinetmaking. Some of the wood comes from logs sawed into boards at the company’s mill, and the remainder derives from boards purchased green (undried). Lumber from both sources must be dried in the company’s kilns before being cut into blanks.

Following are two tables, the first of which shows the purchase price per log, yield of green lumber per log, and availability for each of the three log diameters. Each board foot of green lumber sawed from logs gives 0.09 blank. The second table shows the price, yield in blanks, and availability of the two grades of purchased green lumber.

Logs Purchased			
Diameter	\$/Log	Bd-ft	Logs/Week
10	70	100	50
15	200	240	25
20	620	400	10

Lumber Purchases			
Grade	\$/bd-ft	Blanks/bd-ft	Bd-ft/Week
1	1.55	0.10	5000
2	1.30	0.08	Unlimited

We seek a minimum cost plan for producing at least 2350 blanks per week with the current mill’s capacity to saw 1500 logs and dry 26,500 board feet of lumber each week.

- (a) Explain why suitable decision variables for this model are
 $x_d \triangleq$ number of logs of diameter d purchased ($d = 10, 15, 20$)
 $y_g \triangleq$ board feet of green lumber grade g purchased) ($g = 1, 2$)
- (b) Numbers x_d must physically be integer (whole numbers), but it makes sense to model them as continuous. Explain why.
- (c) Formulate an objective function minimizing total material purchase cost. (We assume that other costs are essentially fixed.)

¹⁶Based on R. L. Bulfin and T. T. deMars (1983), “Fuel Allocation in Processing Copper Ore,” *IIE Transactions*, 15, 217–222.

¹⁷Based on H. F. Carino and C. H. LeNoir (1988), “Optimizing Wood Procurement in Cabinet Manufacturing,” *Interfaces*, 18:2, 10–19.

- (d) Formulate a main constraint assuring that the required number of blanks will be produced.
- (e) Formulate 2 main constraints enforcing sawing and drying capacities.
- (f) Formulate 4 upper-bound constraints on decision variables.
- (g) Complete your model with a suitable system of variable-type constraints.
- (h) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (i) Enter and solve your model with class optimization software.

2-38 The Bottles Film Festival draws thousands of people each year to view some of the latest motion pictures and award medals for the best. Planners are now selecting one of time slots $t = 1, \dots, n$ for each of the $j = 1, \dots, m$ films to be shown. From past experience, the festival can estimate numbers $a_{j,j'} \triangleq$ number of guests who would like to watch both film j and file j' . Now they wish to schedule the films so that no more than 4 are shown at any time and the minimum total number of guests are inconvenienced by two movies they would like to see being scheduled at the same hour.

- (a) Explain why appropriate decision variables for a mathematical programming model of this problem are ($j = 1, \dots, m, t = 1, \dots, n$)

$$x_{j,t} \triangleq \begin{cases} 1 & \text{if } j \text{ occurs at } t \\ 0 & \text{otherwise} \end{cases}$$

- (b) Formulate an objective function to minimize the total number of guests inconvenienced by movies scheduled at the same time. (*Hint:* When $x_{j,t}x_{j',t} = 1$, films j and j' are scheduled at the same time t .)
- (c) Formulate a system of m constraints assuring that each film is scheduled at some time.
- (d) Formulate a system of n constraints assuring that no more than 4 movies are schedule at any time.
- (e) Complete your model with an appropriate system of variable-type constraints.

- (f) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (g) Code your model in AMPL (paralleling Tables 2.6 and 2.7).

2-39 To improve tax compliance¹⁸ the Texas Comptroller's staff regularly audits at corporate home offices the records of out-of-state corporations doing business in Texas. Texas is considering the opening of a series of small offices near these corporate locations to reduce the travel costs now associated with such out-of-state audits. The following table shows the fixed cost (in thousands of dollars) of operating such offices at 5 sites i , the number of audits required in each of 5 states j , and the travel cost (in thousands of dollars) per audit performed in each state from a base at any of the proposed office sites.

Tax Site	Fixed Cost	Cost to Audit of Corporate Location:				
		1	2	3	4	5
1	160	0	0.4	0.8	0.4	0.8
2	49	0.7	0	0.8	0.4	0.4
3	246	0.6	0.4	0	0.5	0.4
4	86	0.6	0.4	0.9	0	0.4
5	100	0.9	0.4	0.7	0.4	0
Audits		200	100	300	100	200

We seek a minimum total cost auditing plan.

- (a) Briefly explain why appropriate decision variables for an optimization model of this problem are

$$x_{i,j} \triangleq \text{fraction of audits at } j \text{ done from } i$$

$$y_i \triangleq \begin{cases} 1 & \text{if office } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

- (b) Explain why the y_i must be modeled as discrete.
- (c) Assign suitable symbolic names to the constants in the foregoing table: the fixed cost of office i , the travel cost for audits done at j from i , and the number of audits at j .
- (d) Formulate an objective function minimizing the sum of fixed office operating

¹⁸Based on J. A. Fitzsimmons and L. A. Allen (1983), "A Warehouse Location Model Helps Texas Comptroller Select Out-of-State Tax Offices," *Interfaces*, 13:5, 40-46.

cost plus travel costs to audit sites. (*Hint:* The number of audits done at i from j is $x_{i,j}$ times the total number required at j .)

- (e) Formulate a system of 5 main constraints requiring that 100% of audits at each j be performed.
- (f) Formulate a system of 25 main constraints specifying that no part of the audits at any j can be done from i unless an office is opened at i .
- (g) Complete your model with systems of variable-type constraints for the x and y decision variables.
- (h) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- ✓ (i) Enter and solve your model with class optimization software.
- ☐ (j) Code your model in AMPL (paralleling Table 2.7).

2-40 Channel 999 TV has staff to provide on-scene coverage of up to 4 high school football games this Friday. The following table indicates 3 possibilities are in the town where Channel 999 is located. At least 2 of them must be covered, as well as at least 1 out of town. The table below shows 4 of the games involve a team likely to compete for the state championship, at least 2 of which must be covered. Games must be fully covered or not at all. Within these requirements Channel 999 wants to maximize its total audience across the ratings points shown for possible game choices.

Game Number	1	2	3	4	5	6	7	8
In town?	Y	Y	Y					
State champ?		Y	Y	Y				Y
Ratings points	3.0	3.7	2.6	1.8	1.5	1.3	1.6	2.0

- ✓ (a) Formulate (but do not solve) a mathematical program to compute optimal choice of games to cover. Be sure to define your decision variables and briefly annotate the objective function, and each (main or variable-type) constraint with a few words indicating its meaning.
- ✓ (b) Code your model in AMPL (paralleling Tables 2.6 and 2.7).

- ✓ (c) Is your model best described as an LP, an ILP, an NLP, or an INLP? Why?

2-41 The Speculators Fund is a stock mutual fund investing in categories $j = 1, \dots, n$ of common stocks. At least a fraction ℓ_j and at most fraction u_j of the fund's capital is invested in any category j . The fund maintains estimates, v_j , of the expected annual return in capital gain and dividends for each dollar invested in category j . They also estimate the risk, r_j , per dollar invested in each category j . The goal is to maximize return at minimum risk.

- (a) Explain why appropriate decision variables for a model of this problem are $x_j \triangleq$ fraction of fund capital invested in category j
- (b) Formulate an objective function maximizing expected return per dollar invested.
- (c) Formulate an objective function minimizing risk per dollar invested assuming that risks for different categories are independent of one another.
- (d) Formulate a main constraint assuring that 100% of the fund's capital is invested somewhere.
- (e) Formulate a system of n constraints enforcing lower bounds on the fraction of fund capital invested in each category.
- (f) Formulate a system of n constraints enforcing upper bounds on the fraction of fund capital invested in each category.
- (g) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.

2-42 Engineers¹⁹ are designing the location for modules $i = 1, \dots, m$ from among the $j = 1, \dots, n$ available sites on a computer board. They already know

$$a_{i,i'} \triangleq \begin{cases} 1 & \text{if a wire is required from module } i \text{ to module } i' \\ 0 & \text{otherwise} \end{cases}$$

$$d_{j,j'} \triangleq \text{distance between sites } j \text{ and } j'$$

¹⁹Based on L. Steinberg (1961), "The Backboard Wiring Problem: A Placement Algorithm," *SIAM Review*, 3, 37-50.

Using this information, they wish to choose a combination of locations that will minimize the total wire length required.

- (a) Explain why appropriate decision variables for a model of this problem are $(i = 1, \dots, m; j = 1, \dots, n)$

$$x_{i,t} \triangleq \begin{cases} 1 & \text{if } i \text{ goes to site } j \\ 0 & \text{otherwise} \end{cases}$$

- (b) Explain why the length of any wire required between modules i and i' can be expressed as

$$\sum_{j=1}^n \sum_{j'=1}^n d_{j,j'} x_{i,j} d_{i',j'}$$

- (c) Use the expression of part (b) to formulate an objective function minimizing total wire length.
- (d) Formulate a system of m constraints assuring that each module is assigned a location.
- (e) Formulate a system of n constraints assuring that each location gets at most one module.
- (f) Complete your model with an appropriate system of variable-type constraints.
- (g) Is your model best classified as an LP, an NLP, an ILP, or an INLP, and is it single- or multiobjective? Explain.
- (h) Code your model in AMPL (paralleling Tables 2.6 and 2.7)

☑ **2-43** Consider the following model and data sections of AMPL input for a mixed-integer linear program.

```

model;
param p;
param q;
set prods := 1 .. p;
set procs := 1 .. q;
param a{i in procs, j in prods};
param b{i in procs};
param v{j in prods};
param f{i in procs};
var x{j in prods} >= 0;
var y{i in procs} integer >= 0;
maximize net: sum{j in prods} v[j]*x[j]-sum{i in procs} f[i]*y[i];
subject to
caps{ i in procs }: sum{j in prods} a[i,j]*x[j] <= b[i]*y[i];
dem: sum{j in prods } x [j] >= 205;
cnt: sum{i in procs } Y [i] <= 2;
data;
param p:= 4;
param q:= 3;
param v:= 1 199 2 229 3 188 4 205;
param b:= 1 2877 2 2333 3 3011;
param f:= 1 180 2 224 3 497;
param a: 1 2 3 4 :=
1 0 0 23 41
2 14 29 0 0
3 0 0 11 27 ;

```

Determine the corresponding MILP in standard mathematical format with all constant values shown explicitly.

2-44 Consider the following model and data sections of AMPL input for a linear program.

```

model;
param m;
param n;
param l;
set plants := 1 .. n;
set procs := 1 .. m;
set periods := 1 .. l;
param p{i in procs, j in plants, t in periods};
param b{i in procs, t in periods};
param r{j in plants t in periods};
param d{t in periods};
var x{j in plants, t in periods} >= 0;
maximize retn: sum{j in plants, t in periods} r[j, t]*x[j, t];
subject to
caps{ i in procs, t in periods}: sum{j in plants} p[i,j,t]*x[j, t] <= b[i,t];
demd {t in periods}: sum{j in plants} x[j,t] >= d[t];
data;
param l := 4;
param m := 2;
param n := 3;
param d := 1 200 2 300 3 250 4 500;
param r: 1 2 3 4 :=
1 11 15 19 10
2 19 23 44 67
3 17 18 24 55
param b: 1 2 3 4 :=
1 7600 8200 6015 5000
2 6600 7900 5055 7777 ;
param p: 1 2 3 4 :=
1 1 15 19 23 14
1 2 24 26 18 33
1 3 17 13 16 14
2 1 31 25 39 29
2 2 26 28 22 31
2 3 21 17 20 18 ;

```

Do Exercise 2-43 for this AMPL encoding to show the LP in standard mathematical format with all constant values shown explicitly.

REFERENCES

- Fourer, Robert, David M. Gay, and Brian W. Kernighan (2003), *AMPL; A Modeling Language for Mathematical Programming*, Thomson-Brooks-Cole, Canada.
- Hillier, Fredrick S. and Gerald J. Lieberman (2001), *Introduction to Operations Research*, McGraw-Hill, Boston.
- Taha, Hamdy (2011), *Operations Research - An Introduction*, Prentice-Hall, Upper Saddle River, New Jersey.
- Winston, Wayne L. (2003), *Operations Research - Applications and Algorithms*, Duxbury Press, Belmont California.

Improving Search

To this point we have encountered a variety of types and sizes of deterministic optimization models but succeeded in analyzing only one or two. The time has come to begin looking seriously at solution methods.

Some optimization models admit closed-form solutions or similarly elegant analysis, but the overwhelming majority are approached by **numerical search**—repeatedly trying different values of the decision variables in a systematic way until a satisfactory one emerges. In fact, most optimization procedures can be thought of as variations on a single search theme: improving search.

Improving search tries to better a current solution by checking others nearby. If any proves superior, the search advances to such a solution, and the process repeats. Otherwise, we stop with the current solution. Synonyms for improving search include **local improvement**, **hillclimbing**, **local search**, and **neighborhood search**.

Improving search is one of the major themes of the book, and this chapter provides an elementary introduction. We explore the main strategies underlying improving search algorithms and identify special cases that prove particularly tractable. Familiarity with the model classifications of Chapter 2 is assumed.

Readers are strongly advised to absorb thoroughly the central ideas treated in this chapter before proceeding to the rest of the book. Much later development builds directly on Chapter 3.

3.1 IMPROVING SEARCH, LOCAL, AND GLOBAL OPTIMA

Searching means hunting, and improving searches are hunts. We trek through a sequence of choices for decision variables, trying to find one good enough to justify stopping.

Solutions

Solutions are the points visited in a search.

Definition 3.1 A **solution** is a choice of values for all decision variables.

For example, in the Two Crude model of Section 2.1, which had decision variables

$x_1 \triangleq$ thousands of barrels of Saudi crude refined per day

$x_2 \triangleq$ thousands of barrels of Venezuelan crude refined per day

a solution is a choice of two nonnegative quantities x_1 and x_2 .

Notice that a solution need not be an “answer.” Any pair of nonnegative real numbers forms a solution in the Two Crude case, but we were satisfied in our analysis only by an optimal solution—one that conforms to all constraints and minimizes cost.

Solutions as Vectors

If an optimization model has n decision variables, solutions are n -dimensional. It is convenient to deal with them as **n -vectors**—linear arrays of n **components**. For example, a Two Crude example solution refining 3 thousand barrels of Saudi crude per day and 2 thousand barrels of Venezuelan can be expressed as the vector $\mathbf{x} = (3, 2)$ with components $x_1 = 3$ and $x_2 = 2$.

Since improving search is about moving among such whole solutions, we use vector representations in most of our discussion. Primer 1 reviews vector notations and computations for those who feel rusty. We employ superscripts on solution vectors to indicate the order in which they are explored by a search.

Definition 3.2 For a model with decision vector \mathbf{x} , the first solution visited by a search is denoted $\mathbf{x}^{(0)}$, the next $\mathbf{x}^{(1)}$, and so on.

EXAMPLE 3.1: EXPRESSING SOLUTIONS AS VECTORS

The following table shows the sequence of solutions encountered by an improving search of an optimization model with 4 decision variables.

x_1	x_2	x_3	x_4
1	0	1	2
1	1	-2	4
2	1	-1	4
5	1	-1	6

(a) Express the solutions in standard vector notation [3.2](#).

(b) Identify the value of $x_1^{(3)}$ and $x_3^{(1)}$ in part (a).

PRIMER I: VECTORS

A **scalar** is a single real number such as 2, -0.25 , $\frac{3}{7}$, $\sqrt{17}$, or π . **Scalar variables** such as x , y_6 , Δp , and α are those that take on scalar values. Notice that we always show scalar variables in italic type.

Computations in operations research often involve procedures working on several quantities or variables at the same time. It is convenient to write such operations in terms of **vectors**—one-dimensional arrays of scalars. The vector may be displayed either vertically or horizontally. In this book it makes no difference. Representations

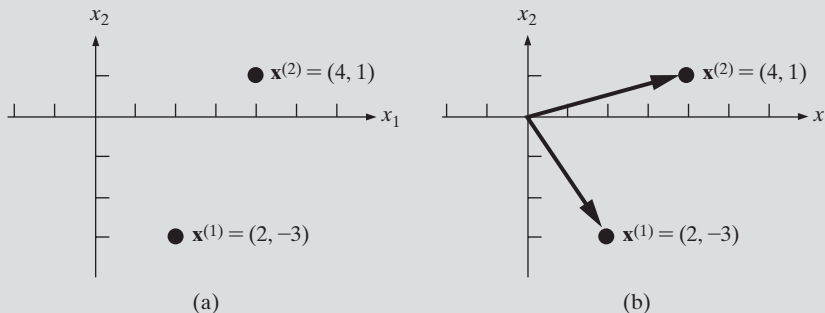
$$\begin{pmatrix} 3 \\ \frac{1}{3} \\ -2 \end{pmatrix} = (3, \frac{1}{3}, -2)$$

are just vertical and horizontal presentations of the same vector.

The number of scalars in a vector is its **dimension**. The example above is a 3-vector because it has dimension 3. We term the scalars making up a vector its **components**. Thus the second component of the example above is $\frac{1}{3}$.

Vector variables represent vectors of scalar quantities. In this book, vector variables are typeset in boldface (e.g., \mathbf{x} , \mathbf{a} , $\Delta \mathbf{p}$), and their components are indicated by subscripts. Thus if \mathbf{p} is a 5-vector (five-dimensional) variable, one possible value would be $\mathbf{p} = (0, -2, \frac{2}{5}, 0, 11)$ with components $p_2 = -2$ and $p_5 = 11$. In this book we distinguish vectors with superscripts in parentheses. That is, $\mathbf{y}^{(7)}$ and $\mathbf{y}^{(13)}$ represent distinct vectors having third components $y_3^{(7)}$ and $y_3^{(13)}$, respectively.

There are two closely related ways to conceptualize vectors geometrically. One scheme simply thinks of an n -vector as a point in n -dimensional space having coordinates equal to its components. For example, 2-vectors $\mathbf{x}^{(1)} = (2, -3)$ and $\mathbf{x}^{(2)} = (4, 1)$ correspond to points $(2, -3)$ and $(4, 1)$ in plot (a) in the following figure. The alternative conceptualization sees vectors as movements in n -space with components indicating displacements in different coordinates. Arrows in part (b) illustrate that this is exactly the same thing if movement is assumed to begin at the origin.



(Continued)

The **length** or **norm** of n -vector \mathbf{x} , denoted $\|\mathbf{x}\|$, is defined accordingly:

$$\|\mathbf{x}\| \triangleq \sqrt{\sum_{j=1}^n (x_j)^2}$$

For example, $\|\mathbf{x}^{(1)}\|$ above is $\sqrt{(2)^2 + (-3)^2} = \sqrt{13}$.

Vectors of the same dimension are added and subtracted component by component. Thus for $\mathbf{x}^{(1)} = (4, 1)$ and $\mathbf{x}^{(2)} = (2, -3)$ above,

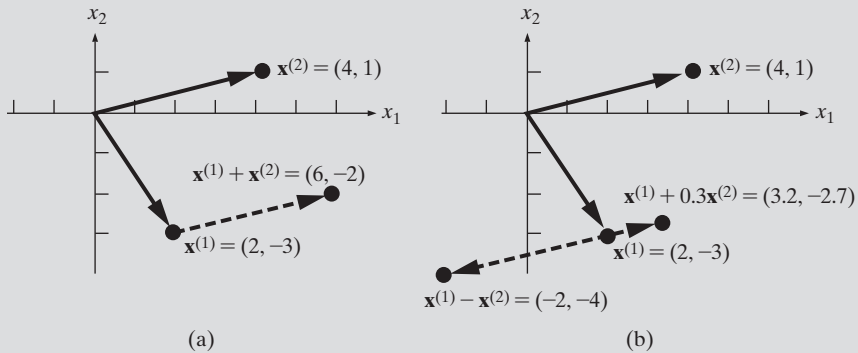
$$\mathbf{x}^{(1)} + \mathbf{x}^{(2)} = \begin{pmatrix} 2 + 4 \\ -3 + 1 \end{pmatrix} = \begin{pmatrix} 6 \\ -2 \end{pmatrix}, \quad \mathbf{x}^{(1)} - \mathbf{x}^{(2)} = \begin{pmatrix} 2 - 4 \\ -3 - 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -4 \end{pmatrix}$$

Similarly, scalar multiples of vectors are formed by simply multiplying each component by the scalar. Using the same $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ gives

$$0.3\mathbf{x}^{(1)} = (0.3(2), 0.3(-3)) = (0.6, -0.9)$$

$$\mathbf{x}^{(1)} + 0.3\mathbf{x}^{(2)} = (2 + 0.3(4), -3 + (0.3)(1)) = (3.2, -2.7)$$

The plots in the following figure demonstrate that these arithmetic operations also have a geometric interpretation. Adding vectors $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ has the effect of concatenating their associated movements in part (a). Similarly, in part (b), subtracting $\mathbf{x}^{(2)}$ from $\mathbf{x}^{(1)}$ extends $\mathbf{x}^{(1)}$'s movement by the negative of $\mathbf{x}^{(2)}$'s, and $\mathbf{x}^{(1)} + 0.3\mathbf{x}^{(2)}$ combines $\mathbf{x}^{(1)}$ with $\frac{3}{10}$ of $\mathbf{x}^{(2)}$.



Vectors of the same dimension can also be multiplied. Although it seems natural to define the product of two vectors as the product of components, a less intuitive definition of multiplication is the one convenient in operations research. The **dot product** of two n -vectors \mathbf{x} and \mathbf{y} is the scalar quantity

$$\mathbf{x} \cdot \mathbf{y} \triangleq \mathbf{y} \cdot \mathbf{x} \triangleq \sum_{j=1}^n x_j y_j$$

(Continued)

For example, the 2-vectors $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ above yield $\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)} = \mathbf{x}^{(2)} \cdot \mathbf{x}^{(1)} = 2(4) + (-3)(1) = 3$. Notice that the dot product of vectors is indicated simply by writing the vectors with a multiplication dot between, and that it makes no difference which vector is mentioned first.

We want multiplication of vectors to mean a dot product because we then have an easy way to denote weighted sums. For example, when we wish to show that the numbers x_1, x_2, \dots, x_6 are summed with the weights w_1, w_2, \dots, w_6 , the result is exactly $\mathbf{w} \cdot \mathbf{x}$, where \mathbf{w} is the vector of w_j and \mathbf{x} the vector of x_j .

Solution:

(a) In the notation of [3.2](#) the four solutions are

$$\mathbf{x}^{(0)} = (1, 0, 1, 2)$$

$$\mathbf{x}^{(1)} = (1, 1, -2, 4)$$

$$\mathbf{x}^{(2)} = (2, 1, -1, 4)$$

$$\mathbf{x}^{(3)} = (5, 1, -1, 6)$$

(b) The first component of solution 3 is $x_1^{(3)} = 5$. The third component of solution 1 is $x_3^{(1)} = -2$.

APPLICATION 3.1: DCLUB LOCATION

To illustrate some search ideas, consider the fictitious problem of choosing a location for the latest DClub discount department store. Dots on the map in Figure 3.1 show the three population centers of the area to be served. Population center 1 has approximately 60,000 persons, center 2 has 20,000, and center 3 has 30,000.

DClub wishes to locate one new store somewhere in the area in a way that maximizes business from the three populations. The obvious decision variables are x_1 and x_2 , the coordinates of the chosen location.

The new store can be located anywhere except in the congested areas within $\frac{1}{2}$ mile of each population center. That is, constraints of the model are

$$[x_1 - (-1)]^2 + (x_2 - 3)^2 \geq \left(\frac{1}{2}\right)^2$$

$$(x_1 - 1)^2 + (x_2 - 3)^2 \geq \left(\frac{1}{2}\right)^2$$

$$(x_1 - 0)^2 + [x_2 - (-4)]^2 \geq \left(\frac{1}{2}\right)^2$$

Figure 3.1 shades the corresponding feasible set.

For an objective function, assume that experience shows that the business attracted from any population follows a “gravity” pattern—proportional to population (here in thousands) and inversely proportional to $1 +$ the square of its distance from

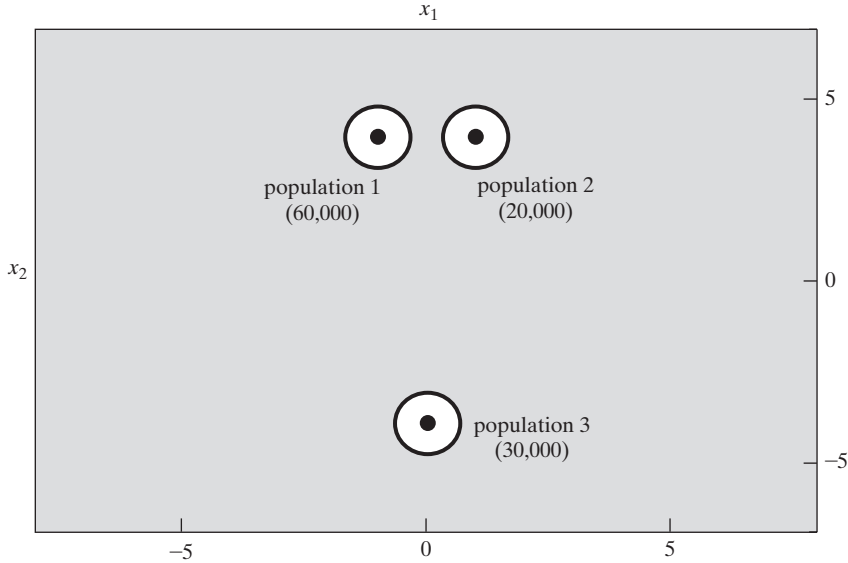


FIGURE 3.1 DClub Location Application

the chosen location. Using this relationship and the coordinates of the three population centers, our objective function becomes

$$\begin{aligned} \max p(x_1, x_2) \triangleq & \frac{60}{1 + (x_1 + 1)^2 + (x_2 - 3)^2} + \frac{20}{1 + (x_1 - 1)^2 + (x_2 - 3)^2} \\ & + \frac{30}{1 + (x_1)^2 + (x_2 + 4)^2} \end{aligned} \quad (3.1)$$

Figure 3.2 provides a 3-dimensional view of the DClub example’s nonlinear objective function. The peak occurs near population center 1. Figure 3.3 gives an easier-to-read contour view (as in Section 2.2) of the full model,

$$\begin{aligned} \max p(x_1, x_2) \triangleq & \frac{60}{1 + (x_1 + 1)^2 + (x_2 - 3)^2} + \frac{20}{1 + (x_1 - 1)^2 + (x_2 - 3)^2} \\ & + \frac{30}{1 + (x_1)^2 + (x_2 + 4)^2} \quad (\text{patronage}) \\ \text{s.t. } & (x_1 + 1)^2 + (x_2 - 3)^2 \geq \frac{1}{4} \quad (\text{avoid 1}) \\ & (x_1 - 1)^2 + (x_2 - 3)^2 \geq \frac{1}{4} \quad (\text{avoid 2}) \\ & (x_1 - 0)^2 + (x_2 + 4)^2 \geq \frac{1}{4} \quad (\text{avoid 3}) \end{aligned} \quad (3.2)$$

As usual, dashed lines connect points of equal objective value.

We want to maximize patronage $p(x_1, x_2)$ subject to avoiding congested circular areas around population centers. The point marked $\mathbf{x}^{(4)}$ in Figure 3.3 is (approximately) optimal because it is the feasible point falling on the highest contour (principle [2.13](#)).

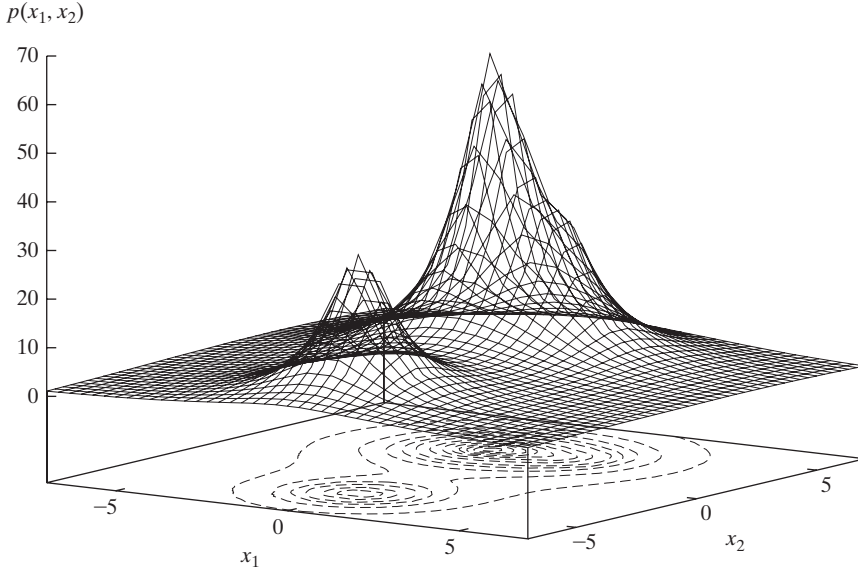


FIGURE 3.2 Three-Dimensional View of the DClub Patronage Function

Example of an Improving Search

Figure 3.3 also traces an improving search leading to optimal DClub solution $\mathbf{x}^{(4)}$. Beginning at

$$\mathbf{x}^{(0)} = (-5, 0) \quad \text{with} \quad p(\mathbf{x}^{(0)}) \approx 3.5$$

it advances through solutions

$$\mathbf{x}^{(1)} = (-3, 4) \quad \text{with} \quad p(\mathbf{x}^{(1)}) \approx 11.5$$

$$\mathbf{x}^{(2)} = (-1, 4.5) \quad \text{with} \quad p(\mathbf{x}^{(2)}) \approx 21.6$$

$$\mathbf{x}^{(3)} = (0, 3.5) \quad \text{with} \quad p(\mathbf{x}^{(3)}) \approx 36.1$$

to optimum

$$\mathbf{x}^{(4)} = (-0.5, 3) \quad \text{with} \quad p(\mathbf{x}^{(4)}) \approx 54.8$$

Do not be concerned at the moment about where the various moves come from. Most of this chapter deals with principles for constructing such a search sequence.

For now, simply notice why we call it an improving search. The process begins with a feasible solution $\mathbf{x}^{(0)}$ and passes exclusively through feasible points. Furthermore, contours show that the objective function value constantly improves along the search path.

Definition 3.3 | **Improving searches** are numerical algorithms that begin at a feasible solution to a given optimization model and advance along a search path of feasible points with ever-improving objective function value.

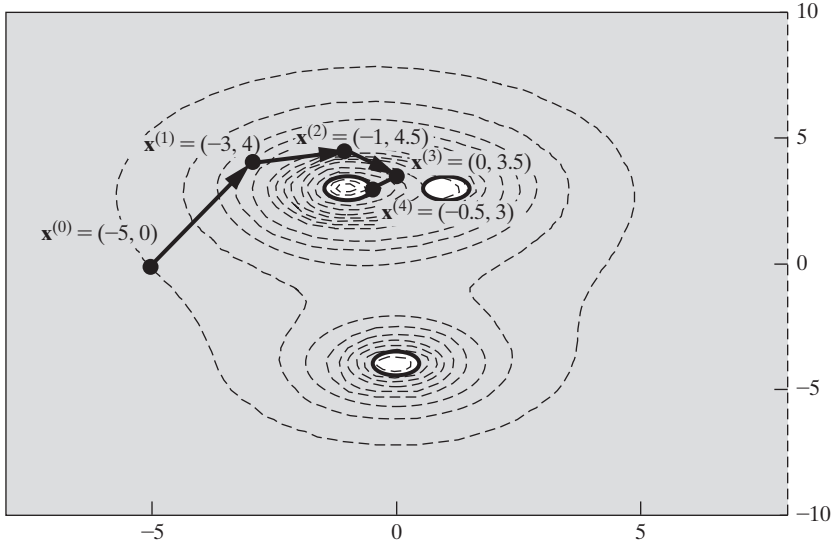


FIGURE 3.3 DClub Example Search Leading to an Optimal Solution

Neighborhood Perspective

For optimization models small enough to graph, as in Figure 3.3, it is easy to spot whether a search path maintains feasibility and constantly improves the objective function. Identifying optimal solutions is not much harder.

Unfortunately, such a global viewpoint is unavailable in typical models which have many decision variables. What we normally have to work with is illustrated by Figure 3.4. That plot zooms in on the region around $\mathbf{x}^{(4)} = (-0.5, 3)$ and blanks out the rest of the graph. We can tell something of the shape of the objective function near $\mathbf{x}^{(4)}$, and we can see the constraint limiting movement to the left. But we know nothing about other parts of the feasible region.

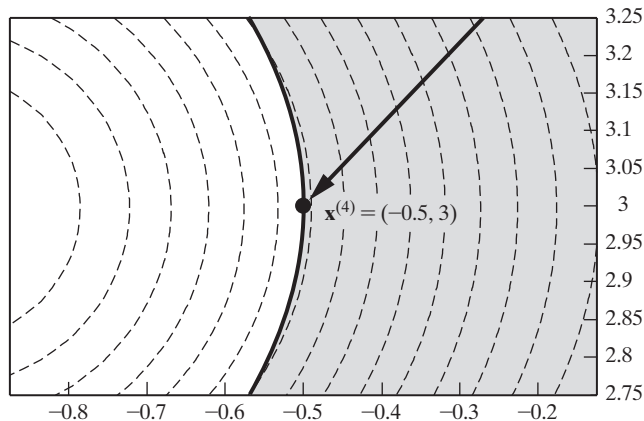


FIGURE 3.4 Neighborhood Perspective in DClub Search

Such a local perspective is typical of real searches. Lacking more complete insight, we must base search choices mostly on information about the neighborhood of the current solution.

Definition 3.4 The **neighborhood** of a current solution $\mathbf{x}^{(t)}$ consists of all nearby points; that is, all points within a small positive distance of $\mathbf{x}^{(t)}$.

Local Optima

If Figure 3.4 were all we knew about our DClub model, we would be unable to tell whether points outside the visible range might prove superior to $\mathbf{x}^{(4)}$. All we could say is that $(-0.5, 3)$ seems the best point in its neighborhood; that is, it is a local optimum.

Definition 3.5 A solution is a **local optimum** (**local maximum** for a maximize problem or **local minimum** for a minimize problem) if it is feasible and if sufficiently small neighborhoods surrounding it contain no points that are both feasible and superior in objective value.

Local Optima and Improving Search

Observe that an improving search which has reached a local optimum can go no further.

Principle 3.6 Improving searches stop if they encounter a local optimum.

Some solutions in the neighborhood of a local optimum may have better objective function value [e.g., $\mathbf{x} = (-0.55, 3)$ in Figure 3.4]. Other neighbors may be feasible [e.g., $\mathbf{x} = (-0.5, 3.05)$]. But no neighboring solution can continue the path of feasible points with ever-improving objective value (definition [3.3](#)) because none is both feasible and superior in objective value.

Local versus Global Optima

Truly optimal solutions to mathematical programs are feasible solutions with as good an objective function value as any other feasible point—neighbor or not. To distinguish this comprehensive notion of optimal, we employ the term global optimum.

Definition 3.7 A solution is a **global optimum** (**global maximum** for a maximize problem or **global minimum** for a minimize problem) if it is feasible and no other feasible solution has superior objective value.

Notice that global optima cannot be improved in any neighborhood.

Principle 3.8 Global optima are always local optima.

No matter how large a radius we consider around $\mathbf{x}^{(4)}$ of Figure 3.3, for example, the corresponding neighborhood contains no better point.

Unfortunately, the converse is not true.

Principle 3.9 Local optima may not be global optima.

Figure 3.5 illustrates with another improving search of the DClub example. This new search conforms to definition [3.3] in starting at a feasible point (in fact, the same one as Figure 3.3) and following an ever-improving path through feasible solutions. Still, it terminates at local maximum $\mathbf{x}^3 = (0, -3.5)$, where the patronage objective function $p(\mathbf{x}^{(3)}) \approx 25.8$, because no neighboring solution is both feasible and superior in the objective (principle [3.6]). We already know from the earlier search of Figure 3.3 that solution $\mathbf{x}^* = (-0.5, 3)$ has a superior objective function value of approximately 54.8. The strictly worst local maximum of Figure 3.5 cannot be globally optimal.

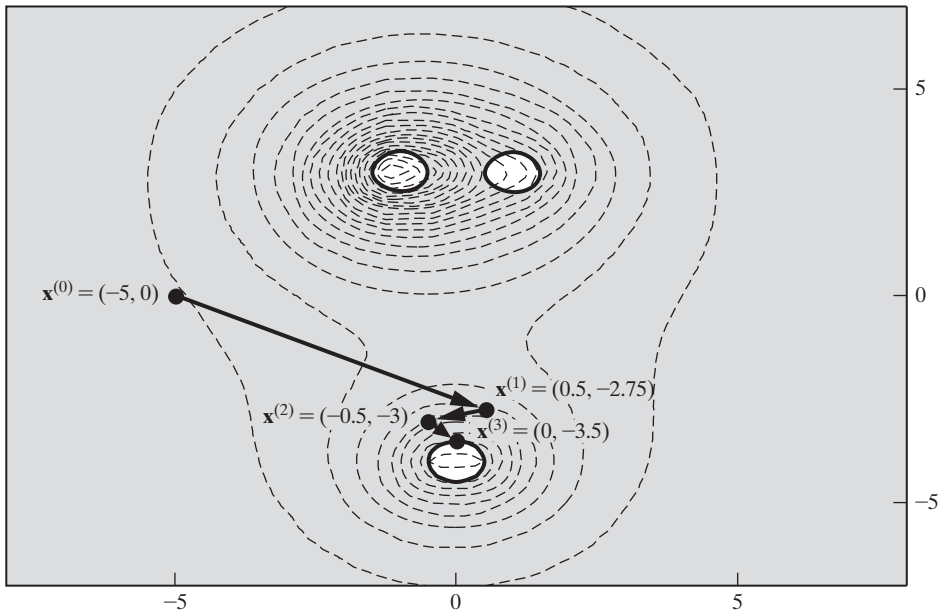


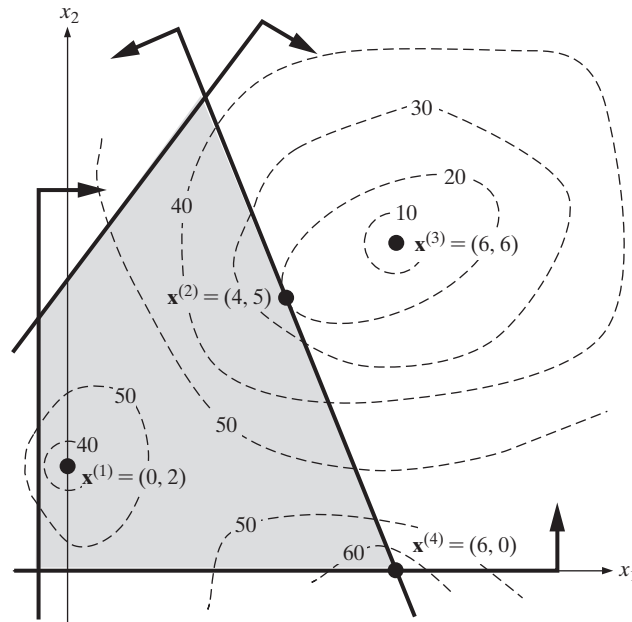
FIGURE 3.5 DClub Search Leading to a Local Maximum Not Global

EXAMPLE 3.2: IDENTIFYING LOCAL AND GLOBAL OPTIMA

The figure that follows depicts constraints and contours of a minimizing optimization model.

Determine whether each of the following points is apparently a global minimum, local minimum, or neither.

- (a) $\mathbf{x}^{(1)} = (0, 2)$
- (b) $\mathbf{x}^{(2)} = (4, 5)$
- (c) $\mathbf{x}^{(3)} = (6, 6)$
- (d) $\mathbf{x}^{(4)} = (6, 0)$



Solution: We apply definitions [3.5](#) and [3.7](#).

(a) Point $\mathbf{x}^{(1)}$ is apparently a local minimum because no neighboring point has better objective value even though all are feasible. Still, point $\mathbf{x}^{(2)}$ has superior objective function value, so $\mathbf{x}^{(1)}$ cannot be globally optimal.

(b) Point $\mathbf{x}^{(2)}$ appears to be a global and thus local minimum (principle [3.8](#)). No other feasible point can match its objective value of 20.

(c) Even though point $\mathbf{x}^{(3)}$ has a very good objective function value, it is neither a local nor a global minimum because it is infeasible.

(d) Point $\mathbf{x}^{(4)}$ is neither a local nor a global minimum because every neighborhood contains feasible points with lower objective function value. One example is $\mathbf{x} = (5.9, 0.1)$.

Dealing with Local Optima

Principle [3.6](#) implies that improving searches can guarantee no more than a local optimum because they stop whenever one is encountered. But our DClub searches show that local optima may not always provide the globally best solutions we prefer (principle [3.9](#)). Must we choose between the computational convenience of improving search and the analytical completeness of global optima?

Fortunately, the answer in many of the most frequently occurring cases is “no.”

Principle 3.10 The most tractable optimization models for improving search are those with mathematical forms assuring every local optimum is a global optimum.

We may pursue an improving search to a local optimum knowing in advance that it will also provide a global optimum.

What can be done with the many models that fail tractability standard [3.10]? Sometimes we can still obtain a global optimum by switching to more complicated forms of search. Often, we must simply settle for less. After trying several improving searches—typically from different starting solutions—we keep the best of the results as an approximate or heuristic optimum.

Principle 3.11 | When models have local optima that are not global, the most satisfactory available analysis is often to run several independent improving searches and accept the best local optimum discovered as a **heuristic** or **approximate optimum**.

3.2 SEARCH WITH IMPROVING AND FEASIBLE DIRECTIONS

Having introduced improving search, we must now make it practical. Just how do we efficiently construct search paths satisfying the always feasible, constantly improving requirements of definition [3.3]? In this section we develop the short list of principles that point the way, and in Section 3.3 we translate them into algebraic conditions. Together, they comprise the foundation for nearly all practical implementations of improving search and much of this book.

Direction-Step Paradigm

Another look at the improving searches of Figures 3.3 and 3.5 will begin to reveal how practical search paths are constructed. Notice that the direction of search does not constantly change. Instead, we pursue a sequence of steps along straight-line move directions. Each begins at one of the numbered solutions $\mathbf{x}^{(t)}$. There, a move direction is chosen along with a step size specifying how far the direction should be pursued. Together they determine new point $\mathbf{x}^{(t+1)}$, and the search continues.

This **direction-step** paradigm lies at the heart of virtually all improving searches.

Definition 3.12 | Improving searches advance from current solution $\mathbf{x}^{(t)}$ to new solution $\mathbf{x}^{(t+1)}$ as

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$$

where vector $\Delta \mathbf{x}$ defines a **move direction** of solution change at $\mathbf{x}^{(t)}$, and **step size** multiplier $\lambda > 0$ determines how far to pursue the direction.

To illustrate, consider the first move of the search in Figure 3.5, which takes us from $\mathbf{x}^{(0)} = (-5, 0)$ to $\mathbf{x}^{(1)} = (0.5, -2.75)$. Most improving search algorithms would accomplish this move by first choosing a vector $\Delta \mathbf{x}$ of relative movement from $\mathbf{x}^{(0)}$ and then applying a suitable step size multiplier λ .

One vector sure to describe the direction chosen is the difference

$$\Delta \mathbf{x} = \mathbf{x}^{(1)} - \mathbf{x}^{(0)} = (0.5, -2.75) - (-5, 0) = (5.5, -2.75)$$

A step size of $\lambda = 1$ then yields the move

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x} = (-5, 0) + 1(5.5, -2.75) = (0.5, -2.75)$$

However, vector $\Delta \mathbf{x}' = (2, -1)$ defines the same direction of movement. Application of step size $\lambda' = 2.75$ produces the identical move

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda' \Delta \mathbf{x}' = (-5, 0) + 2.75(2, -1) = (0.5, -2.75)$$

EXAMPLE 3.3: DETERMINING SOLUTIONS FROM DIRECTIONS AND STEP SIZES

An improving search beginning at solution $\mathbf{w}^{(0)} = (5, 1, -1, 11)$ employs first move direction $\Delta \mathbf{w}^{(1)} = (0, 1, 1, 3)$ for step $\lambda_1 = \frac{1}{3}$, then $\Delta \mathbf{w}^{(2)} = (2, 0, \frac{1}{4}, -1)$ for step $\lambda_2 = 4$, and finally, $\Delta \mathbf{w}^{(3)} = (1, -\frac{1}{3}, 0, 2)$ for step $\lambda_3 = 1$. Determine the solutions visited.

Solution: Applying [3.12](#) gives

$$\begin{aligned}\mathbf{w}^{(1)} &= \mathbf{w}^{(0)} + \lambda_1 \Delta \mathbf{w}^{(1)} = (5, 1, -1, 11) + \frac{1}{3}(0, 1, 1, 3) = (5, \frac{4}{3}, -\frac{2}{3}, 12) \\ \mathbf{w}^{(2)} &= \mathbf{w}^{(1)} + \lambda_2 \Delta \mathbf{w}^{(2)} = (5, \frac{4}{3}, -\frac{2}{3}, 12) + 4(2, 0, \frac{1}{4}, -1) = (13, \frac{4}{3}, \frac{1}{3}, 8) \\ \mathbf{w}^{(3)} &= \mathbf{w}^{(2)} + \lambda_3 \Delta \mathbf{w}^{(3)} = (13, \frac{4}{3}, \frac{1}{3}, 8) + 1(1, -\frac{1}{3}, 0, 2) = (14, 1, \frac{1}{3}, 10)\end{aligned}$$

EXAMPLE 3.4: DETERMINING MOVE DIRECTIONS FROM SOLUTIONS

The first four solutions visited by an improving search are $\mathbf{y}^{(0)} = (5, 11, 0)$, $\mathbf{y}^{(1)} = (4, 9, 3)$, $\mathbf{y}^{(2)} = (4, 9, 7)$, and $\mathbf{y}^{(3)} = (0, 8, 7)$. Determine the move directions employed assuming that all step sizes $\lambda = 1$.

Solution: With all $\lambda = 1$, the sequence of move directions in computation [3.12](#) must merely be the sequence of differences between successive solutions. First

$$\Delta \mathbf{y}^{(1)} = \mathbf{y}^{(1)} - \mathbf{y}^{(0)} = (4, 9, 3) - (5, 11, 0) = (-1, -2, 3)$$

so that

$$\begin{aligned}\mathbf{y}^{(1)} &= \mathbf{y}^{(0)} + \lambda \Delta \mathbf{y} \\ &= (5, 11, 0) + 1(-1, -2, 3) = (4, 9, 3)\end{aligned}$$

Then

$$\begin{aligned}\Delta \mathbf{y}^{(2)} &= \mathbf{y}^{(2)} - \mathbf{y}^{(1)} = (4, 9, 7) - (4, 9, 3) = (0, 0, 4) \\ \Delta \mathbf{y}^{(3)} &= \mathbf{y}^{(3)} - \mathbf{y}^{(2)} = (0, 8, 7) - (4, 9, 7) = (-4, -1, 0)\end{aligned}$$

Improving Directions

We saw in Figure 3.4 that practical improving searches usually take a local perspective, limiting algorithmic decisions to information about the immediate neighborhood of current solution $\mathbf{x}^{(t)}$. Still, improving search definition [3.3] demands that every move improve the objective function value.

How can we be sure of progress when we can “see” only a tiny neighborhood? We limit our choice to immediately improving directions.

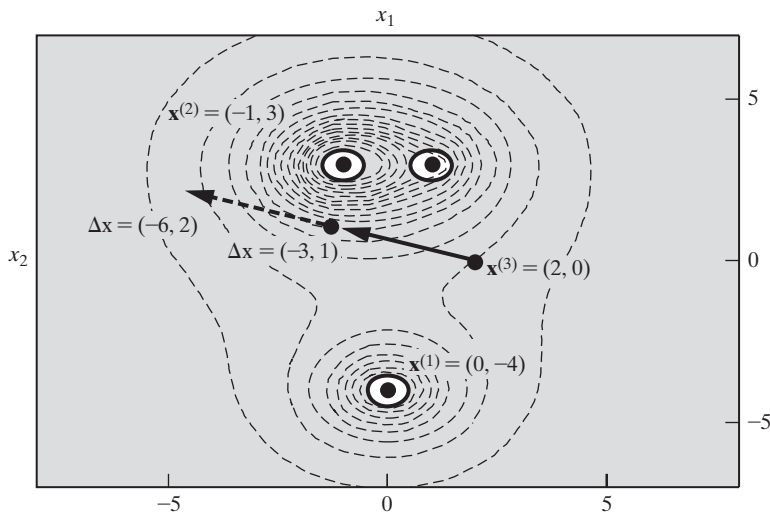
Definition 3.13 Vector $\Delta \mathbf{x}$ is an **improving direction** at current solution $\mathbf{x}^{(t)}$ if the objective function value at $\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$ is superior to that of $\mathbf{x}^{(t)}$ for all $\lambda > 0$ sufficiently small.

If a direction $\Delta \mathbf{x}$ does not immediately improve the objective in the neighborhood of current $\mathbf{x}^{(t)}$, it will not be pursued, regardless of how it affects the objective over larger steps.

Figure 3.6 illustrates for objective function (3.1) of our DClub location example. Constraints have been omitted since they have nothing to do with whether a direction improves.

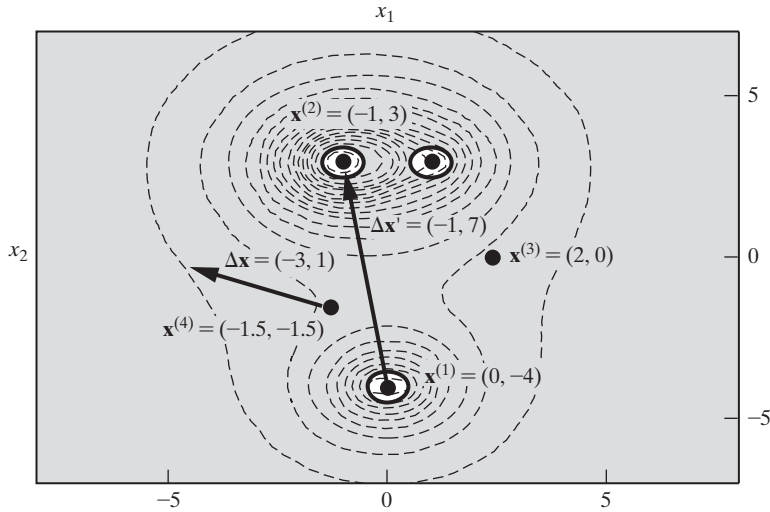
Contours in part (a) show that direction $\Delta \mathbf{x} = (-3, 1)$ improves at $\mathbf{x}^{(3)} = (2, 0)$ because the (maximize) objective function increases. Notice that an improving direction is not required to yield progress forever. Multiple $\Delta \mathbf{x} = (-6, 2)$ (dashed line) remains improving even though a large enough λ produces a point with an objective value worse than that of $\mathbf{x}^{(3)}$.

Part (b) of Figure 3.6 demonstrates that an improving direction at one point need not improve everywhere. The same $\Delta \mathbf{x} = (-3, 1)$ that improved at $\mathbf{x}^{(3)} = (2, 0)$ fails for $\mathbf{x}^{(4)} = (-1.5, -1.5)$.



(a) Improving

FIGURE 3.6 Improving Directions of the DClub Objective

(b) *Nonimproving***FIGURE 3.6** Improving Directions of the DClub Objective (*Continued*)

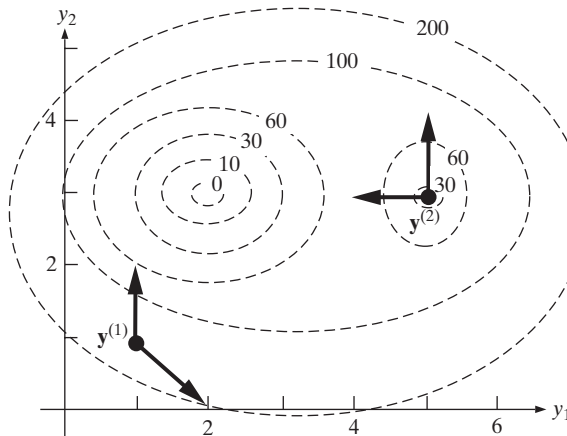
Finally, consider local maximum $\mathbf{x}^{(1)} = (0, -4)$ in Figure 3.6(b). We can see from contours that $\mathbf{x}^{(2)}$ has a better objective function value. Still, indicated direction

$$\Delta \mathbf{x}' = \mathbf{x}^{(2)} - \mathbf{x}^{(1)} = (-1, 3) - (0, -4) = (-1, 7)$$

is not an improving direction at $\mathbf{x}^{(1)}$. It fails definition [3.13](#) because the improvement does not start immediately as we depart $\mathbf{x}^{(1)}$.

EXAMPLE 3.5: RECOGNIZING IMPROVING DIRECTIONS GRAPHICALLY

The following figure plots contours of a minimizing objective function over decision variables y_1 and y_2 .



Determine graphically whether each of the following directions improves at the point indicated.

- (a) $\Delta \mathbf{y} = (1, -1)$ at $\mathbf{y}^{(1)} = (1, 1)$
- (b) $\Delta \mathbf{y} = (0, 1)$ at $\mathbf{y}^{(1)} = (1, 1)$
- (c) $\Delta \mathbf{y} = (0, 1000)$ at $\mathbf{y}^{(1)} = (1, 1)$
- (d) $\Delta \mathbf{y} = (0, 1000)$ at $\mathbf{y}^{(2)} = (5, 3)$
- (e) $\Delta \mathbf{y} = (-1, 0)$ at $\mathbf{y}^{(2)} = (5, 3)$

Solution: We apply definition [3.13](#).

(a) At $\mathbf{y}^{(1)}$ a small movement in the indicated direction $\Delta \mathbf{y} = (1, -1)$ increases (degrades) the objective function value. Thus the direction is not improving.

(b) At the same $\mathbf{y}^{(1)}$ a small movement in the y_2 -coordinate direction $\Delta \mathbf{y} = (0, 1)$ decreases (improves) the objective function value. Thus that direction improves at $\mathbf{y}^{(1)}$.

(c) The length of a move direction has no impact on whether it improves because definition [3.13](#) addresses only sufficiently small steps λ . Thus this case improves for the same reasons as part (b), albeit with smaller λ .

(d) This same direction of part (c) that improved at $\mathbf{y}^{(1)}$ fails to improve at $\mathbf{y}^{(2)}$ because a small step in the y_2 -coordinate direction increases the objective value.

(e) Even though a move in direction $\Delta \mathbf{y} = (-1, 0)$ will eventually decrease the objective function from its value at $\mathbf{y}^{(2)} = (5, 3)$, the progress does not start immediately. Thus this $\Delta \mathbf{y}$ is not an improving direction at $\mathbf{y}^{(2)}$.

Feasible Directions

Moves in improving searches of constrained optimization models must both improve the objective function and maintain feasibility. For the latter, practical implementations parallel the foregoing discussion of improving directions by requiring immediately feasible directions.

Definition 3.14 Vector $\Delta \mathbf{x}$ is a **feasible direction** at current solution $\mathbf{x}^{(t)}$ if point $\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$ violates no model constraint if $\lambda > 0$ is sufficiently small.

Just as with definition [3.13](#), we evaluate directions $\Delta \mathbf{x}$ by considering only the immediate neighborhood of current solution $\mathbf{x}^{(t)}$. If feasibility is maintained for small enough steps, the direction is feasible. Otherwise, $\Delta \mathbf{x}$ will not be considered as our next search direction because we have no way to assess its impact outside the neighborhood.

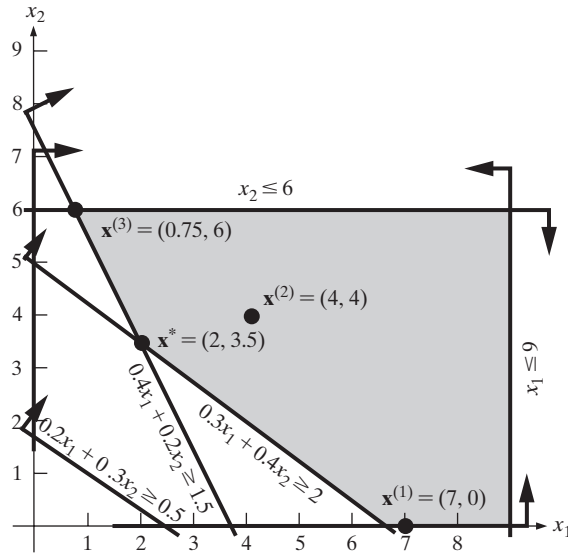


FIGURE 3.7 Constraints of the Two Crude Refinery Example

To illustrate, consider Figure 3.7, which depicts the variable-type and inequality constraints that define the feasible set of the Two Crude refinery model introduced in Sections 2.1 and 2.2:

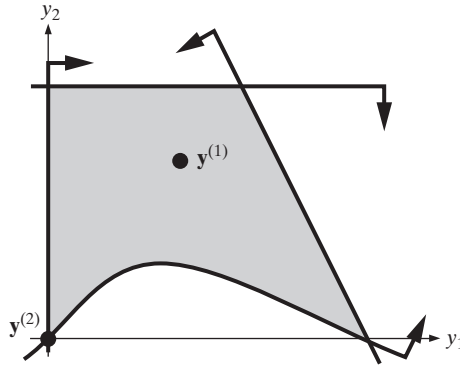
$$\begin{aligned}
 \min \quad & 100x_1 + 75x_2 \\
 \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2 \\
 & 0.4x_1 + 0.2x_2 \geq 1.5 \\
 & 0.2x_1 + 0.3x_2 \geq 0.5 \\
 & 0 \leq x_1 \leq 9 \\
 & 0 \leq x_2 \leq 6
 \end{aligned}$$

Every direction is feasible at solution $\mathbf{x}^{(2)} = (4, 4)$ because a short move in any direction leads to no violation of constraints.

Contrast with $\mathbf{x}^{(1)} = (7, 0)$. There direction $\Delta \mathbf{x} = (0, 1)$ is feasible because a small movement in the x_2 -coordinate direction violates no constraints. As with improving directions, it does not matter that too big a step will produce a violation. The same direction is infeasible at $\mathbf{x}^{(3)} = (0.75, 6)$ because even a tiny $\lambda > 0$ would lead to a violation of the bound constraint $x_2 \leq 6$.

EXAMPLE 3.6: RECOGNIZING FEASIBLE DIRECTIONS GRAPHICALLY

The following figure shows the feasible region of a mathematical program over decision variables y_1 and y_2 .



Determine graphically whether each of the following directions is feasible at the point indicated.

- (a) $\Delta \mathbf{y} = (1, 0)$ at $\mathbf{y}^{(1)}$
- (b) $\Delta \mathbf{y} = (1, 0)$ at $\mathbf{y}^{(2)}$
- (c) $\Delta \mathbf{y} = (0, 1)$ at $\mathbf{y}^{(2)}$
- (d) $\Delta \mathbf{y} = (0, 1000)$ at $\mathbf{y}^{(2)}$

Solution: We apply definition [3.14](#).

(a) We can move from $\mathbf{y}^{(1)}$ in any direction without (immediately) violating constraints. Thus all directions, including $\Delta \mathbf{y} = (1, 0)$, are feasible.

(b) At $\mathbf{y}^{(2)}$ a small step in the y_1 -coordinate direction $\Delta \mathbf{y} = (1, 0)$ takes us outside the feasible region. Thus the direction is not feasible even though a long enough step would restore feasibility.

(c) A small step from $\mathbf{y}^{(2)}$ in the y_2 -coordinate direction $\Delta \mathbf{y} = (0, 1)$ violates no constraint. Thus the direction is feasible.

(d) The length of a move direction has no impact on whether it is feasible because definition [3.14](#) addresses only sufficiently small steps λ . Thus this direction is feasible for the same reason as in part (c), albeit with smaller λ .

Step Size: How Far?

Once an improving feasible move direction has been discovered at the current solution, how far should we follow it? That is, what step size λ should be applied?

We know from definitions [3.13](#) and [3.14](#) that we will improve the objection and retain feasibility for at least small steps. But having an improving feasible direction in hand, it is natural to pursue it as long as it remains so.

Principle 3.15 Improving searches normally apply the maximum step λ for which the selected move direction continues to retain feasibility and improve the objective function.

Notice that principle [3.15](#) involves two issues: how long the direction improves the objective function and how long it remains feasible. We fix λ and choose a new direction when either the objective function stops improving or a constraint is violated.

EXAMPLE 3.7: DETERMINING MAXIMUM STEP SIZE

Suppose that we are searching for an optimal solution to the mathematical program

$$\begin{array}{ll} \min & 10w_1 + 3w_2 \\ \text{s.t.} & w_1 + w_2 \leq 9 \\ & w_1, w_2 \geq 0 \end{array}$$

For current point $\mathbf{w}^{(19)} = (4, 5)$, determine the maximum step in improving feasible direction $\Delta\mathbf{w} = (-3, -8)$ consistent with principle [3.15](#).

Solution: Direction $\Delta\mathbf{w}$ reduces the objective function at every point because it decreases both decision variables, and they both have positive costs. Thus it remains improving for any $\lambda > 0$.

For feasibility, we first consider the main constraint. Any step $\lambda > 0$ will result in

$$(w_1 + \lambda \Delta w_1) + (w_2 + \lambda \Delta w_2) = (4 - 3\lambda) + (5 - 8\lambda) \leq 9$$

That is, the constraint remains satisfied.

We conclude that the maximum step size λ will be determined by feasibility in the nonnegativity constraints $w_1 \geq 0$ and $w_2 \geq 0$. Any $\lambda > \frac{4}{3}$ makes negative the first component of the new solution:

$$\mathbf{w}^{(20)} = \mathbf{w}^{(19)} + \lambda \Delta\mathbf{w} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} + \lambda \begin{pmatrix} -3 \\ -8 \end{pmatrix} = \begin{pmatrix} 4 - 3\lambda \\ 5 - 8\lambda \end{pmatrix}$$

Similarly, any $\lambda > \frac{5}{8}$ will violate nonnegativity on w_2 . Thus the maximum step retaining both improvement and feasibility in the objective function is

$$\lambda = \min \left\{ \frac{4}{3}, \frac{5}{8} \right\} = \frac{5}{8}$$

Search of the DClub Example

Algorithm 3A collects definitions [3.12](#) to [3.14](#) and principle [3.15](#) in a formal improving search procedure. We can illustrate with the DClub search shown in Figure 3.5. As usual, the search begins at a feasible solution vector, here $\mathbf{x}^{(0)} = (-5, 0)$. Its objective value is $p(-5, 0) \approx 3.5$.

Passing to step 1 of Algorithm 3A, we look for an improving feasible direction. The one adopted in Figure 3.5 is $\Delta\mathbf{x}^{(1)} = (2, -1)$. Although there are many other choices, this one clearly does conform to definitions [3.13](#) and [3.14](#) in improving the objective and maintaining feasibility near $\mathbf{x}^{(0)}$.

We must now pick a λ value at step 3. Any movement in the direction chosen leaves us at a feasible point. Thus the step size will be determined by where the objective function quits improving. In models with many variables, some work

is required to find such a maximum λ . Here we can proceed graphically. Contours show stops after a step of approximately $\lambda_1 = 2.75$. Thus step 4's update yields

$$\begin{aligned}\mathbf{x}^{(1)} &\leftarrow \mathbf{x}^{(0)} + \lambda_1 \Delta \mathbf{x}^{(1)} \\ &= (-5, 0) + 2.75(2, -1) \\ &= (0.5, -2.75)\end{aligned}$$

ALGORITHM 3A: CONTINUOUS IMPROVING SEARCH

Step 0: Initialization. Choose any starting feasible solution $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Local Optimum. If no improving feasible direction $\Delta \mathbf{x}$ exists at current solution $\mathbf{x}^{(t)}$, stop. Under mild assumptions about the form of the model, point $\mathbf{x}^{(t)}$ is a local optimum.

Step 2: Move Direction. Construct an improving feasible direction at $\mathbf{x}^{(t)}$ as $\Delta \mathbf{x}^{(t+1)}$.

Step 3: Step Size. If there is a limit on step sizes for which direction $\Delta \mathbf{x}^{(t+1)}$ continues to both improve the objective function and retain feasibility, choose the largest such step size as λ_{t+1} . If not, stop; the model is unbounded.

Step 4: Advance. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda_{t+1} \Delta \mathbf{x}^{(t+1)}$$

Then, increment $t \leftarrow t + 1$, and return to Step 1.

Our first iteration is now complete. Incrementing $t \leftarrow 1$, we return to step 1. Improving feasible directions still exist. This time the search selected $\Delta \mathbf{x}^{(2)} = (-4, -1)$. As before, the maximum appropriate step size is determined by progress in the objective function. Picking $\lambda_2 = 0.25$ leads to

$$\begin{aligned}\mathbf{x}^{(2)} &\leftarrow \mathbf{x}^{(1)} + \lambda_2 \Delta \mathbf{x}^{(2)} \\ &= (0.5, -2.75) + 0.25(-4, -1) \\ &= (-0.5, -3)\end{aligned}$$

Returning again to step 1, we begin a third iteration. Improving feasible directions still exist, and the search chose $\Delta \mathbf{x}^{(3)} = (-1, 1)$. Unlike previous iterations, feasibility is now a consideration. A step of 0.5 in that direction brings us up against a constraint, even though further objective progress is possible. Thus $\lambda_3 = 0.5$, and

$$\begin{aligned}\mathbf{x}^{(3)} &\leftarrow \mathbf{x}^{(2)} + \lambda_3 \Delta \mathbf{x}^{(3)} \\ &= (-0.5, -3) + 0.5(1, -1) \\ &= (0, -3.5)\end{aligned}$$

Upon still another return to step 1, no improving feasible direction is apparent. Algorithm 3A terminates with locally optimal solution $\mathbf{x} = (0, -3.5)$.

When Improving Search Stops

Algorithm 3A keeps going while improving feasible directions are available because local progress is still possible.

Principle 3.16 No optimization model solution at which an improving feasible direction is available can be a local optimum.

Since a direction improves and maintains feasibility for the smallest of steps $\lambda > 0$, every neighborhood includes a better point.

What if the algorithm stops at a point admitting no improving feasible direction? In most applications the result is a local optimum (definition 3.5).

Principle 3.17 When a continuous improving search terminates at a solution admitting no improving feasible direction, and mild assumptions hold, the point is a local optimum.

Figures 3.3 and 3.5 illustrate this typical case. Both searches lead us to points where no improving feasible direction is apparent. Both results are local maxima.

The “mild assumptions” caveat appears in principle 3.17 because it is possible to make up examples where there are no improving feasible directions at the current solution, yet it is not a local optimum. Figure 3.8 shows two. The \mathbf{w} in part (a) is not a local minimum of the unconstrained model indicated because it is possible to decrease the unusually shaped objective function for arbitrarily small steps along the curved path shown. Still, every straight-line direction fails to improve. Thus Algorithm 3A would terminate at \mathbf{w} because there are no improving directions.

Even if the objective function has a very simple form, constraints can cause the same sorts of anomalies. Algorithm 3A would also terminate at the point \mathbf{x} shown in part (b) because no (straight-line) feasible direction leads to higher objective function contours. Still, there are feasible solutions arbitrarily close to \mathbf{x} with superior objective values. Stopping point \mathbf{x} cannot be a local maximum.

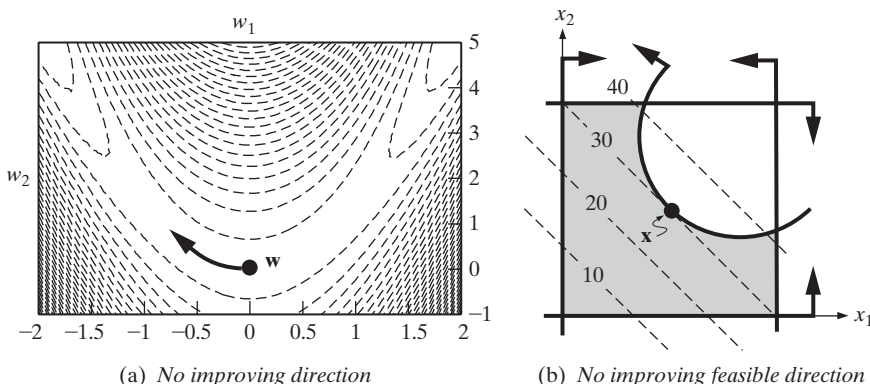


FIGURE 3.8 Nonoptimal Points Admitting No Improving Feasible Directions

Fortunately, such examples are rare for the standard models treated in this book. In the interest of efficiency, analysts are almost always willing to accept the result of an improving search as at least a local optimum.

Detecting Unboundedness

Most of the time, improving searches following Algorithm 3A will terminate at step 1 because there is no improving feasible direction. However, they can also stop in λ -choosing step 3.

An optimization model is **unbounded** if it admits feasible solutions with arbitrarily good objective value (definition [2.20](#)).

Principle 3.18 If an improving search discovers an improving feasible direction for a model that can be pursued forever without ceasing to improve or losing feasibility, the model is unbounded.

Figure 3.9 shows such a problem and an application of Algorithm 3A starting at $\mathbf{y}^{(0)} = (1, 0)$. Contour lines illustrate how the objective value improves forever as y_1 becomes large.

The first iteration depicted in Figure 3.9 reached step 3 with chosen direction $\Delta \mathbf{y}^{(1)} = (1, 2)$. Progress in that direction is limited by constraints, so the step was fixed at the maximum $\lambda_1 = 2$ that retains feasibility (principle [3.15](#)). On the next iteration, however, the chosen improving feasible direction is $\Delta \mathbf{y}^{(2)} = (1, 0)$. Step 3 terminates with a conclusion of unboundedness because there is no limit on step sizes improving the objective and retaining feasibility in that direction.

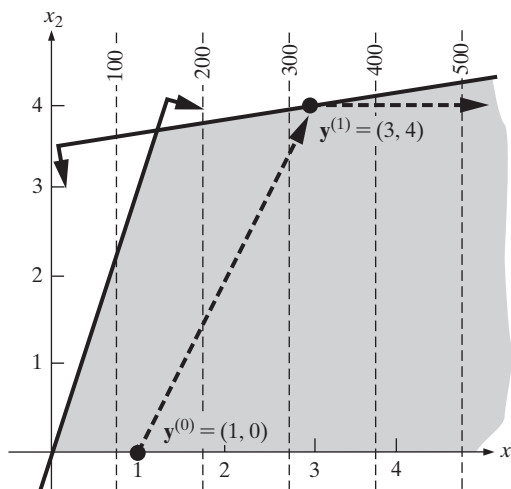


FIGURE 3.9 Improving Search of an Unbounded Model

EXAMPLE 3.8: DETECTING UNBOUNDEDNESS IN IMPROVING SEARCH

An improving search begins at $\mathbf{w}^{(0)} = (0, 0)$ in the model

$$\begin{array}{ll} \min & w_2 \\ \text{s.t.} & 0 \leq w_1 \leq 1 \end{array}$$

- (a) Explain why the model is unbounded.
 (b) Identify an improving feasible direction $\Delta \mathbf{w}$ at $\mathbf{w}^{(0)}$ which demonstrates that the model is unbounded.

Solution:

- (a) Variable w_2 has no lower bound in the model. Thus we may leave w_1 at feasible value 0 and decrease w_2 to produce arbitrarily good objective function values.
 (b) For the reasons given in part (a), arbitrarily good feasible solutions are obtained by leaving w_1 unchanged and decreasing w_2 . The obvious move direction to accomplish this is $\Delta \mathbf{w} = (0, -1)$. There is no limit on the $\lambda > 0$ that may be applied to this direction while maintaining feasibility and improving the objective function.

3.3 ALGEBRAIC CONDITIONS FOR IMPROVING AND FEASIBLE DIRECTIONS

What distinguishes one implementation of improving search Algorithm 3A from another is the process employed to identify an improving feasible direction at step 2 (or to prove that none exists). We are now ready to develop the fundamental algebraic conditions at the heart of nearly every such construction.

Readers are advised to dwell on these simple conditions and to experiment with their own examples and plots until each idea has been absorbed completely. We will return to the algebra of this section many times as we develop the algorithms of subsequent chapters.

Gradients

To obtain algebraic characterizations of improving directions, we resort to a bit of differential calculus. If the objective function of our optimization model is **smooth** (i.e., differentiable with respect to all decision variables), simple conditions can easily be devised. Readers whose calculus is a bit inadequate or rusty should not panic. The brief synopsis in Primer 2 includes everything you will need.

When f is a function of n -vector $\mathbf{x} \triangleq (x_1, \dots, x_n)$, it has n first partial derivatives. Our principal interest is the n -vector of such partial derivatives known as the **gradient**.

Definition 3.19 | The gradient of $f(\mathbf{x}) \triangleq f(x_1, \dots, x_n)$, denoted $\nabla f(\mathbf{x})$, is the vector of partial derivatives $\nabla f(\mathbf{x}) \triangleq (\partial f / \partial x_1, \dots, \partial f / \partial x_n)$ evaluated at \mathbf{x} .

The gradient describes the shape of the objective function because each partial derivative of the objective function at any current solution quantifies the slope or rate of change per unit change in one of the coordinate directions.

To illustrate, we can return to patronage objective function (3.1) of Section 3.1’s DClub example:

$$\max p(x_1, x_2) \triangleq \frac{60}{1 + (x_1 + 1)^2 + (x_2 - 3)^2} + \frac{20}{1 + (x_1 - 1)^2 + (x_2 - 3)^2} + \frac{30}{1 + (x_1)^2 + (x_2 + 4)^2}$$

PRIMER 2: DERIVATIVES AND PARTIAL DERIVATIVES

One of the major concepts of calculus is **derivatives**—rates of change in the value of a function with respect to small increases in its arguments. In this book we assume only an elementary understanding of derivatives, but we will need to compute them occasionally and to have an intuitive feel for derivatives as “slopes.”

A function is **differentiable** or **smooth** at a point if its rate of change is unambiguous (i.e., if the function has no sudden rate changes). The prototypical example of a function that is not always differentiable is $f(x) \triangleq |x|$. At $x = 0$ the function makes a stark change from derivative -1 applicable for $x < 0$ to the $+1$ appropriate for $x > 0$. No derivative exists for x exactly zero.

For a function $f(x)$ of a single variable x , it is customary to denote the derivative of f with respect to x by df/dx or $f'(x)$. Thus constant function $f(x) \triangleq a$ has derivative $f'(x) = 0$ at every x because its value does not change with x .

Many familiar functions have easily expressed derivatives (a constant):

$f(x)$	$\frac{df}{dx}$	$f(x)$	$\frac{df}{dx}$	$f(x)$	$\frac{df}{dx}$
ax	ax	x^a	ax^{a-1}	$\sin(ax)$	$a \cos(ax)$
a^x	$a^x \ln(a)$	$\ln(ax)$	$\frac{1}{x}$	$\cos(ax)$	$-a \sin(ax)$

Also, a variety of computing formulas determine the derivative of a function $f(x)$ in terms of simpler functions g and h forming f :

$f(x)$	$\frac{df}{dx}$	$f(x)$	$\frac{df}{dx}$
$g(h(x))$	$\frac{dg}{dh} \cdot \frac{dh}{dx}$	$g(x) \cdot h(x)$	$g(x) \frac{dh}{dx} + h(x) \frac{dg}{dx}$
$g(x) \pm h(x)$	$\frac{dg}{dx} \pm \frac{dh}{dx}$	$\frac{g(x)}{h(x)}$	$\left[h(x) \frac{dg}{dx} - g(x) \frac{dh}{dx} \right] / h(x)^2$

For example, the derivative of $f(x) \triangleq (3x)^4$ can be computed as $dg/dh \cdot dh/dx = 4(-3x)^3(-3)$ by taking $g(h) \triangleq h^4$ and $h(x) \triangleq 3x$. Similarly, the derivative with

respect to x of $f(x) \triangleq (4x)(e^x)$ can be computed as $(4x)(e^x)(1) + (e^x)(4)$ by thinking of f as the product of functions $g(x) \triangleq 4x$ and $h(x) \triangleq e^x$.

When a function has more than one argument, **partial derivatives** show rates of change with respect to single variables with all others held constant. The partial derivatives of $f(x_1, x_2, \dots, x_n)$ are usually denoted $\partial f/\partial x_i$, $i = 1, 2, \dots, n$. To illustrate, consider $f(x_1, x_2, x_3) \triangleq (x_1)^5(x_2)^7(x_3)$. Differentiating with x_2 and x_3 treated as constants produces the partial derivative $\partial f/\partial x_1 = 5(x_1)^4(x_2)^7(x_3)$. Similarly, $\partial f/\partial x_2 = 7(x_1)^5(x_2)^6(x_3)$ and $\partial f/\partial x_3 = (x_1)^5(x_2)^7$.

Figure 3.10 shows its now familiar contours. Differentiating yields

$$\nabla p(x_1, x_2) \triangleq \begin{pmatrix} \frac{\partial p}{\partial x_1} \\ \frac{\partial p}{\partial x_2} \end{pmatrix} = \begin{pmatrix} -\frac{120(x_1 + 1)}{[1 + (x_1 + 1)^2 + (x_2 - 3)^2]^2} - \frac{40(x_1 - 1)}{[1 + (x_1 - 1)^2 + (x_2 - 3)^2]^2} - \frac{60(x_1)}{[1 + (x_1)^2 + (x_2 + 4)^2]^2} \\ -\frac{120(x_2 - 3)}{[1 + (x_1 + 1)^2 + (x_2 - 3)^2]^2} - \frac{40(x_2 - 3)}{[1 + (x_1 - 1)^2 + (x_2 - 3)^2]^2} - \frac{60(x_2 + 4)}{[1 + (x_1)^2 + (x_2 + 4)^2]^2} \end{pmatrix} \quad (3.3)$$

Thus at point $\mathbf{x} = (2, 0)$.

$$\nabla p(2, 0) \approx (-1.60, 1.45)$$

Thus in the neighborhood of $\mathbf{x} = (2, 0)$, function p declines at the rate of about -1.60 per unit increase in x_1 and grows at the rate of roughly 1.45 per unit increase in x_2 . Of course, these rates may change if we move away from $(2, 0)$, but they provide fairly complete information at points in the immediate neighborhood.

Figure 3.10 also illustrates the geometry of gradient vectors.

Principle 3.20 Gradients show graphically as vectors perpendicular to contours of the objective function and point in the direction of most rapid objective value increase.

For example, at $\mathbf{x} = (2, 0)$ the objective increases fastest by moving at right angles to the contour in direction $\Delta \mathbf{x} = (-1.60, 1.45)$, which is precisely $\nabla p(2, 0)$.

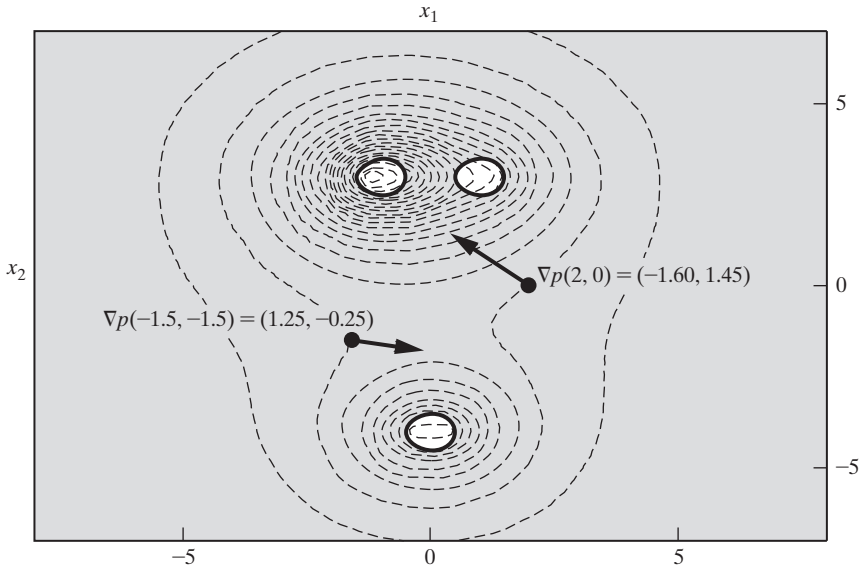


FIGURE 3.10 Gradients of the DClub Example Objective

Gradient Conditions for Improving Directions

Suppose that our search of objective function f has arrived at current solution \mathbf{x} . Then the change associated with a step of size λ in direction $\Delta \mathbf{x}$ can be approximated as¹

$$\text{objective change} \approx \sum_j \left(\frac{\partial f}{\partial x_j} \right) (\lambda \Delta x_j) = \lambda (\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x})$$

That is, the rate of objective function change near the current \mathbf{x} is roughly the dot product of $\nabla f(\mathbf{x})$ and $\Delta \mathbf{x}$ because it is approximated by the weighted sum of rates of change described by partial derivatives times move components in the various coordinate directions.

When dot product $\nabla f \cdot \Delta \mathbf{x} \neq 0$, this gradient information provides a simple algebraic test of whether a direction fulfills definition [3.13] as improving:

Principle 3.21 | Direction $\Delta \mathbf{x}$ is improving for maximize objective function f at point \mathbf{x} if $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} > 0$. On the other hand, if $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} < 0$, $\Delta \mathbf{x}$ does not improve at \mathbf{x} .

Principle 3.22 | Direction $\Delta \mathbf{x}$ is improving for minimize objective function f at point \mathbf{x} if $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} < 0$. On the other hand, if $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} > 0$, $\Delta \mathbf{x}$ does not improve at \mathbf{x} .

¹This Taylor series approximation is developed more formally in Section 16.3.

Cases where dot product $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} = 0$ cannot be resolved without further information.

To illustrate, consider again the patronage function of Figure 3.10. We have already computed the gradient at $\mathbf{x} = (2, 0)$ as $\nabla p(2, 0) \approx (-1.60, 1.45)$. A glance at the figure shows that $\Delta \mathbf{x} = (-1, 1)$ is an improving direction because it leads us to higher contours. Confirming characterization [3.21], we have

$$\nabla p(2, 0) \cdot \Delta \mathbf{x} \approx (-1.60, 1.45) \cdot (-1, 1) = 3.05 > 0$$

On the other hand, Figure 3.10 reports that evaluation of gradient expression (3.3) at $\mathbf{x} = (-1.5, -1.5)$ yields $\nabla p(-1.5, -1.5) \approx (1.25, -0.25)$. Testing the same direction $\Delta \mathbf{x} = (-1, 1)$, we see that

$$\nabla p(-1.5, -1.5) \cdot \Delta \mathbf{x} \approx (1.25, -0.25) \cdot (-1, 1) = -1.50 < 0$$

which verifies that direction $\Delta \mathbf{x} = (-1, 1)$ does not improve at $(-1.5, -1.5)$.

EXAMPLE 3.9: USING GRADIENTS TO DETERMINE IF DIRECTIONS IMPROVE

Either determine by an appropriate gradient test whether each of the following directions is improving for the specified objective function and point or show why further information is required.

- (a) $\Delta \mathbf{w} = (1, 0, -2)$ for minimize $f(\mathbf{w}) \triangleq (w_1)^2 + 5w_2w_3$ at $\mathbf{w} = (2, 1, 0)$.
- (b) $\Delta \mathbf{y} = (3, -6)$ for maximize $f(\mathbf{y}) \triangleq 9y_1 + 40y_2$ at $\mathbf{y} = (13, 2)$.
- (c) $\Delta \mathbf{z} = (-6, 2)$ for minimize $f(\mathbf{z}) \triangleq 5(z_1)^2 - 3z_1z_2 + (z_2)^2$ at $\mathbf{z} = (1, 3)$.

Solution:

- (a) Computing the objective function gradient at the indicated \mathbf{w} yields

$$\nabla f(\mathbf{w}) = \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \frac{\partial f}{\partial w_3} \end{pmatrix} = \begin{pmatrix} 2w_1 \\ 5w_3 \\ 5w_2 \end{pmatrix} = \begin{pmatrix} 2(2) \\ 5(0) \\ 5(1) \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 5 \end{pmatrix}$$

so that

$$\nabla f(\mathbf{w}) \cdot \Delta \mathbf{w} = (4, 0, 5) \cdot (1, 0, -2) = -6 < 0$$

By applying condition [3.22], we conclude that $\Delta \mathbf{w}$ does improve at \mathbf{w} for the minimize objective.

(b) Computing the objective function gradient at the \mathbf{y} indicated gives

$$\nabla f(\mathbf{y}) = \begin{pmatrix} \frac{\partial f}{\partial y_1} \\ \frac{\partial f}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 9 \\ 40 \end{pmatrix}$$

so that

$$\nabla f(\mathbf{y}) \cdot \Delta \mathbf{y} = (9, 40) \cdot (3, -6) = -213 < 0$$

Thus by condition [3.21](#) the direction does not improve for the maximize objective.

(c) Computing partial derivatives of the objective function at the indicated \mathbf{z} , we have

$$\nabla f(\mathbf{z}) = \begin{pmatrix} \frac{\partial f}{\partial z_1} \\ \frac{\partial f}{\partial z_2} \end{pmatrix} = \begin{pmatrix} 10z_1 - 3z_2 \\ -3z_1 + 2z_2 \end{pmatrix} = \begin{pmatrix} 10(1) - 3(3) \\ -3(1) + 2(3) \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

and

$$\nabla f(\mathbf{z}) \cdot \Delta \mathbf{z} = (1, 3) \cdot (-6, 2) = 0$$

With the dot product = 0, conditions [3.21](#) and [3.22](#) are insufficient to determine whether this $\Delta \mathbf{z}$ improves.

Objective Function Gradients as Move Directions

One consequence of dot product tests [3.21](#) and [3.22](#) is that we can derive improving directions directly from any nonzero gradient (although we will see in Chapters 16 and 17 that gradients are not always the best directions to choose). Since the dot product of a nonzero gradient with itself is

$$\nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x}) = \sum_j \left(\frac{\partial f}{\partial x_j} \right)^2 > 0$$

we need only choose $\Delta \mathbf{x} = \pm \nabla f(\mathbf{x})$.

Principle 3.23 When objective function gradient $\nabla f(\mathbf{x}) \neq \mathbf{0}$, $\Delta \mathbf{x} = \nabla f(\mathbf{x})$ is an improving direction for a maximize objective f , and $\Delta \mathbf{x} = -\nabla f(\mathbf{x})$ is an improving direction for minimizing f .

Contours confirm that both of the gradients displayed for the maximize model in Figure 3.10 are indeed improving directions. We can verify the first algebraically by choosing

$$\Delta \mathbf{x} = \nabla p(0, 2) = (-1.60, 1.45)$$

then (principle [3.21](#)) the dot product

$$\nabla p(0, 2) \cdot \Delta \mathbf{x} = (-1.60, 1.45) \cdot (-1.60, 1.45) = (-1.60)^2 + (1.45)^2 > 0$$

establishes that $\Delta \mathbf{x}$ improves at $\mathbf{x} = (0, 2)$.

EXAMPLE 3.10: CONSTRUCTING IMPROVING DIRECTIONS FROM GRADIENTS

Use the gradient of each of the following objective functions f to construct an improving direction at the indicated point.

(a) Minimize $f(\mathbf{w}) \triangleq (w_1)^2 \ln(w_2)$ at $\mathbf{w} = (5, 2)$.

(b) Maximize $f(\mathbf{y}) \triangleq 4y_1 + 5y_2 - 8y_3$ at $\mathbf{y} = (2, 0, 0.5)$.

Solution:

(a) Computing the gradient at the indicated point yields

$$\nabla f(\mathbf{w}) \triangleq \begin{pmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \end{pmatrix} = \begin{pmatrix} 2w_1 \ln(w_2) \\ \frac{(w_1)^2}{w_2} \end{pmatrix} = \begin{pmatrix} 2(5) \ln(2) \\ \frac{(5)^2}{2} \end{pmatrix} \approx \begin{pmatrix} 6.93 \\ 12.5 \end{pmatrix}$$

Since this gradient is nonzero, we may apply construction [3.23](#). Negative gradient

$$\Delta \mathbf{w} = -\nabla f(\mathbf{w}) = \begin{pmatrix} -6.93 \\ -12.5 \end{pmatrix}$$

must improve for the minimize objective.

(b) Computing the gradient at the indicated point yields

$$\nabla f(\mathbf{y}) \triangleq \begin{pmatrix} \frac{\partial f}{\partial y_1} \\ \frac{\partial f}{\partial y_2} \\ \frac{\partial f}{\partial y_3} \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ -8 \end{pmatrix}$$

Since this gradient is nonzero, we may again apply construction [3.23](#). Gradient

$$\Delta \mathbf{y} = \nabla f(\mathbf{y}) = \begin{pmatrix} 4 \\ 5 \\ -8 \end{pmatrix}$$

will improve for the maximize objective.

Active Constraints and Feasible Directions

Turning now to conditions for feasible directions (definition [3.14](#)), we focus our attention on constraints. Constraints define the boundary of the feasible region for an optimization model, so they will be the source of conditions for feasible directions.

Figure 3.11's repeat of our familiar Two Crude example demonstrates that not all the constraints of a model are relevant to whether a direction $\Delta \mathbf{x}$ is feasible at a particular solution \mathbf{x} . For example, at $\mathbf{x}^{(1)} = (7, 0)$ only

$$x_2 \geq 0$$

poses a threat to feasibility; small movements from $\mathbf{x}^{(1)}$ cannot violate other constraints. At $\mathbf{x}^{(3)} = (0.75, 6)$ the relevant constraints are different:

$$x_2 \leq 6 \quad \text{and} \quad 0.4x_1 + 0.2x_2 \geq 1.5$$

Principle 3.24 Whether a direction is feasible at a solution \mathbf{x} depends on whether it would lead to immediate violation of any **active constraint** at \mathbf{x} , that is, any constraint satisfied as equality at \mathbf{x} .

Other names for active constraints are **tight constraints** and **binding constraints**.

Equality constraints are active at every feasible point, because they are always satisfied as equalities. Inequalities are more complex. For example, the Two Crude constraint

$$0.4x_1 + 0.2x_2 \geq 1.5$$

is satisfied, but not active at $\mathbf{x}^{(1)} = (7, 0)$ of Figure 3.11 because

$$0.4(7) + 0.2(0) = 2.8 > 1.5$$

However, the same inequality is active at $\mathbf{x}^* = (2, 3.5)$. There

$$0.4(2) + 0.2(3.5) = 1.5$$

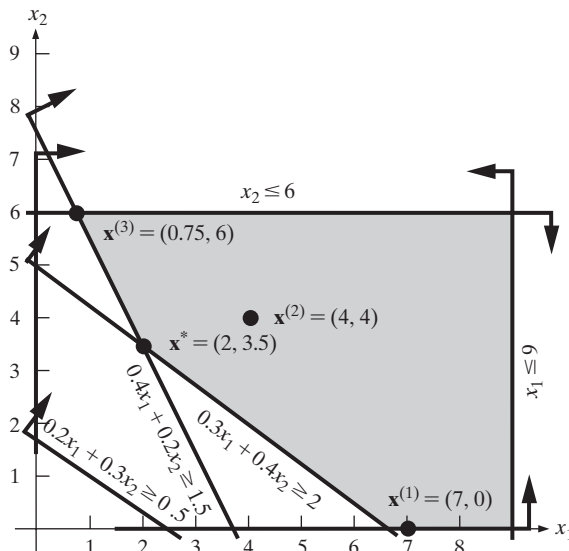


FIGURE 3.11 Active Constraints of the Two Crude Example

EXAMPLE 3.11: RECOGNIZING ACTIVE CONSTRAINTS

Consider an optimization model with main constraints

$$(w_1 - 4)^2 + 3w_2 - 7w_3 \geq 31 \quad (3.4)$$

$$2w_1 + w_3 \leq 13 \quad (3.5)$$

$$w_1 + w_2 + w_3 = 25 \quad (3.6)$$

Determine which of these constraints are active at feasible solutions

(a) $\mathbf{w} = (8, 20, -3)$ (b) $\mathbf{w} = (5, 17, 3)$

Solution: Equality (3.6) must be active at every feasible solution.

(a) Checking definition 3.24 for inequalities (3.4) and (3.5) at $\mathbf{w} = (8, 20, -3)$, we have

$$(8 - 4)^2 + 3(20) - 7(-3) = 97 > 31$$

and

$$2(8) + (-3) = 13$$

Thus only (3.5) and (3.6) are active at $\mathbf{w} = (8, 20, -3)$.

(b) Checking definition 3.24 for inequalities (3.4) and (3.5) at $\mathbf{w} = (5, 17, 3)$, we find that

$$(5 - 4)^2 + 3(17) - 7(3) = 31$$

and

$$2(5) + (3) = 13$$

Thus all three constraints are active at $\mathbf{w} = (5, 17, 3)$.

Linear Constraints

Constraints of the Two Crude example are all **linear**, that is, inequalities or equalities constraining a weighted sum of the decision variables by a right-hand-side constant (definition 2.28). We may denote general forms by

$$\mathbf{a} \cdot \mathbf{x} \triangleq \sum_{j=1}^n a_j x_j \geq b \quad (3.7)$$

$$\mathbf{a} \cdot \mathbf{x} \triangleq \sum_{j=1}^n a_j x_j \leq b \quad (3.8)$$

$$\mathbf{a} \cdot \mathbf{x} \triangleq \sum_{j=1}^n a_j x_j = b \quad (3.9)$$

where n denotes the number of decision variables, a_j is the constraint's coefficient for decision variable x_j , \mathbf{a} is the vector of coefficients a_j , and b is the constraint's right-hand side.

One example is the first main Two Crude constraint,

$$0.3x_1 + 0.4x_2 \geq 2$$

In the notation of (3.7),

$$n = 2, \quad a_1 = 0.3, \quad a_2 = 0.4, \quad \mathbf{a} = (0.3, 0.4), \quad \text{and} \quad b = 2$$

Conditions for Feasible Directions with Linear Constraints

Conditions can be devised to characterize feasible directions in more complex cases (see Section 14.4), but we focus here on linear forms (3.7) to (3.9). Consider the Two Crude model constraint

$$0.4x_1 + 0.2x_2 \geq 1.5$$

At solution $\mathbf{x}^{(3)} = (0.75, 6)$ the constraint is active because

$$0.4(0.75) + 0.2(6) = 1.5$$

A step in move direction $\Delta \mathbf{x} \triangleq (\Delta x_1, \Delta x_2)$ will change the left-hand side as follows:

$$0.4(0.75 + \lambda \Delta x_1) + 0.2(6 + \lambda \Delta x_2) = 1.5 + \lambda(0.4 \Delta x_1 + 0.2 \Delta x_2)$$

Feasibility limit

$$1.5 + \lambda(0.4 \Delta x_1 + 0.2 \Delta x_2) \geq 1.5$$

can be maintained only if we restrict the sign “net change” coefficient of λ by requiring

$$\sum_{j=1}^n a_j \Delta x_j = 0.4 \Delta x_1 + 0.2 \Delta x_2 \geq 0$$

This sort of analysis leads immediately to general conditions for directions feasible to linear constraints:

Principle 3.25 | Direction $\Delta \mathbf{x} \triangleq (\Delta x_1, \dots, \Delta x_n)$ is feasible for a linearly constrained optimization model at solution $\mathbf{x} \triangleq (x_1, \dots, x_n)$ if and only if

$$\mathbf{a} \cdot \Delta \mathbf{x} \triangleq \sum_{i=1}^n a_i \Delta x_i \geq 0$$

for all active greater than or equal to constraints $\sum_j a_j x_j \geq b$;

$$\mathbf{a} \cdot \Delta \mathbf{x} \triangleq \sum_{i=1}^n a_i \Delta x_i \leq 0$$

for all active less than or equal to constraints $\sum_j a_j x_j \leq b$; and

$$\mathbf{a} \cdot \Delta \mathbf{x} \triangleq \sum_{i=1}^n a_i \Delta x_i = 0$$

for all equality constraints $\sum_j a_j x_j = b$.

To illustrate, return to Figure 3.11. At $\mathbf{x}^{(2)} = (4, 4)$, no constraints are active, and every direction is feasible.

At $\mathbf{x}^{(3)} = (0.75, 6)$, there are two active constraints:

$$0.4x_1 + 0.2x_2 \geq 1.5$$

$$x_2 \leq 6$$

If direction $\Delta \mathbf{x}$ is to be feasible, it must violate neither. Thus principle [3.25](#) yields one condition for an active \geq form and one for an active \leq :

$$0.4 \Delta x_1 + 0.2 \Delta x_2 \geq 0$$

$$\Delta x_2 \leq 0$$

EXAMPLE 3.12: FORMING CONDITIONS FOR A DIRECTION TO BE FEASIBLE

Consider an optimization model with linear constraints

$$3w_1 + w_3 \geq 26 \tag{3.10}$$

$$5w_1 - 2w_3 \leq 50 \tag{3.11}$$

$$2w_1 + w_2 + w_3 = 20 \tag{3.12}$$

$$w_1 \geq 0 \tag{3.13}$$

$$w_2 \geq 0 \tag{3.14}$$

State all conditions that must be satisfied for $\Delta \mathbf{w}$ to be a feasible move direction at $\mathbf{w} = (10, 0, 0)$.

Solution: When we substitute the solution $\mathbf{w} = (10, 0, 0)$, the active constraints are (3.11), (3.12), and (3.14). Thus, applying conditions [3.25](#), the corresponding requirements for a feasible direction are

$$5 \Delta w_1 - 2 \Delta w_3 \leq 0$$

$$2 \Delta w_1 + \Delta w_2 + \Delta w_3 = 0$$

$$\Delta w_2 \geq 0$$

EXAMPLE 3.13: TESTING DIRECTIONS FOR FEASIBILITY

Return to the feasible region defined by constraints (3.10) to (3.14) and determine whether direction $\Delta \mathbf{w} = (0, -1, 1)$ is feasible at point $\mathbf{w} = (6, 0, 8)$.

Solution: At $\mathbf{w} = (6, 0, 8)$, the active constraints are (3.10), (3.12), and (3.14). Thus required conditions [3.25](#) for a feasible direction are

$$3 \Delta w_1 + \Delta w_3 \geq 0$$

$$2 \Delta w_1 + \Delta w_2 + \Delta w_3 = 0$$

$$\Delta w_2 \geq 0$$

Direction $\Delta \mathbf{w} = (0, -1, 1)$ meets the first two conditions because

$$\begin{aligned} 3\Delta w_1 + \Delta w_3 &= 3(0) + (1) \geq 0 \\ 2\Delta w_1 + \Delta w_2 + \Delta w_3 &= 2(0) + (-1) + (1) = 0 \end{aligned}$$

Still, the direction is not feasible because it violates the third condition,

$$\Delta w_2 = (-1) \not\geq 0$$

3.4 TRACTABLE CONVEX AND LINEAR CASES

Tractability in a model means convenience for analysis (definition [1.8](#)). Sometimes difficult forms cannot be avoided, but in Chapter 1 (principle [1.10](#)) we saw that modeling often involves tradeoffs. To obtain a model that is tractable enough to yield useful insights, we sometimes have the choice of making simplifying assumptions or other compromises. It is important to be able to identify the preferred cases.

Earlier sections of this chapter have provided a broad introduction to Improving Search and its challenges including whether local optima may be assumed to be global. We are now ready to build on details of Sections 3.2 and 3.3 to define the most tractable models forms commonly encountered in terms of whether they assure local optima are global. Such forms are preferred if the implied models are sufficiently valid for the application.

Special Tractability of Linear Objective Functions

Recall that a **linear objective function** is one expressible as a weighted sum of the decision variables (definition [2.28](#)). The general form is

$$\min \text{ or } \max f(\mathbf{x}) \equiv \sum_{j=1}^n c_j x_j = \mathbf{c} \cdot \mathbf{x}$$

where \mathbf{x} is the n -vector of the n decision variable, and \mathbf{c} is the corresponding n -vector of objective function coefficients. For instance, linear objective

$$\min \quad 3.5x_1 - 2x_2 + x_3$$

has weights $c_1 = 3.5$, $c_2 = -2$, and $c_3 = 1$.

Testing for whether a direction improves with a linear objective is particularly easy under principles [3.21](#) and [3.22](#) because the gradient of such objectives is simply constant $\nabla f = \mathbf{c}$.

Principle 3.26 Direction $\Delta \mathbf{x}$ is improving for a maximizing objective function $\mathbf{c} \cdot \mathbf{x}$ at any feasible point if and only if $\mathbf{c} \cdot \Delta \mathbf{x} > 0$, and for a minimizing objective if and only if $\mathbf{c} \cdot \Delta \mathbf{x} < 0$.

In the minimizing example above, this means direction $\Delta \mathbf{x}$ is improving if and only if

$$\mathbf{c} \cdot \Delta \mathbf{x} = 3.5 \Delta x_1 - 2 \Delta x_2 + \Delta x_3 < 0$$

The convenience of this extra tractability arises as we decide step sizes λ , that is, how far a direction can be pursued at step 3 of Improving Search Algorithm 3A.

With a linear objective, an improving direction remains improving at all feasible solutions. Thus, the objective need not be considered in fixing the step limit.

Constraints and Local Optima

Objective functions are not the only parts of mathematical programming models that can induce local optima. For example, the only unconstrained local maximum in the part of the DClub feasible space displayed in Figure 3.12 occurs at unconstrained global optimum $\mathbf{x} = (-1, 3)$.

Solutions $\mathbf{x}^{(1)} = (-1.5, 3)$ and $\mathbf{x}^{(2)} = (1.5, 3)$ illustrate a new, constraint-induced form of local optimum. Both are (constrained) local maxima in the full model that do not achieve as high an objective value as global maximum $\mathbf{x}^* = (-0.5, 3)$. The difficulty is not the shape of the objective function, because improving directions $\Delta \mathbf{x} = (\mathbf{x}^* - \mathbf{x}^{(1)})$ at $\mathbf{x}^{(1)}$ and $\Delta \mathbf{x} = (\mathbf{x}^* - \mathbf{x}^{(2)})$ at $\mathbf{x}^{(2)}$ leads straight to the global optimum. The difficulty is constraints. No neighboring point at either $\mathbf{x}^{(1)}$ or $\mathbf{x}^{(2)}$ both improves the objective function and satisfies all constraints.

Convex Feasible Sets

Convex feasible sets avoid such difficulties.

Definition 3.27 | The feasible set of an optimization problem is **convex** if the line segment between every pair of feasible points falls entirely within the feasible region.

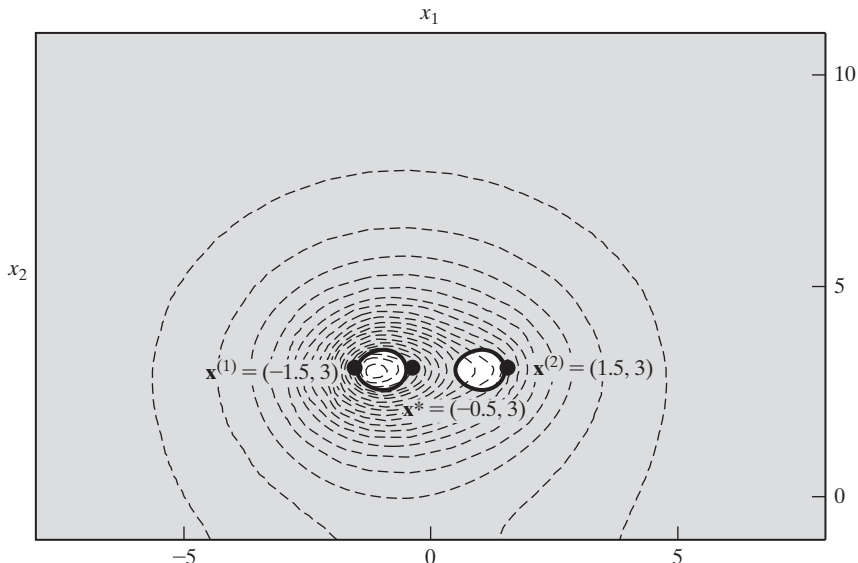
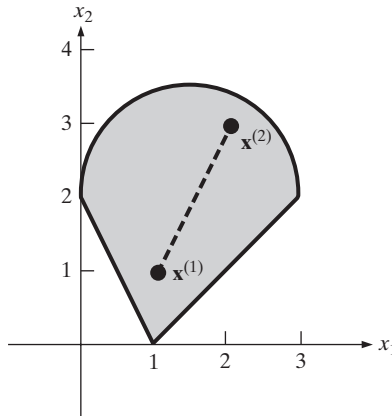


FIGURE 3.12 Constraint-Induced Local Optima for the DClub Example

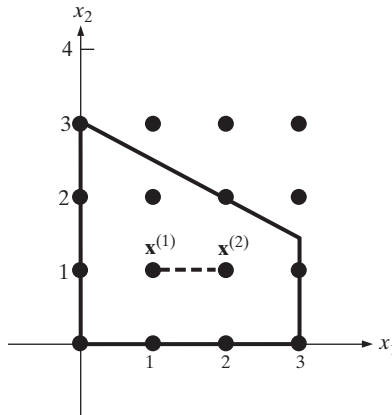
The idea is illustrated by the following feasible set:



This set is convex because the line segment between every pair of feasible points lies entirely within the feasible set. The segment for $\mathbf{x}^{(1)} = (1, 1)$ and $\mathbf{x}^{(2)} = (2, 3)$ is shown.

Contrast with the, DClub feasible region of Figure 3.12. That set is not convex because some pairs of points fail definition [3.27](#). One of the many is $\mathbf{x}^{(1)} = (-1.5, 3)$ and $\mathbf{x}^{(2)} = (1.5, 3)$. The line segment joining them includes infeasible solutions in both of the forbidden boxes around population centers.

A similar situation occurs with discrete feasible sets such as

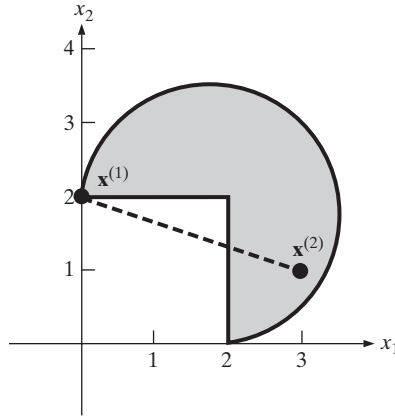


Here only the black integer points within the main constraints are feasible. Thus, the line segment connecting any pair of feasible points passes primarily through noninteger points violating constraints.

Principle 3.28 | Discrete feasible sets are never convex (except in the trivial case where there is only one feasible point).

EXAMPLE 3.14: DEMONSTRATING NONCONVEXITY GRAPHICALLY

Show graphically that the following feasible region is not convex.



Solution: To show that the set fails definition [3.27], we need to find any pair of feasible solutions joined by a line segment that lies partly outside the feasible region. Solutions $\mathbf{x}^{(1)} = (0, 2)$ and $\mathbf{x}^{(2)} = (3, 1)$ sketched in the figure suffice because the line segment between them clearly includes infeasible points.

Algebraic Description of Line Segments

What does it mean to have a line segment in more than two or three dimensions? There is an easy algebraic characterization. Since direction $(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$ is the straight-line move from $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(2)}$, the line segment between the two points consists exactly of those solutions obtained when we add a fraction of that direction to $\mathbf{x}^{(1)}$.

Definition 3.29 The line segment between vector solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ consists of all points of the form $\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$ with $0 \leq \lambda \leq 1$.

As an example, suppose that $\mathbf{x}^{(1)} = (1, 5, 0)$ and $\mathbf{x}^{(2)} = (0, 1, 2)$ are both feasible. One end of the line segment connecting them is $\mathbf{x}^{(1)}$ with $\lambda = 0$ in [3.29]; the other end is $\mathbf{x}^{(2)}$ with $\lambda = 1$. In between lie all the points with $0 < \lambda < 1$. For instance, the one for $\lambda = 0.25$ is

$$\mathbf{x}^{(1)} + 0.25(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}) = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} + 0.25 \left[\begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} - \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} 0.75 \\ 4 \\ 0.5 \end{pmatrix}$$

If the feasible set is to be convex, $(0.75, 4, 0.5)$ and all other solutions along the line segment must satisfy all constraints.

EXAMPLE 3.15: REPRESENTING LINE SEGMENTS

Return to the line segment joining $\mathbf{x}^{(1)} = (0, 2)$ and $\mathbf{x}^{(2)} = (3, 1)$ in Example 3.15.

(a) Represent the line segment algebraically.

(b) Show algebraically that one point on the line segment is $\mathbf{x} = (1, \frac{5}{3})$.

Solution:

(a) Applying [3.29], the vectors along that line segment are those representable as

$$\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}) = \begin{pmatrix} 0 \\ 2 \end{pmatrix} + \lambda \left[\begin{pmatrix} 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 2 \end{pmatrix} \right] = \begin{pmatrix} 3\lambda \\ 2 - \lambda \end{pmatrix}$$

with $0 \leq \lambda \leq 1$.

(b) To represent $(1, \frac{5}{3})$, we must choose an appropriate λ . If first component $3\lambda = 1$, then $\lambda = \frac{1}{3}$, so that

$$(3\lambda, 2 - \lambda) = (3(\frac{1}{3}), 2 - (\frac{1}{3})) = (1, \frac{5}{3})$$

as required.

EXAMPLE 3.16: DEMONSTRATING NONCONVEXITY ALGEBRAICALLY

Demonstrate algebraically that the feasible set excluding the unit circle with constraint

$$(w_1)^2 + (w_2)^2 \geq 1$$

is not convex.

Solution: We must first pick a pair of feasible solutions joined by a line segment that passes through an infeasible point within the unit circle, say $\mathbf{w}^{(1)} = (-1, 0)$ and $\mathbf{w}^{(2)} = (1, 0)$. Under [3.29], the line segment between these two points includes all vectors expressible as

$$\mathbf{w}^{(1)} + \lambda(\mathbf{w}^{(2)} - \mathbf{w}^{(1)}) = \begin{pmatrix} -1 \\ 0 \end{pmatrix} + \lambda \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} -1 + 2\lambda \\ 0 \end{pmatrix}$$

with $0 \leq \lambda \leq 1$. Thus we have only to choose a λ corresponding to an infeasible point to complete the argument that the set is not convex. One is $\lambda = \frac{1}{2}$, which yields the vector $\mathbf{w} = (0, 0)$ in the expression above, which clearly violates the unit circle constraint.

Convenience of Convex Feasible Sets for Improving Search

The attraction of convex feasible sets when using Improving Search is that there is always a feasible direction $\Delta \mathbf{x} = (\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$ leading from any feasible solution $\mathbf{x}^{(1)}$ to any other $\mathbf{x}^{(2)}$.

Principle 3.30 If the feasible set of an optimization model is convex, there is a feasible direction leading from any feasible solution to any other.

This means that feasibility cannot trap a search at a local optimum when there is a better feasible solution.

Global Optimality of Linear Objectives over Convex Feasible Sets

The simplifying tractability of convex feasible sets and linear objectives can combine to assure us not to worry about Algorithm 3A stopping at a local optimum that is not globally optimal.

Principle 3.31 If Improving Search Algorithm 3A over a linear objective and a convex feasible set stops with a feasible solution \mathbf{x}^* where no improving feasible solution exists, then \mathbf{x}^* is a global optimum. That is, local optima are global optima for models with linear objectives and convex feasible sets.

To see why, suppose there is a better feasible solution than \mathbf{x}^* , say \mathbf{x}' . Then principle 3.30 assures there is a feasible direction $\Delta\mathbf{x} = (\mathbf{x}' - \mathbf{x}^*)$ available at \mathbf{x}^* , and if, say, the problem minimizes,

$$\mathbf{c}\mathbf{x}' < \mathbf{c}\mathbf{x}^* \text{ implies } \mathbf{c}(\mathbf{x}' - \mathbf{x}^*) = \mathbf{c}\Delta\mathbf{x} < 0$$

so that $\Delta\mathbf{x}$ is both improving and feasible at \mathbf{x}^* . Algorithm 3A would not have stopped.

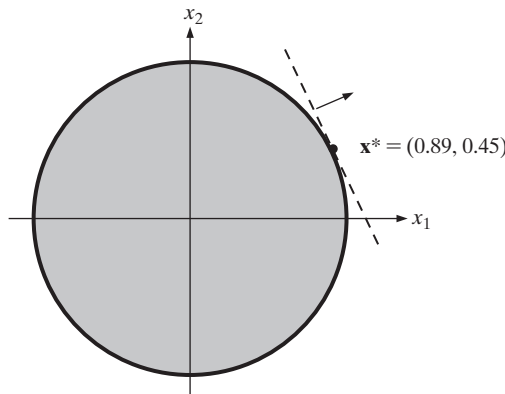
EXAMPLE 3.17: VERIFYING THAT LOCAL OPTIMA FOR LINEAR OBJECTIVES OVER CONVEX FEASIBLE SETS ARE GLOBAL

Consider an optimization problem maximizing objective function $2x_1 + x_2$ over the unit circle with $(x_1)^2 + (x_2)^2 \leq 1$.

- Demonstrate that this problem meets the assumptions of property 3.31.
- Show graphically that the local maximum is indeed global.

Solution:

(a) The plot below shows clearly that the unit circle feasible set is convex. Furthermore, the objective is a weighted sum of the decision variables, and thus linear.



(b) The plot also shows an optimal contour of the objective identifying a solution $\mathbf{x}^* \approx (0.89, 0.46)$, which is locally optimal because every improving direction would take the solution outside the feasible space. From the plot, it is equally clear that the solution is also globally optimum, confirming principle [3.31].

Convexity of Linearly Constrained Feasible Sets

Linear constraints may take any of the forms (3.7–3.9). As with linear objective functions, feasible sets defined by linear constraints, also known as **polyhedral sets**, have unusual tractability.

Principle 3.32 If all constraints of an optimization model are linear (both main and variable-type), its feasible space is convex.

How can we be sure? Pick two solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ feasible in an optimization model with linear constraints. For each model constraint of, say, the \geq form (3.7), feasibility means

$$\sum_{j=1}^n a_j x_j^{(1)} \geq b \quad \text{and} \quad \sum_{j=1}^n a_j x_j^{(2)} \geq b \tag{3.15}$$

What about points along the line segment between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$? We know they are formed as in [3.29] by some step λ with $0 < \lambda < 1$.

Choose any such λ . We may add $(1 - \lambda)$ times the first inequality of (3.15) to λ times the second, to obtain

$$(1 - \lambda) \sum_{j=1}^n a_j x_j^{(1)} + \lambda \sum_{j=1}^n a_j x_j^{(2)} \geq (1 - \lambda)b + \lambda b = b$$

Then a bit of regrouping gives

$$\sum_{j=1}^n a_j x_j^{(1)} - \lambda \sum_{j=1}^n a_j x_j^{(1)} + \lambda \sum_{j=1}^n a_j x_j^{(2)} \geq b$$

or

$$\sum_{j=1}^n a_j \left[x_j^{(1)} + \lambda(x_j^{(2)} - x_j^{(1)}) \right] \geq b$$

The last says that point $\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$ satisfies the \geq constraint. Since we could perform exactly the same computations for each of the other constraints, we may conclude that points along the line segment between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are all feasible—exactly what is required for the feasible set to be convex.

EXAMPLE 3.18: SHOWING THAT LINEARLY CONSTRAINED SETS ARE CONVEX

Establish that the set of (w_1, w_2, w_3) satisfying

$$\begin{aligned} 19w_1 + 3w_2 - w_3 &\leq 14 \\ w_1 &\geq 0 \end{aligned}$$

is convex.

Solution: Pick two arbitrary feasible solutions $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ satisfying both constraints. Then

$$\begin{aligned} 19w_1^{(1)} + 3w_2^{(1)} - w_3^{(1)} &\leq 14 \quad \text{and} \quad w_1^{(1)} \geq 0 \\ 19w_1^{(2)} + 3w_2^{(2)} - w_3^{(2)} &\leq 14 \quad \text{and} \quad w_1^{(2)} \geq 0 \end{aligned}$$

We must show that any point along the line segment joining $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$ also satisfies both constraints. Each such point corresponds to some λ strictly between 0 and 1 in representation [3.29], so pick such a λ .

Multiplying the foregoing main inequality for $\mathbf{w}^{(1)}$ by $(1 - \lambda)$ and that of $\mathbf{w}^{(2)}$ by λ , we may conclude that

$$\begin{aligned} (1 - \lambda)(19w_1^{(1)} + 3w_2^{(1)} - w_3^{(1)}) + \lambda(19w_1^{(2)} + 3w_2^{(2)} - w_3^{(2)}) \\ \leq (1 - \lambda)14 + (\lambda)14 \end{aligned}$$

Noting that the right-hand side is just 14 and regrouping the left, it follows that

$$\begin{aligned} 19[w_1^{(1)} + \lambda(w_1^{(2)} - w_1^{(1)})] + 3[w_2^{(1)} + \lambda(w_2^{(2)} - w_2^{(1)})] \\ - [w_3^{(1)} + \lambda(w_3^{(2)} - w_3^{(1)})] \leq 14 \end{aligned}$$

As required, the point corresponding to λ on the line segment satisfies the constraint.

For the second, nonnegativity constraint, the same computation yields

$$(1 - \lambda)w_1^{(1)} + \lambda(w_1^{(2)}) \geq (1 - \lambda)0 + (\lambda)0$$

or

$$w_1^{(1)} + \lambda(w_1^{(2)} - w_1^{(1)}) \geq 0$$

Thus, the line segment point corresponding to λ also satisfies constraint $w_1 \geq 0$.

Global Optimality of Improving Search for Linear Programs

Recall from definition [2.29] that a **linear program (or LP)** is an optimization model over continuous decision variables with a linear objective function and linear constraints (both main and variable type). Principles [3.31] and [3.32] now combine to establish one reason why this linear objective function over the convex set of points feasible for the linear constraints case is considered among the most tractable of mathematical programs.

Principle 3.33 | If a linear program has a global optimal solution, then improving search Algorithm 3A will stop only upon reaching one. That is, local optima are global optima in linear programs.

Blocking Constraints in Linear Programs

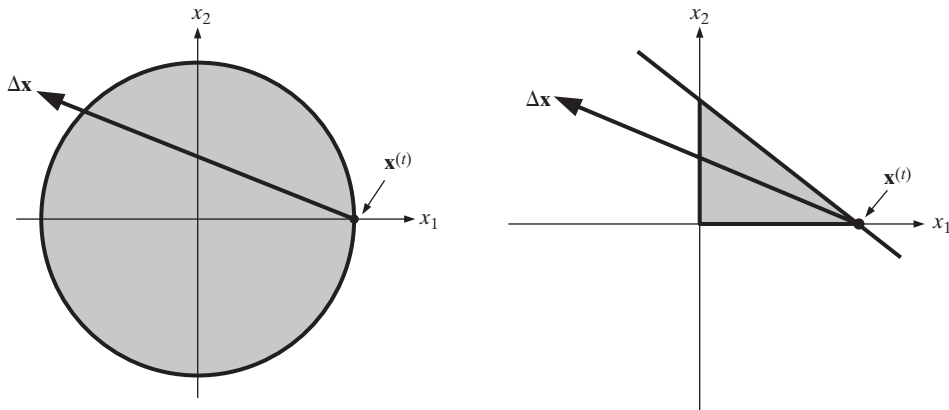
We have already seen with principle [3.26] that an improving direction for a linear objective function remains improving at all feasible points, that is, it cannot be the limiting element in step size Step 3 of Improving Search Algorithm 3A over linear

programs. Likewise, the simplicity of feasible direction conditions [3.25] over active linear constraints assures a conforming direction $\Delta \mathbf{x}$ will never violate an active constraint it was constructed to avoid, no matter what the step size λ . The final element of our investigation of tractability in linear programs is to see that the only feasibility consideration that can limit step size, that is, a **blocking** constraint, must be one that was inactive when $\Delta \mathbf{x}$ was chosen.

EXAMPLE 3.19: VERIFYING THAT BLOCKING CONSTRAINTS FOR LINEARLY CONSTRAINED FEASIBLE SETS MUST BE NONBINDING, BUT NOT ALL CONVEX SETS

Consider the two feasible sets plotted below: the unit circle $\{x_1, x_2: (x_1)^2 + (x_2)^2 \leq 1\}$ of Example 3.17, and a linear program over $\{x_1, x_2 \geq 0: (x_1)^2 + (x_2)^2 \leq 1\}$. In both cases the current solution is $\mathbf{x}^{(t)} = (1, 0)$ and the search has chosen to pursue feasible direction $\Delta \mathbf{x} = (-2, 1)$.

- (a) Identify the active constraints in both cases.
- (b) Show that the blocking constraint for the LP must be one inactive at $\mathbf{x}^{(t)}$, but that this need not be true of the nonlinear case even though the feasible set is convex.



Solution:

(a) In the first, nonlinear case the only constraint $(x_1)^2 + (x_2)^2 \leq 1$ is active. In the linear case active constraints are $x_2 \geq 0$ and $x_1 + x_2 \leq 1$.

(b) After a maximum feasible step in the nonlinear case, the search is blocked by the same constraint active at $\mathbf{x}^{(t)}$. But with linear constraints in the LP, the previously active constraints remains satisfied for any positive step in the chosen direction. Only previously inactive $x_1 \geq 0$ limits feasible progress.

3.5 SEARCHING FOR STARTING FEASIBLE SOLUTIONS

Up to now, all our discussion of improving search has envisioned moving from feasible solution to better feasible solution. But what if a first feasible solution is not readily available? Complex optimization problems, with thousands of constraints and variables, often have no obviously feasible solution. In fact, an early goal of analysis may be to determine whether any feasible solution exists. In this section we introduce the **two-phase** and **big- M** strategies, which deal with this startup issue.

Two-Phase Method

Improving search can be adapted to deal with the absence of a starting feasible solution, simply by using improving search to find one. More precisely, optimization is effected in two phases. **Phase I** addresses an artificial problem with easier-to-satisfy constraints. Starting from some feasible solution for this artificial problem, we minimize how much the artificial solution violates constraints of the true problem. If the violation can be driven to $= 0$, a completely feasible solution is at hand; **Phase II** then performs a usual improving search starting from the Phase I result. If the (globally) optimal value in Phase I is not $\neq 0$, constraint violations cannot be completely eliminated; the true problem is infeasible. Algorithm 3B formalizes this two-phase method.

ALGORITHM 3B: TWO-PHASE IMPROVING SEARCH

Step 0: Artificial Model. Choose any convenient solution for the true model and construct a corresponding Phase I model by adding (or subtracting) non-negative artificial variables in each violated constraint.

Step 1: Phase I. Assign values to artificial variables to complete a starting feasible solution for the artificial model. Then begin at that solution and perform an improving search to minimize the sum of the artificial variables.

Step 2: Infeasibility. If Phase I search terminated with artificial sum $= 0$, proceed Step 3; the original model is feasible. If Phase I search terminated with a global minimum having artificial sum > 0 , stop; the original model is infeasible. Otherwise, repeat Step 1 from a different starting solution.

Step 3: Phase II. Construct a starting feasible solution for the original model by deleting artificial components of the Phase I optimum. Then begin at that solution and perform an improving search to optimize the original objective function subject to original constraints.

Two Crude Model Application Revisited

The Two Crude model of Figure 3.11 provides a familiar example:

$$\begin{array}{ll}
 \min & 100x_1 + 75x_2 \\
 \text{s.t.} & 0.3x_1 + 0.4x_2 \geq 2 \\
 & 0.4x_1 + 0.2x_2 \geq 1.5 \\
 & 0.2x_1 + 0.3x_2 \geq 0.5 \\
 & 0 \leq x_1 \leq 9, 0 \leq x_2 \leq 6
 \end{array}$$

For this tiny case, it is easy to find a starting feasible solution graphically or by trial and error. But for typical instances we employ a more formal procedure.

Algorithm 3B begins by picking arbitrarily some convenient values—feasible or not—for the true decision variables. Often, the choice is to make all decision variables = 0 as we will do here.

Our choice of $x_1 = x_2 = 0$ satisfies the bound constraints

$$0 \leq x_1 \leq 9 \quad \text{and} \quad 0 \leq x_2 \leq 6$$

Still, it violates all 3 main constraints.

Artificial Variables

We deal with constraints unsatisfied at our arbitrary starting solution by introducing artificial variables to absorb the infeasibility.

Principle 3.34 Phase I constraints are derived from those of the original model by considering each in relation to the starting solution chosen. Satisfied constraints simply become part of the Phase I model. Violated ones are augmented with a nonnegative **artificial variable** to permit artificial feasibility.

Including artificial variables x_3 , x_4 , and x_5 in the 3 constraints violated at $x_1 = x_2 = 0$ produces Phase I constraints

$$\begin{aligned} 0.3x_1 + 0.4x_2 + x_3 &\geq 2 \\ 0.4x_1 + 0.2x_2 + x_4 &\geq 1.5 \\ 0.2x_1 + 0.3x_2 + x_5 &\geq 0.5 \\ 0 \leq x_1 \leq 9, 0 \leq x_2 \leq 6 \\ x_3, x_4, x_5 &\geq 0 \end{aligned}$$

Notice that a different artificial variable was introduced in each constraint violated. It was added in each case because extra left-hand side is needed to satisfy the \geq constraints at $x_1 = x_2 = 0$.

Occasionally, it is appropriate to subtract the artificial. For example, if the model had included a constraint

$$x_1 - x_2 = -10$$

the Phase I construction with artificial x_6 is

$$x_1 - x_2 - x_6 = -10$$

Then for $x_1 = x_2 = 0, x_6 = 10$ produces feasibility. Had we used a + sign on x_6 , the corresponding solution would require x_6 negative, a violation of nonnegativity.

Phase I Models

Artificial variables provide a way to get started in Phase I, but our goal is to drive out all the infeasibility, that is, to find a solution feasible with artificial variables all zero. This defines the Phase I objective function.

Principle 3.35 The **Phase I objective function** minimizes the sum of the artificial variables.

Minimizing the total minimizes each artificial because all are required to be nonnegative.

In our Two Crude example, the resulting Phase I model is

$$\begin{aligned}
 \min \quad & x_3 + x_4 + x_5 \\
 \text{s.t.} \quad & 0.3x_1 + 0.4x_2 + x_3 \geq 2 \\
 & 0.4x_1 + 0.2x_2 + x_4 \geq 1.5 \\
 & 0.2x_1 + 0.3x_2 + x_5 \geq 0.5 \\
 & 0 \leq x_1 \leq 9, 0 \leq x_2 \leq 6 \\
 & x_3, x_4, x_5 \geq 0
 \end{aligned} \tag{3.16}$$

EXAMPLE 3.20: CONSTRUCTING A PHASE I MODEL

Consider the optimization model

$$\begin{aligned}
 \max \quad & 14(w_1 - 10)^2 + (w_2 - 3)^2 + (w_3 + 5)^2 \\
 \text{s.t.} \quad & 12w_1 + w_3 \geq 19
 \end{aligned} \tag{3.17}$$

$$4w_1 + w_2 - 7w_3 \leq 10 \tag{3.18}$$

$$-w_1 + w_2 - 6w_3 = -8 \tag{3.19}$$

$$w_1, w_2, w_3 \geq 0 \tag{3.20}$$

Construct an artificial model to begin Phase I improving search with $w_1 = w_2 = w_3 = 0$.

Solution: The solution $w_1 = w_2 = w_3 = 0$ satisfies the main constraint (3.18) and nonnegativity constraints (3.20). In accord with principle [3.39](#), an artificial variable w_4 must be added to deal with violated constraint (3.18), and another w_5 must be subtracted (because of the negative right-hand side -8) to satisfy the violated equality (3.19).

An objective function minimizing the sum of the two artificial variables (principle [3.35](#)) completes the Phase I model.

$$\begin{aligned}
 \min \quad & w_4 + w_5 \\
 \text{s.t.} \quad & 12w_1 + w_3 + w_4 \geq 19 \\
 & 4w_1 + w_2 - 7w_3 \leq 10 \\
 & -w_1 + w_2 - 6w_3 - w_5 = -8 \\
 & w_1, w_2, w_3, w_4, w_5 \geq 0
 \end{aligned}$$

Starting Artificial Solution

Using separate artificials in each violated constraint makes it extremely easy to complete a starting feasible solution for the Phase I search.

Principle 3.36 After fixing original variables at their arbitrarily chosen values, each artificial variable is initialized at the smallest value still needed to achieve feasibility in the corresponding constraint.

For example, in Two Crude model (3.16), we have decided to initiate Phase I search at solution $\mathbf{x}^{(0)}$ with $x_1^{(0)} = x_2^{(0)} = 0$. For these values, the first main constraint becomes

$$0.3(0) + 0.4(0) + x_3 \geq 2$$

It is violated without artificial variable x_3 present, but choosing $x_3^{(0)} = 2$ is just enough to provide feasibility in Phase I. Similarly,

$$0.4(0) + 0.2(0) + x_4 \geq 1.5$$

demands $x_4^{(0)} = 1.5$, and

$$0.2(0) + 0.3(0) + x_5 \geq 0.5$$

implies that $x_5^{(0)} = 0.5$. The result is $\mathbf{x}^{(0)} = (0, 0, 2, 1.5, 0.5)$, a starting feasible solution for the Phase I problem.

EXAMPLE 3.21: CONSTRUCTING A PHASE I STARTING SOLUTION

Return to the artificial problem of Example 3.20. Determine a starting Phase I solution having $w_1 = w_2 = w_3 = 0$.

Solution: To start Phase I, we set $w_1^{(0)} = w_2^{(0)} = w_3^{(0)} = 0$. Then making $w_4^{(0)} = 19$ achieves feasibility in the first main constraint, and $w_5^{(0)} = 8$ satisfies the last. Phase I would start at $\mathbf{w}^{(0)} = (0, 0, 0, 19, 8)$.

Phase I Outcomes

How could the Phase I search end? Certainly it will not terminate with a negative objective function value. Artificial variables are restricted to be nonnegative, so their sum must be nonnegative. For the same reason, the Phase I problem cannot be unbounded. The objective value cannot ever fall below zero.

Three possibilities remain.

Principle 3.37 If Phase I terminates with a solution having (Phase I) objective function value = 0, the components of the Phase I solution corresponding to original variables provide a feasible solution for the original model.

Principle 3.38 If Phase I terminates with a global minimum having (Phase I) objective function value > 0, the original model is infeasible.

Principle 3.39 If Phase I terminates with a local minimum that may not be global but has (Phase I) objective function value > 0 , we can conclude nothing. Phase I search should be repeated from a new starting solution.

Begin with the happiest case [3.37]. Here Phase I has been able to drive the sum of the artificial variables to zero. For example, in the Two Crude Phase I problem above, two iterations of improving search might produce the Phase I solution $\mathbf{x}^{(2)} = (4, 4, 0, 0, 0)$ with objective function value

$$x_3^{(2)} + x_4^{(2)} + x_5^{(2)} = 0 + 0 + 0 = 0$$

The only way nonnegative numbers can sum to zero is for all of them to equal zero. Thus, every artificial is necessarily zero in this final Phase I solution. Since artificials no longer have any effect on their constraints, the values for nonartificial variables must be feasible in the original model. We can simply drop all artificial variables and proceed with Phase II.

The starting Phase II solution will be that part of the Phase I result that involves variables of the real model. For example, our Two Crude Phase I solution $\mathbf{x}^{(2)} = (4, 4, 0, 0, 0)$ has components $x_1^{(2)} = x_2^{(2)} = 4$ on nonartificial variables. Phase II search can start from feasible solution $\mathbf{x}^{(0)} = (4, 4)$.

EXAMPLE 3.22: VERIFYING FEASIBILITY WITH PHASE I

Verify that $w_1 = 2$, $w_2 = 0$, $w_3 = 1$ is a feasible solution to the original model in Example 3.20. Then construct a corresponding optimal solution to exercise's Phase I model, and explain why it is optimal.

Solution: The solution indicated is feasible because all components are nonnegative and

$$\begin{aligned} 12(2) + 1 &= 25 \geq 19 \\ 4(2) + (0) - 7(1) &= 1 \leq 10 \\ -(2) + (0) - 6(1) &= -8 \end{aligned}$$

To construct a corresponding optimum for the Phase I problem, we set artificials $w_4 = w_5 = 0$. Full Phase I solution $\mathbf{w} = (2, 0, 1, 0, 0)$ is feasible because artificials are unneeded to satisfy constraints with $w_1 = 2$, $w_2 = 0$, and $w_3 = 1$. It is Phase I optimal because the sum of nonnegative quantities is minimum when all are zero.

Concluding Infeasibility from Phase I

Now consider the [3.38] and [3.39] cases, where Phase I improving search terminates with a positive objective function value. The final solution from Phase I improving search will probably approximate a local minimum (principle [3.17]), but it may or may not be global.

If we have some way of being sure that the Phase I solution is a global optimum, our conclusion is clear. The original model is infeasible (principle [3.38]),

because every solution to the Phase I model has some artificial variables at positive values. Their sum just cannot be driven to zero.

To illustrate, let us modify our Two Crude model until it is infeasible. Specifically, reverse the direction of the last main inequality so that it reads

$$0.2x_1 + 0.3x_2 + x_5 \leq 0.5 \quad (3.21)$$

in the Phase I problem.

Improving search on this revised Phase I problem will terminate at some solution $\mathbf{x}^{(t)} = (2.5, 0, 1.25, 0.5, 0)$ with objective function value $1.25 + 0.5 + 0 = 1.75 > 0$. Since this Phase I problem has all linear constraints and a linear objective function, we know from Section 3.4 (principle 3.32) that every Phase I local minimum is a global minimum. It follows that no feasible solution to our modified Phase I example can have artificial variable total less than $\mathbf{x}^{(t)}$'s 1.75. That is, artificials simply cannot all be driven out of the solution. We conclude (principle 3.38) that the model with last constraint reversed as in (3.21) is infeasible.

Notice how this analysis depended on our certainty that a local minimum in Phase I was a global minimum. When Phase I's solution may just be a local minimum with positive objective function value, there might still be a solution with artificials all zero. Like any improving search with local outcomes possible, we would have no choice but to repeat the analysis from a different Phase I starting solution (principle 3.39).

EXAMPLE 3.23: PROCESSING PHASE I OUTCOMES

Suppose that a linearly constrained optimization model over variables z_1, z_2 , and z_3 is converted for Phase I by adding nonnegative artificial variables z_4, z_5 , and z_6 . For each of the following original model objective functions and Phase I search results \mathbf{z} , indicate what we can conclude and how Algorithm 3B processing should proceed.

(a) Original model objective: maximize $14z_1 - z_3$; Phase I local optimum: $\mathbf{z} = (1, -1, 3, 0, 0, 0)$

(b) Original model objective: minimize $(z_1 z_2)^2 + \sin(z_3)$; Phase I local optimum: $\mathbf{z} = (1, 2, -3, 0, 1, 1)$

Solution:

(a) All 3 artificial variables = 0 in the Phase I local maximum. Thus (principle 3.37) $\mathbf{z}^{(0)} = (1, -1, 3)$ provides a feasible solution for the original model. Proceed to Phase II search, beginning at this $\mathbf{z}^{(0)}$.

(b) The corresponding Phase I model has a linear objective function and linear constraints. Thus \mathbf{z} is a global optimum (principle 3.37) despite the highly non-unimodal nature of the true objective function. Applying 3.38, we may conclude that the original model is infeasible because artificial sum $z_4 + z_5 + z_6 = 0 + 1 + 1 = 2 > 0$ in a Phase I global optimum.

Big-M Method

Two-Phase Algorithm 3B deals with feasibility and optimality separately. Phase I search tests feasibility. Phase II proceeds to an optimum.

The Big-M method combines these activities in a single search. Artificial variables are included in constraints exactly as in Phase I of Two-Phase search. However, the effort to drive their total to $=0$ is combined with a search for an optimal solution to the original model.

The key to combining feasibility and optimality considerations is a composite objective function.

Definition 3.40 The **Big-M method** uses a large positive multiplier M to combine feasibility and optimality in a single-objective function of the form

$$\max (\text{original objective}) - M(\text{artificial variable sum})$$

for an originally maximize problem, or

$$\min (\text{original objective}) + M(\text{artificial variable sum})$$

for a minimize problem.

Infeasibility reflected in positive values of artificial variables is forced toward zero by adding or subtracting that infeasibility in the objective with the large penalty multiplier M that gives the method its name.

We again illustrate with the Two Crude example. The big- M form of the problem is

$$\begin{aligned} \min \quad & 100x_1 + 75x_2 + M(x_3 + x_4 + x_5) \\ \text{s.t.} \quad & 0.3x_1 + 0.4x_2 + x_3 \geq 2 \\ & 0.4x_1 + 0.2x_2 + x_4 \geq 1.5 \\ & 0.2x_1 + 0.3x_2 + x_5 \geq 0.5 \\ & 0 \leq x_1 \leq 9, 0 \leq x_2 \leq 6 \\ & x_3, x_4, x_5 \geq 0 \end{aligned} \tag{3.22}$$

Notice that the constraints are identical to those of Phase I model (3.16). That means that the $\mathbf{x}^{(0)} = (0, 0, 2, 1.5, 0.5)$ of principle [3.36](#) continues to provide a suitable starting solution.

What is new is the objective function. It combines the original objective to minimize $100x_1 + 75x_2$ with a large positive multiple of artificial variable sum $(x_3 + x_4 + x_5)$.

Pick, say, $M = 10,000$. Then a single search of big- M version (3.22) will lead to an optimal solution of the original model. Any solution with positive artificial variable total will be highly penalized and thus not optimal in (3.22). Among those that have artificial total $= 0$, (i.e., among the feasible solutions of the original model), one with lowest original objective value will be preferred.

EXAMPLE 3.24: CONSTRUCTING A BIG- M MODEL

Return to the optimization of Example 3.20 and construct the corresponding Big- M model.

Solution: Constraints and artificial variables are exactly as in Example 3.21. Including the Big- M objective of definition [3.40](#) results in the following big- M model:

$$\begin{aligned} \max \quad & 14(w_1 - 10)^2 + (w_2 - 3)^2 + (w_3 + 5)^2 - M(w_4 + w_5) \\ \text{s.t.} \quad & 12w_1 + w_3 + w_4 \geq 19 \\ & 4w_1 + w_2 - 7w_3 \leq 10 \\ & -w_1 + w_2 - 6w_3 - w_5 = -8 \\ & w_1, w_2, w_3, w_4, w_5 \geq 0 \end{aligned}$$

The artificial sum is subtracted because the objective is maximized.

Big- M Outcomes

Possible outcomes from a Big- M search closely parallel those of the two-phase method detailed in principles [3.37](#) to [3.39](#). When Big- M search terminates, we may have an optimum for the original model, a proof that the original model was infeasible, or just confusion.

Consider first the optimal case.

Principle 3.41 | If a Big- M search terminates with a locally optimal solution having all artificial variables = 0, the components of the solution corresponding to original variables form a locally optimal solution for the original model.

For example, with $M = 10,000$, infeasibility is so expensive that big- M search of model (3.22) will compute global optimum $\mathbf{x} = (2, 3.5, 0, 0, 0)$. Components $x_1 = 2$ and $x_2 = 3.5$, which were the original decision variables, constitute an optimal solution to model of interest.

When Big- M search terminates with artificial variables at positive values, matters are more murky.

Principle 3.42 | If M is sufficiently large and Big- M search terminates with a global optimum having some artificial variables > 0 , the original model is infeasible.

Principle 3.43 | If Big- M search terminates with a local optimum having some artificial variables > 0 , or the multiplier M may not be large enough, we can conclude nothing. The search should be repeated with a larger M and/or a new starting solution.

As with two-phase outcomes [3.38] and [3.39], we will not be able to reach any conclusions when big- M search stops with a positive artificial variable total unless we can be sure that the search has produced a global optimum. If the outcome is known only to be a local optimum, a feasible solution might yet be found.

However, notice in [3.42] and [3.43] that we encounter a new difficulty when we adopt the big- M approach. If artificial variables take nonzero values in the big- M optimum, it may only mean that we did not choose a large enough multiplier M . For example, solution of big- M form (3.22) with $M = 1$ will yield optimal solution $\mathbf{x} = (0, 0, 2.0, 1.5, 0.5)$ because the penalty of

$$M(x_3 + x_4 + x_5) = 1(2.0 + 1.5 + 0.5)$$

is insufficient to force out infeasibility. Only when M is sufficiently large—something that depends on details of each model—does a big- M optimum with positive artificial variables imply infeasibility.

EXAMPLE 3.25: PROCESSING BIG- M OUTCOMES

Suppose that a linearly constrained optimization model over variables z_1 , z_2 , and z_3 is converted for big- M by including nonnegative artificial variables z_4 , z_5 , and z_6 and penalizing the original objective function (as in [3.40]) by $M = 1000$ times their sum. For each of the following original model objective functions and big- M search results \mathbf{z} , indicate what we can conclude.

- (a) Original model objective: maximize $14z_1 - z_3$; big- M local optimum: $\mathbf{z} = (1, -1, 3, 0, 0, 0)$
- (b) Original model objective: maximize $z_2 + z_3$; big- M local optimum: $\mathbf{z} = (0, 0, 0, 1, 0, 2)$
- (c) Original model objective: minimize $(z_1 z_2)^2 + \sin(z_3)$; big- M local optimum: $\mathbf{z} = (1, 1, 3, 0, 0, 0)$
- (d) Original model objective: minimize $(z_1 z_2)^2 + \sin(z_3)$; big- M local optimum: $\mathbf{z} = (1, 2, -3, 0, 1, 1)$

Solution:

(a) All three artificial variables = 0 in the big- M local optimum. Thus (principle [3.41]) $\mathbf{z} = (1, -1, 3)$ provides a local maximum for the original model. With both objective and constraints linear, it is also globally optimal.

(b) With both objective and constraints linear, solution \mathbf{z} is a global optimum in the big- M model. However, we can conclude that the original model is infeasible (principle [3.42]) only if we know that $M = 1000$ imposes a large enough penalty to make any infeasible solution have a lower objective value than any feasible one.

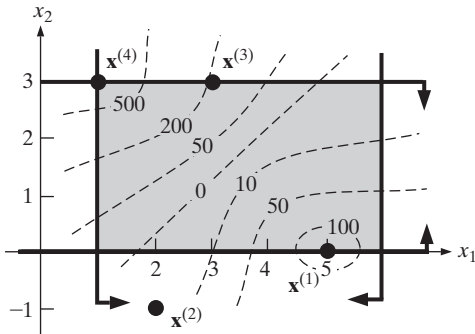
(c) All three artificial variables = 0 in the big- M local optimum. Thus (principle [3.41]) $\mathbf{z} = (1, 1, 3)$ provides a local optimum for the original model. The highly non-unimodal nature of the objective makes it impossible to tell whether it is a global optimum.

(d) The highly non-unimodal nature of the big- M objective function makes it impossible to know whether the indicated \mathbf{z} is a global minimum. Thus since some artificial variables have positive value, our only choices (principle 3.43) are to increase M and/or repeat the search from a different starting point.

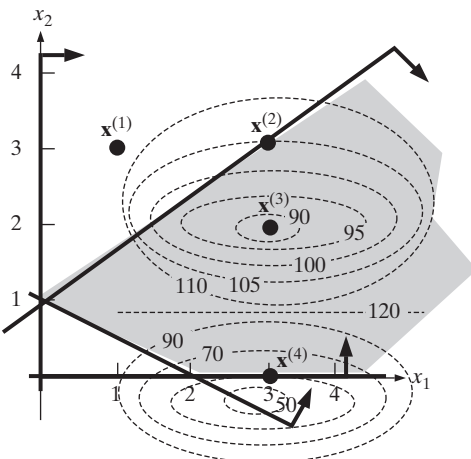
EXERCISES

3-1 In each of the following plots, determine whether the specified points are feasible, infeasible, local optimal, and/or global optimal in the depicted mathematical program over two continuous variables. Dashed lines indicate contours of the objective function, and solid lines show constraints.

- ✔ (a) Maximize problem; $\mathbf{x}^{(1)} = (5, 0)$, $\mathbf{x}^{(2)} = (2, -1)$, $\mathbf{x}^{(3)} = (3, 3)$, $\mathbf{x}^{(4)} = (1, 3)$



- (b) Minimize problem; $\mathbf{x}^{(1)} = (1, 3)$, $\mathbf{x}^{(2)} = (3, 3)$, $\mathbf{x}^{(3)} = (3, 2)$, $\mathbf{x}^{(4)} = (3, 0)$



3-2 Each of the following shows the sequence of directions and steps employed by an improving search that began at $\mathbf{y}^{(0)} = (2, 0, 5)$. Compute the sequence of points visited by the search.

- ✔ (a) $\Delta \mathbf{y}^{(1)} = (3, -1, 0)$, $\lambda_1 = 2$, $\Delta \mathbf{y}^{(2)} = (-1, 2, 1)$, $\lambda_2 = 5$, $\Delta \mathbf{y}^{(3)} = (0, 6, 0)$, $\lambda_3 = \frac{1}{2}$
- (b) $\Delta \mathbf{y}^{(1)} = (1, 3, -2)$, $\lambda_1 = 2$, $\Delta \mathbf{y}^{(2)} = (1, 0, 2)$, $\lambda_2 = \frac{1}{2}$, $\Delta \mathbf{y}^{(3)} = (4, 3, 2)$, $\lambda_3 = 12$

3-3 Each of the following shows the sequence of points visited by an improving search. Compute the corresponding sequence of move directions assuming that all step sizes $\lambda = 1$.

- ✔ (a) $\mathbf{w}^{(0)} = (0, 1, 1)$, $\mathbf{w}^{(1)} = (4, -1, 7)$, $\mathbf{w}^{(2)} = (4, -3, 19)$, $\mathbf{w}^{(3)} = (3, -3, 22)$
- (b) $\mathbf{w}^{(0)} = (4, 0, 7)$, $\mathbf{w}^{(1)} = (4, 2, 10)$, $\mathbf{w}^{(2)} = (-2, 4, 5)$, $\mathbf{w}^{(3)} = (5, 5, 5)$

3-4 Refer to the plots of Exercise 3-1 and determine graphically whether the following directions appear to be improving at the points indicated.

- ✔ (a) $\Delta \mathbf{x} = (-3, 3)$ at $\mathbf{x}^{(1)}$ of 3-1(a)
- (b) $\Delta \mathbf{x} = (0, 1)$ at $\mathbf{x}^{(2)}$ of 3-1(a)
- ✔ (c) $\Delta \mathbf{x} = (-10, 1)$ at $\mathbf{x}^{(3)}$ of 3-1(a)
- (d) $\Delta \mathbf{x} = (0, -3)$ at $\mathbf{x}^{(2)}$ in 3-2(b)
- (e) $\Delta \mathbf{x} = (2, 2)$ at $\mathbf{x}^{(3)}$ in 3-2(b)
- (f) $\Delta \mathbf{x} = (-1, -10)$ at $\mathbf{x}^{(4)}$ in 3-2(b)

3-5 Refer to the plots of Exercise 3-2 and determine graphically whether the following directions appear to be feasible at the points indicated.

- ✔ (a) $\Delta \mathbf{x} = (-5, 5)$ at $\mathbf{x}^{(1)}$ of 3-2(a)
- (b) $\Delta \mathbf{x} = (0, 1)$ at $\mathbf{x}^{(3)}$ of 3-2(a)
- ✔ (c) $\Delta \mathbf{x} = (-10, 0)$ at $\mathbf{x}^{(3)}$ of 3-2(a)
- (d) $\Delta \mathbf{x} = (-3, -2)$ at $\mathbf{x}^{(2)}$ in 3-2(b)

- ✓ (e) $\Delta \mathbf{x} = (2, 5)$ at $\mathbf{x}^{(3)}$ in 3-2(b)
- (f) $\Delta \mathbf{x} = (-1, -10)$, at $\mathbf{x}^{(4)}$ in 3-2(b)

3-6 Consider a mathematical program with constraints

$$x_1 - 2x_2 + 3x_3 \leq 25$$

$$x_1, x_2, x_3 \geq 0$$

Determine the maximum step (possibly $+\infty$) that preserves feasibility in the direction indicated from the point specified. Also indicate whether that step indicates that the model is unbounded, assuming that directions improve everywhere.

- ✓ (a) $\Delta \mathbf{x} = (-1, 3, -2)$ from $\mathbf{x} = (4, 0, 6)$
- (b) $\Delta \mathbf{x} = (-2, 1, -1)$ at $\mathbf{x} = (8, 5, 3)$
- ✓ (c) $\Delta \mathbf{x} = (1, 3, 1)$ from $\mathbf{x} = (0, 0, 4)$
- (d) $\Delta \mathbf{x} = (2, 7, 4)$ at $\mathbf{x} = (20, 4, 3)$

3-7 For each of the following combinations of objective function, point, and direction, determine whether conditions 3.21 and 3.22 show that the direction improves at the point, does not improve at the point, or that further information is required.

- ✓ (a) $\max 4y_1 - 2y_3 + y_5, \mathbf{y} = (1, 0, 19, 4, 6), \Delta \mathbf{y} = (2, -3, 4, 0, 6)$
- (b) $\max y_1 + 7y_3 + 2y_5, \mathbf{y} = (1, 0, 9, 0, 0), \Delta \mathbf{y} = (-10, -20, 2, 0, 4)$
- ✓ (c) $\min y_1y_2 + (y_1)^2 + 4y_2, \mathbf{y} = (3, -1), \Delta \mathbf{y} = (-7, 5)$
- (d) $\min y_1y_2 + 4y_1 + (y_2)^2, \mathbf{y} = (2, 1), \Delta \mathbf{y} = (-1, 3)$
- ✓ (e) $\max (y_1 - 5)^2 + (y_2 + 1)^2, \mathbf{y} = (4, 1), \Delta \mathbf{y} = (-1, 2)$
- (f) $\min (y_1 - 2)^2 + y_1y_2 + (y_2 - 3)^2, \mathbf{y} = (1, 1), \Delta \mathbf{y} = (3, -1)$

3-8 Construct an improving direction from the gradient of each objective function at the point indicated.

- ✓ (a) $\max 3w_1 - 2w_2 + w_4$ at $\mathbf{w} = (2, 0, 5, 1)$
- (b) $\min -4w_2 + 5w_3 - w_4$, at $\mathbf{w} = (2, 2, 1, 0)$
- ✓ (c) $\min (w_1 + 2)^2 - w_1w_2$ at $\mathbf{w} = (3, 2)$
- (d) $\max -4w_1 + 9w_2 + 2(w_2)^2$ at $\mathbf{w} = (11, 2)$

3-9 Determine which of the constraints

$$(z_1 - 2)^2 + (z_2 - 1)^2 \leq 25 \quad [\text{i}]$$

$$2z_1 - z_2 = 8 \quad [\text{ii}]$$

$$z_1 \geq 0 \quad [\text{iii}]$$

$$z_2 \geq 0 \quad [\text{iv}]$$

are active at each of the following solutions.

- ✓ (a) $\mathbf{z} = (4, 0)$
- (b) $\mathbf{z} = (6, 4)$

3-10 Determine whether each of the directions specified is feasible at the solution indicated to the linear constraints

$$3y_1 - 2y_2 + 8y_3 = 14$$

$$6y_1 - 4y_2 - 1y_3 \leq 11$$

$$y_1, y_2, y_3 \geq 0$$

- ✓ (a) $\Delta \mathbf{y} = (0, 4, 1)$ at $\mathbf{y} = (2, 0, 1)$
- (b) $\Delta \mathbf{y} = (0, -4, 1)$ at $\mathbf{y} = (2, 0, 1)$
- ✓ (c) $\Delta \mathbf{y} = (2, 0, 1)$ at $\mathbf{y} = (0, 1, 2)$
- (d) $\Delta \mathbf{y} = (-2, 1, 1)$ at $\mathbf{y} = (0, 1, 2)$

3-11 State all conditions that must be satisfied by a feasible direction $\Delta \mathbf{w}$ at the solution indicated to each of the following systems of linear constraints.

- ✓ (a) $2w_1 + 3w_3 = 18$
 $1w_1 + 1w_2 + 2w_3 = 14$
 $w_1, w_2, w_3 \geq 0$
 at $\mathbf{w} = (0, 2, 6)$
- (b) Same constraints as part (a) at $\mathbf{w} = (6, 4, 2)$.
- ✓ (c) $1w_1 + 1w_2 = 10$
 $2w_1 - 1w_2 \geq 8$
 $1w_1 - 8w_2 \leq 1$
 at $\mathbf{w} = (6, 4)$
- (d) Same constraints as part (c) at $\mathbf{w} = (7, 3)$.

3-12 Consider the linear program

$$\min -y_1 + 5y_2$$

$$\text{s.t. } -y_1 + y_2 \leq 3$$

$$y_2 \geq 2$$

$$y_2 \geq y_1$$

$$y_1, y_2 \geq 0$$

at current solution $\mathbf{y}^{(1)} = (0, 3)$.

- ✔ (a) List the condition for a direction $\Delta \mathbf{y}$ to be improving at $\mathbf{y}^{(1)}$.
- ✔ (b) Show that direction $\Delta \mathbf{y} = (1, -1)$ satisfies your condition of part (a).
- ✔ (c) Determine which constraints are active at $\mathbf{y}^{(1)}$.
- ✔ (d) List and justify all conditions for any direction $\Delta \mathbf{y}$ to be feasible at point $\mathbf{y}^{(1)}$.
- ✔ (e) Show that direction $\Delta \mathbf{y} = (1, -1)$ satisfies your conditions of part (d), determine the maximum feasible step λ in that direction from $\mathbf{y}^{(1)}$, and compute the next solution point $\mathbf{y}^{(2)}$.
- (f) Draw a 2-dimensional plot of the feasible space for this LP including contours of its objective. Then show how $\Delta \mathbf{y} = (1, -1)$ improves the objective, identify $\mathbf{y}^{(1)}$, and demonstrate how the same $\Delta \mathbf{y}$ preserves all constraints until it encounters an inactive one at the λ of part (e) to produce $\mathbf{y}^{(2)}$.

3-13 Do Exercise 3-12(a)–(e), on LP

$$\begin{aligned} \min \quad & 3x_1 - 13x_3 \\ \text{s.t.} \quad & 11x_1 + 3x_2 + 4x_3 = 69 \\ & x_1 + x_2 + x_3 \leq 16 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

using $\mathbf{x}^{(1)} = (3, 0, 9)$ and $\Delta \mathbf{x} = (-1, 1, 2)$.

3-14 Consider the mathematical program

$$\begin{aligned} \max \quad & 4z_1 + 7z_2 \\ \text{s.t.} \quad & 2z_1 + z_2 \leq 9 \\ & 0 \leq z_1 \leq 4 \\ & 0 \leq z_2 \leq 3 \end{aligned}$$

- ✔ (a) Show that directions $\Delta \mathbf{z}^{(1)} = (2, 0)$ and $\Delta \mathbf{z}^{(2)} = (-2, 4)$ are improving directions for this model at every \mathbf{z} .
- ✔ (b) Beginning at $\mathbf{z}^{(0)} = (0, 0)$, execute Improving Search Algorithm 3A on the model. Limit your search to the two directions of part (a), and continue until neither is both improving and feasible.
- (c) Show in a two-dimensional plot the feasible space and objective function contours of the model. Then plot the path of your search in part (b).

3-15 Do Exercise 3-14 for mathematical program

$$\begin{aligned} \min \quad & z_1 + z_2 \\ \text{s.t.} \quad & z_1 + 2z_2 \geq 4 \\ & 0 \leq z_1 \leq 6 \\ & 0 \leq z_2 \leq 4 \end{aligned}$$

directions $\Delta \mathbf{z}^{(1)} = (0, -2)$, $\Delta \mathbf{z}^{(2)} = (-4, 2)$, and initial point $\mathbf{z}^{(0)} = (6, 4)$.

3-16 Consider the line segment between each of the following solution pairs $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$. Write an algebraic expression representing all points on the line segment, show that the given $\mathbf{z}^{(3)}$ is on the line segment, and show that the $\mathbf{z}^{(4)}$ specified is not.

- ✔ (a) $\mathbf{z}^{(1)} = (3, 1, 0)$, $\mathbf{z}^{(2)} = (0, 4, 9)$, $\mathbf{z}^{(3)} = (2, 2, 3)$, $\mathbf{z}^{(4)} = (3, 5, 9)$
- (b) $\mathbf{z}^{(1)} = (6, 4, 4)$, $\mathbf{z}^{(2)} = (10, 0, 7)$, $\mathbf{z}^{(3)} = (9, 1, 25/4)$, $\mathbf{z}^{(4)} = (14, -4, 10)$

3-17 Determine whether the feasible set for each of the following systems of constraints is convex, and if not, indicate points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ that violate definition [3.27](#).

- ✔ (a) $(x_1)^2 + (x_2)^2 \geq 9$
 $x_1 + x_2 \leq 10$
 $x_2, x_2 \geq 0$
- (b) $(x_1)^2/4 + (x_2)^2 \leq 25$
 $x_1 \leq 9$
 $x_1 + x_2 \geq 3$
 $x_1, x_2 \geq 0$
- ✔ (c) $x_1 - 2x_2 + x_3 = 2$
 $x_1 + 8x_2 - x_3 \leq 16$
 $x_1 + 4x_2 - x_3 \geq 5$
 $x_1, x_2, x_3 \geq 0$
- (d) $\sum_{j=1}^{12} 3x_j \leq 50$
 $x_j \geq x_{j-1} \quad j = 2, \dots, 12$
 $x_j \geq 0 \quad j = 1, \dots, 12$
- ✔ (e) $x_1 + 2x_2 + 3x_3 + x_4 \leq 24$
 $0 \leq x_j \leq 10, \quad j = 1, \dots, 4$
 x_j integer, $j = 1, \dots, 4$
- (f) $\sum_{j=1}^{50} x_j \leq 200$
 $x_j \leq x_1 \quad j = 2, \dots, 100$
 $x_j \geq 0 \quad j = 1, \dots, 100$
 $x_1 = 0$ or 1

3-18 Construct a Phase I model corresponding to each of the following, and indicate appropriate starting values for the artificial variables. Assume that all original decision variables start at $w_j = 0$.

$$\begin{aligned} \checkmark \text{ (a) } \max & \quad 22w_1 - w_2 + 15w_3 \\ \text{s.t.} & \quad 40w_1 + 30w_2 + 10w_3 = 150 \\ & \quad w_1 - w_2 \leq 0 \\ & \quad 4w_2 + w_3 \geq 0 \\ & \quad w_1, w_2, w_3 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(b) } \min & \quad -w_1 + 5w_2 \\ \text{s.t.} & \quad -w_1 + w_2 \leq 3 \\ & \quad w_2 \geq w_1 + 1 \\ & \quad w_2 \geq w_1 \\ & \quad w_1, w_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \checkmark \text{ (c) } \min & \quad 2w_1 + 3w_2 \\ \text{s.t.} & \quad (w_1 - 3)^2 + (w_2 - 3)^2 \leq 4 \\ & \quad 2w_1 + 2w_2 = 5 \\ & \quad w_1 \geq 3 \end{aligned}$$

$$\begin{aligned} \text{(d) } \max & \quad (w_1)^2 + (w_2)^2 \\ \text{s.t.} & \quad w_1w_2 \leq 9 \\ & \quad 2w_1 = 4w_2 \\ & \quad w_2 \geq 2 \\ & \quad w_1 \geq 0 \end{aligned}$$

3-19 Consider the linear program

$$\begin{aligned} \min & \quad 3w_1 + 7w_2 \\ \text{s.t.} & \quad w_1 + w_2 \geq 5 \\ & \quad 0 \leq w_1 \leq 2 \\ & \quad 0 \leq w_2 \leq 2 \end{aligned}$$

- Justify by inspection that this model must be infeasible.
- Add artificial variables to construct a Phase I version for which improving search could start with $w_1 = w_2 = 0$.
- Explain why your Phase I model must be feasible even though the original LP was not.
- Solve your Phase I model with class optimization software to prove the original model is infeasible.

3-20 Describe how a two-phase improving search of a model with original variables y_1 , y_2 , and y_3 would proceed if Phase I search terminated as follows:

- Global optimum $\mathbf{y} = (40, 7, 0, 9, 0)$
- Global optimum $\mathbf{y} = (6, 3, 1, 0, 0)$
- Local optimum $\mathbf{y} = (1, 3, 1, 0, 0)$
- Local optimum $\mathbf{y} = (0, 5, 1, 2, 0)$, which may not be a global optimum

3-21 Construct a Big- M starting model for each case in Exercise 3-18 and indicate appropriate starting values for the artificial variables. Assume that all original decision variables start at $w_j = 0$.

3-22 Describe how Big- M search of a model with original variables y_1 , y_2 , and y_3 would proceed if improving search of the Big- M version produced each of the outcomes in Exercise 3-20.

REFERENCES

Bazaraa, Mokhtar, Hanif D. Sherali and C. M. Shetty (2006), *Nonlinear Programming - Theory and Algorithms*, Wiley Interscience, Hoboken, New Jersey.

Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009). *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.

Luenberger, David G. and Yinyu Ye (2008), *Linear and Nonlinear Programming*, Springer, New York, New York.

This page intentionally left blank

Linear Programming Models

Linear programs (or LPs) are mathematical programs combining all the characteristics that Chapter 3 showed lead to high tractability.

Definition 4.1 | An optimization model is a **linear program** (or **LP**) if it has continuous variables, a single linear objective function, and all constraints are linear equalities or inequalities.

That is, the model's single-objective function and all constraints must consist of weighted sums of continuous decision variables.

Everything in life is not linear and continuous, but an enormous variety of applications can be validly modeled as LPs. Cases with thousands, millions, or even billions of variables can then be solved to global optimality.

We have already encountered linear programming models in Sections 2.1 to 2.4. Many others are found in network flow Chapter 10.

In this chapter, we develop a more comprehensive picture by presenting several of the classic LP application forms. Most of the models described have been actually used by real organizations. Only the details and numerical data are fictitious.

Linear programs are so central to the study of mathematical programming that four subsequent chapters also treat LP topics. In Chapter 5 we specialize the improving search notions of Chapter 3 to obtain a powerful algorithm called the **simplex** method; in Chapter 6, we add some duality-base variants of simplex; in Chapter 7, we present interior-point varieties of improving search for LPs; also in Chapter 6, we develop the extensive duality and sensitivity analysis available in linear programming and in Chapter 10, we address the special linear programs that model network flows.

4.1 ALLOCATION MODELS

One of the simplest forms of linear programs that occurs widely in application might be termed **allocation models**. The main issue is how to divide or allocate a valuable resource among competing needs. The resource may be land, capital, time, fuel, or anything else of limited availability. For example, the (nonlinear) E-mart model (2.13) of Section 2.4 allocated an advertising budget.

APPLICATION 4.1: FOREST SERVICE ALLOCATION

The U.S. Forest Service has used just such an allocation model to address the sensitive task of managing 191 million acres of national forestland.¹ The Forest Service must trade off timber, grazing, recreational, environmental, national preservation, and other demands on forestland.

Models of a forest begin by dividing land into homogeneous *analysis areas*. Several *prescriptions* or land management policies are then proposed and evaluated for each. The optimization seeks the best possible allocation of land in the analysis areas to particular prescriptions, subject to forest-wide restrictions on land use.

Table 4.1 provides details of the fictional, 788 thousand acre Wagonho National Forest that we model. Wagonho is assumed to have 7 analysis areas, each subject to 3 different prescriptions. The first prescription encourages timbering, the second emphasizes grazing, and the third preserves the land as wilderness. Using index dimensions

$i \triangleq$ analysis area number ($i = 1, \dots, 7$)

$j \triangleq$ prescription number ($j = 1, \dots, 3$)

Table 4.1 provides values for all the following symbolic parameters:

$s_i \triangleq$ size of analysis area i (in thousands of acres)

$p_{i,j} \triangleq$ net present value (NPV) per acre of all uses in area i if managed under prescription j

$t_{i,j} \triangleq$ projected timber yield (in board feet per acre) of analysis area i if managed under prescription j

$g_{i,j} \triangleq$ projected grazing capability (in animal unit months per acre) of analysis area i if managed under prescription j

$w_{i,j} \triangleq$ wilderness index rating (0 to 100) of analysis area i if managed under prescription j

We wish to find an allocation that maximizes net present value while producing 40 million board feet of timber, 5 thousand animal unit months of grazing, and keeping average wilderness index to at least 70.

¹Based on B. Kent, B. B. Bare, R. C. Field, and G. A. Bradley (1991), "Natural Resource Land Management Planning Using Large-Scale Linear Programs: The USDA Forest Service Experience with FORPLAN," *Operations Research*, 39, 13–27.

TABLE 4.1 Forest Service Application Data

Analysis Area, i	Acres, s_i (000)'s	Prescription, j	NPV, (per acre) $p_{i,j}$	Timber, (per acre) $t_{i,j}$	Grazing, (per acre) $g_{i,j}$	Wilderness Index, $w_{i,j}$
1	75	1	503	310	0.01	40
		2	140	50	0.04	80
		3	203	0	0	95
2	90	1	675	198	0.03	55
		2	100	46	0.06	60
		3	45	0	0	65
3	140	1	630	210	0.04	45
		2	105	57	0.07	55
		3	40	0	0	60
4	60	1	330	112	0.01	30
		2	40	30	0.02	35
		3	295	0	0	90
5	212	1	105	40	0.05	60
		2	460	32	0.08	60
		3	120	0	0	70
6	98	1	490	105	0.02	35
		2	55	25	0.03	50
		3	180	0	0	75
7	113	1	705	213	0.02	40
		2	60	40	0.04	45
		3	400	0	0	95

Allocation Decision Variables

As in all such models, our Forest Service example seeks an optimal allocation of a valuable resource. Corresponding decision variables define the allocation.

Principle 4.2 | Principal decision variables in allocation models specify how much of the critical resource is allocated to each use.

Our Forest Service case will employ nonnegative

$x_{i,j} \triangleq$ number of thousands of acres in analysis area i managed by prescription j

Forest Service Allocation Model

The Forest Service's objective is to maximize total net present value (NPV). In terms of the defined notation, this is

$$\begin{aligned} \max \sum_{i=1}^7 \sum_{j=1}^3 p_{i,j} x_{i,j} &= 503x_{1,1} + 140x_{1,2} + 203x_{1,3} + 675x_{2,1} + 100x_{2,2} + 45x_{2,3} \\ &+ \cdots + 705x_{7,1} + 60x_{7,2} + 400x_{7,3} \end{aligned}$$

One system of constraints must assure that all acres of each analysis area are allocated. For example, in analysis area 1,

$$x_{1,1} + x_{1,2} + x_{1,3} = 75$$

Using symbolic constants, all 7 such constraints can be expressed by the system

$$\sum_{j=1}^3 x_{i,j} = s_i \quad i = 1, \dots, 7$$

Finally, there are the performance requirements on timber, grazing, and wilderness index. For example, we want timber output

$$\begin{aligned} \sum_{i=1}^7 \sum_{j=1}^3 t_{i,j} x_{i,j} &= 310x_{1,1} + 50x_{1,2} + 0x_{1,3} + 198x_{2,1} + 46x_{2,2} + 0x_{2,3} \\ &+ \dots + 213x_{7,1} + 40x_{7,2} + 0x_{7,3} \\ &\geq 40,000 \end{aligned}$$

Combining produces the following allocation linear program:

$$\begin{aligned} \max \quad & \sum_{i=1}^7 \sum_{j=1}^3 p_{i,j} x_{i,j} && \text{(present value)} \\ \text{s.t.} \quad & \sum_{j=1}^3 x_{i,j} = s_i \quad i = 1, \dots, 7 && \text{(allocation)} \\ & \sum_{i=1}^7 \sum_{j=1}^3 t_{i,j} x_{i,j} \geq 40,000 && \text{(timber)} \\ & \sum_{i=1}^7 \sum_{j=1}^3 g_{i,j} x_{i,j} \geq 5 && \text{(grazing)} \\ & \frac{1}{788} \sum_{i=1}^7 \sum_{j=1}^3 w_{i,j} x_{i,j} \geq 70 && \text{(wilderness)} \\ & x_{i,j} \geq 0 \quad i = 1, \dots, 7; \quad j = 1, \dots, 3 \end{aligned} \tag{4.1}$$

An optimal allocation makes

$$\begin{aligned} x_{1,1}^* &= 0, & x_{1,2}^* &= 0, & x_{1,3}^* &= 75, & x_{2,1}^* &= 90, & x_{2,2}^* &= 0, & x_{2,3}^* &= 0, \\ x_{3,1}^* &= 140, & x_{3,2}^* &= 0, & x_{3,3}^* &= 0, & x_{4,1}^* &= 0, & x_{4,2}^* &= 0, & x_{4,3}^* &= 60, \\ x_{5,1}^* &= 0, & x_{5,2}^* &= 154, & x_{5,3}^* &= 58, & x_{6,1}^* &= 0, & x_{6,2}^* &= 0, & x_{6,3}^* &= 98, \\ x_{7,1}^* &= 0, & x_{7,2}^* &= 0, & x_{7,3}^* &= 113 \end{aligned}$$

with total net present value \$322,515,000.

EXAMPLE 4.1: FORMULATING ALLOCATION LPs

Jill College is taking courses in operations research, engineering economics, statistics, and material science. She has 30 study hours to prepare for her finals and wishes to divide her time to improve her term grades as much as possible. Naturally, her

favorite course is operations research, so she will spend as much on it as any other. Still, she believes up to 10 hours of study could be useful in any of the courses, with each hour on operations research increasing her grade by 2%, each on engineering economics yielding 3%, each on statistics producing 1%, and each on materials science adding 5%. Form an allocation linear program to help Jill optimize her study.

Solution: Using decision variables

$$h_j \triangleq \text{hours spent on the } j\text{th course}$$

the required model is

$$\begin{array}{ll} \max & 2h_1 + 3h_2 + 1h_3 + 5h_4 & \text{(total gain)} \\ \text{s.t.} & h_1 + h_2 + h_3 + h_4 = 30 & \text{(allocation)} \\ & h_1 \geq h_j & j = 2, \dots, 4 \quad \text{(OR most)} \\ & h_j \leq 10 & j = 1, \dots, 4 \quad \text{(maximum 10)} \\ & h_j \leq 0 & j = 1, \dots, 4 \end{array}$$

The objective maximizes score gain. Main constraints make the allocation total 30, keep OR study as great as any other, and limit the allocation for any class to 10 hours.

4.2 BLENDING MODELS

Allocation models split a resource. Blending models combine them. That is, **blending models** decide what mix of ingredients best fulfills specified output requirements. Various applications blend everything from chemicals, to diets, to metals, to animal food. For example, the Two Crude case of Section 2.1 is a blending model mixing crude petroleum to produce refinery products.

APPLICATION 4.2: SWEDISH STEEL

The steel industry confronts another blending problem when it melts materials in high-temperature furnaces to manufacture new alloys from scrap. Fagersta AB of Fagersta, Sweden, is one of many companies that have used mathematical programming to plan this steel blending.²

An optimization arises each time a furnace is *charged*. Scrap in the available inventory is combined with pure additives to produce a blend having the required percentages of various chemical elements. It is critical to make maximum use of scrap because additives are much more expensive. Although there are some integer programming aspects discussed in Section 11.1, we deal here only with the simpler linear programming form.

Our fictitious version of Swedish steelmaking will produce a 1000-kilogram furnace charge. All steel consists primarily of iron. Table 4.2 shows the much smaller

²Based on C.-H. Westerberg, B. Bjorklund, and E. Hultman (1977), "An Application of Mixed Integer Programming in a Swedish Steel Mill," *Interfaces*, 7:2, 39–43.

TABLE 4.2 Data for Swedish Steel Example

	Composition (%)				Available (kg)	Cost (kr/ kg)
	Carbon	Nickel	Chromium	Molybdenum		
First scrap	0.80	18	12	—	75	16
Second scrap	0.70	3.2	1.1	0.1	250	10
Third scrap	0.85	—	—	—	Unlimited	8
Fourth scrap	0.40	—	—	—	Unlimited	9
Nickel	—	100	—	—	Unlimited	48
Chromium	—	—	100	100	Unlimited	60
Molybdenum	—	—	—	—	Unlimited	53
Minimum blend	0.65	3.0	1.0	1.1		
Maximum blend	0.75	3.5	1.2	1.3		

fractions of carbon, nickel, chromium, and molybdenum in the four available supplies of scrap, on which we can draw, along with the quantities held and their unit cost in Swedish krona. It also shows the three higher-cost additives that can be used and the acceptable ranges for the resulting blend. For example, the 1000 kilograms of steel produced should contain between 0.65 and 0.75% carbon.

Ingredient Decision Variables

It is characteristic that we make **ingredient** decisions.

Principle 4.3 Principal decision variables in blending models specify how much of each available ingredient to include in the mix.

In our Swedish Steel example we have seven such variables:

$$x_j \triangleq \text{number of kilograms of ingredient } j \text{ included in the charge}$$

where $j = 1, \dots, 4$ refers to the four supplies of scrap, and $j = 5, \dots, 7$ to the pure additives.

Composition Constraints

One requirement on any solution to our example is that the total charge sum to 1000 kilograms:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 \tag{4.2}$$

However, most of the main constraints restrict the composition of the mix.

Definition 4.4 **Composition** constraints in blending models enforce upper and/or lower limits on the properties of the resulting blend.

In our Swedish Steel example we have both upper and lower limits on the fraction of carbon, nickel, chromium, and molybdenum in the mix. Each such constraint will have the form

$$\sum_j \begin{pmatrix} \text{fraction in} \\ j\text{th} \\ \text{ingredient} \end{pmatrix} \cdot \begin{pmatrix} \text{amount of} \\ j\text{th} \\ \text{ingredient} \\ \text{used} \end{pmatrix} \begin{matrix} \geq \\ \text{or} \\ \leq \end{matrix} \begin{pmatrix} \text{allowed} \\ \text{fraction in} \\ \text{the blend} \end{pmatrix} \cdot \begin{pmatrix} \text{blend} \\ \text{total} \end{pmatrix} \quad (4.3)$$

Specifically, the composition constraints required are

$$\begin{aligned} 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 &\geq 0.0065 \sum_{j=1}^7 x_j \\ 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 &\leq 0.0075 \sum_{j=1}^7 x_j \\ 0.180x_1 + 0.032x_2 + 1.0x_5 &\geq 0.030 \sum_{j=1}^7 x_j \\ 0.180x_1 + 0.032x_2 + 1.0x_5 &\leq 0.035 \sum_{j=1}^7 x_j \\ 0.120x_1 + 0.011x_2 + 1.0x_6 &\geq .010 \sum_{j=1}^7 x_j \\ 0.120x_1 + 0.011x_2 + 1.0x_6 &\leq 0.012 \sum_{j=1}^7 x_j \\ 0.001x_2 + 1.0x_7 &\geq 0.011 \sum_{j=1}^7 x_j \\ 0.001x_2 + 1.0x_7 &\leq 0.013 \sum_{j=1}^7 x_j \end{aligned}$$

EXAMPLE 4.2: FORMULATING COMPOSITION CONSTRAINTS

A food blending model with 3 ingredients employs nonnegative decision variables

$$x_j \triangleq \text{grams of ingredient } j \text{ used}$$

where ingredient 1 is 4% fiber and has 10 milligrams (mg) of sodium per gram, ingredient 2 is 9% fiber and has 15 mg of sodium per gram, and ingredient 3 is 3% fiber and has 5 mg of sodium per gram. Formulate linear constraints enforcing each of the following requirements.

- (a) The blend must average at least 5% fiber.
- (b) The blend must contain at most 100 mg of sodium.

Solution:

- (a) Following format (4.3), the needed constraint is

$$0.04x_1 + 0.09x_2 + 0.03x_3 \geq 0.05(x_1 + x_2 + x_3)$$

(b) Here the constraint deals with absolute amounts, not fractions. The required form is

$$10x_1 + 15x_2 + 5x_3 \leq 100$$

Swedish Steel Example Model

Collecting the elements derived so far yields the following LP model of our Swedish Steel example:

$$\begin{aligned}
 \min \quad & 16x_1 + 10x_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 && \text{(cost)} \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 && \text{(weight)} \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 && \text{(carbon)} \\
 & \quad + 0.0040x_4 \geq 0.0065(1000) \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 && \\
 & \quad + 0.0040x_4 \leq 0.0075(1000) \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 \geq 0.030(1000) && \text{(nickel)} \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 \leq 0.035(1000) && (4.4) \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 \geq 0.010(1000) && \text{(chromium)} \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 \leq 0.012(1000) \\
 & 0.001x_2 + 1.0x_7 \geq 0.011(1000) && \text{(molybdenum)} \\
 & 0.001x_2 + 1.0x_7 \leq 0.013(1000) \\
 & x_1 \leq 75 && \text{(available)} \\
 & x_2 \leq 250 \\
 & x_1, \dots, x_7 \geq 0 && \text{(nonnegative)}
 \end{aligned}$$

The objective function merely sums the costs of the ingredients used. The only constraints not discussed above enforce supply limits on the first two types of scrap. Composition constraints have been simplified by taking advantage of the fact that total weight is fixed at 1000 kilograms.

The unique optimal solution to model (4.4) has

$$\begin{aligned}
 x_1^* &= 75.00 \text{ kg}, & x_2^* &= 90.91 \text{ kg}, & x_3^* &= 672.28 \text{ kg}, & x_4^* &= 137.31 \text{ kg} \\
 x_5^* &= 13.59 \text{ kg}, & x_6^* &= 0.00 \text{ kg}, & x_7^* &= 10.91 \text{ kg}
 \end{aligned}$$

The total cost of an optimal charge is 9953.7 krona.

Ratio Constraints

The composition constraints of our Swedish Steel example can be viewed as **ratio constraints** because they bound the fraction that one weighted sum of variables forms of another. For example, the lower limit on carbon is

$$0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 \geq 0.0065 \sum_{j=1}^7 x_j$$

or

$$\frac{0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.004x_4}{x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7} \geq 0.0065$$

In the latter form, the constraint does not even appear linear. Still, multiplying by the denominator, which we know will be nonnegative in every feasible solution, produces the linear version without reversing the direction of the inequality.

Many blending models have similar ratio constraints among ingredients. For example, if ingredients 3 and 4 must be in ratio no more than 2:3, we have the requirement that

$$\frac{x_3}{x_4} \leq \frac{2}{3}$$

Again, since we know the sign of denominator $x_4 \geq 0$, we may cross-multiply to produce the linear form

$$x_3 \leq \frac{2}{3}x_4$$

Contrast this with the ratio constraint

$$\frac{x_3}{x_1 - x_2} \leq \frac{2}{3}$$

Over nonnegative x_j the denominator of the latter constraint has an unpredictable sign. Without further information it must be considered nonlinear.

Principle 4.5 **Ratio constraints**, which bound the quotient of linear functions by a constant, can often be converted to linear constraints by cross-multiplication. However, if the constraint is an inequality, the sign of the denominator function must be predictable over feasible solutions.

EXAMPLE 4.3: FORMULATING RATIO CONSTRAINTS

Formulate linear constraints enforcing each of the following ratio requirements on nonnegative decision variables x_1 , x_2 , and x_3 determining the amount of three substances in a blend.

- The amounts of substance 1 and 2 should be in the ratio 4:7.
- The amount of substance 1 is at most half that of substance 3.
- The blend is at least 40% substance 1.

Solution:

- The constraint required is

$$\frac{x_1}{x_2} = \frac{4}{7} \quad \text{or} \quad x_1 = \frac{4}{7}x_2$$

(b) The constraint needed is

$$x_1 \leq \frac{1}{2} x_3$$

(c) The constraint specified is

$$\frac{x_1}{x_1 + x_2 + x_3} \geq 0.40$$

or (with the nonnegative denominator)

$$x_1 \geq 0.40(x_1 + x_2 + x_3)$$

4.3 OPERATIONS PLANNING MODELS

Another classic linear program form deals with **operations planning**. In organizations ranging from volunteer, to government, to manufacturing, to distribution, planners must decide what to do and when and where to do it. For example, Section 2.3's Pi Hybrids model (2.10) was used to plan the production and distribution of seed corn.

APPLICATION 4.3: TUBULAR PRODUCTS OPERATIONS PLANNING

Sometimes a plan involves nothing more than allocation of work to operations. The Tubular Products Division (TP) of Babcock and Wilcox encountered just such a problem in investigating how work should be reallocated upon opening a new mill.³ TP manufactured steel tubing in a variety of sizes and for many different uses, including electrical power generation. At the time of the study three mills handled production. The object was to consider how a fourth mill of different configuration would affect the optimal distribution of work (and associated costs) among the mills.

Table 4.3 shows fictional data for existing mills 1 to 3 and one design for new mill 4, versus an array of 16 products. The products comprise all combinations of standard or high-pressure tubing: $\frac{1}{2}$ -, 1-, 2-, or 8-inch diameters, and thick or thin tube walls. The table includes the cost (in dollars) per 1000 pounds of each product according to which mill does the work, and the required processing time (in hours) per 1000 pounds produced. Missing values indicate products that cannot be manufactured feasibly at the mill indicated.

Table 4.3 also shows the assumed division-wide demand for each of the 16 products in thousands of pounds per week. At present the three existing mills 1 to 3 have 800, 480, and 1280 hours per week of effective production capacity, respectively. New mill 4 is planned for 960 hours per week.

³Based on Wayne Drayer and Steve Seabury (1975), "Facilities Expansion Model," *Interfaces*, 5:2, part 2, 104–109.

TABLE 4.3 Tubular Products Application Data

Product	Mill 1		Mill 2		Mill 3		Mill 4		Weekly Demand, d_p	
	Cost, $c_{p,1}$	Hours, $t_{p,1}$	Cost, $c_{p,2}$	Hours, $t_{p,2}$	Cost, $c_{p,3}$	Hours, $t_{p,3}$	Cost, $c_{p,4}$	Hours, $t_{p,4}$		
Standard										
1: $\frac{1}{2}$ -in. thick	90	0.8	75	0.7	70	0.5	63	0.6	100	
2: $\frac{1}{2}$ -in. thin	80	0.8	70	0.7	65	0.5	60	0.6	630	
3: 1-in. thick	104	0.8	85	0.7	83	0.5	77	0.6	500	
4: 1-in. thin	98	0.8	79	0.7	80	0.5	74	0.6	980	
5: 2-in. thick	123	0.8	101	0.7	110	0.5	99	0.6	720	
6: 2-in. thin	113	0.8	94	0.7	100	0.5	84	0.6	240	
7: 8-in. thick	—	—	160	0.9	156	0.5	140	0.6	75	
8: 8-in. thin	—	—	142	0.9	150	0.5	130	0.6	22	
Pressure										
9: $\frac{1}{2}$ -in. thick	140	1.5	110	0.9	—	—	122	1.2	50	
10: $\frac{1}{2}$ -in. thin	124	1.5	96	0.9	—	—	101	1.2	22	
11: 1-in. thick	160	1.5	133	0.9	—	—	138	1.2	353	
12: 1-in. thin	143	1.5	127	0.9	—	—	133	1.2	55	
13: 2-in. thick	202	1.5	150	0.9	—	—	160	1.2	125	
14: 2-in. thin	190	1.5	141	0.9	—	—	140	1.2	35	
15: 8-in. thick	—	—	190	1.0	—	—	220	1.5	100	
16: 8-in. thin	—	—	175	1.0	—	—	200	1.5	10	

Tubular Products Operations Planning Model

In operations planning models, the decisions always revolve around what operations to undertake. Here the problem has two index dimensions:

$$p \triangleq \text{product number } (p = 1, \dots, 16)$$

$$m \triangleq \text{mill number } (m = 1, \dots, 4)$$

The corresponding decision variables are

$$x_{p,m} \triangleq \text{amount of product } p \text{ produced at mill } m \\ \text{(in thousands of pounds per week)}$$

In terms of these decision variables, it is straightforward to produce a linear programming model. For brevity, define the symbolic constants

$$c_{p,m} \triangleq \text{unit cost of producing product } p \text{ at mill } m \text{ shown in Table 4.3 } [= +\infty \\ \text{if this } (p, m) \text{ combination is impossible}]$$

$$t_{p,m} \triangleq \text{unit time to manufacture product } p \text{ at mill } m \text{ shown in Table 4.3 } [= 0 \\ \text{if this } (p, m) \text{ combination is impossible}]$$

$$d_p \triangleq \text{weekly demand for product } p \text{ shown in Table 4.3}$$

$$b_m \triangleq \text{given production capacity at mill } m$$

Then the model required is

$$\begin{aligned}
 \min \quad & \sum_{p=1}^{16} \sum_{m=1}^4 c_{p,m} x_{p,m} && \text{(total cost)} \\
 \text{s.t.} \quad & \sum_{m=1}^4 x_{p,m} \geq d_p \quad p = 1, \dots, 16 && \text{(demands)} \\
 & \sum_{p=1}^{16} t_{p,m} x_{p,m} \leq b_m \quad m = 1, \dots, 4 && \text{(capacities)} \\
 & x_{p,m} \geq 0 \quad p = 1, \dots, 16: \quad m = 1, \dots, 4
 \end{aligned} \tag{4.5}$$

The objective minimizes total production cost. One system of main constraints enforces product demands and the other mill capacities.

Table 4.4 shows an optimal solution \mathbf{x}^* . Old mill 1 should go virtually unused; mills 2 and 3 concentrate on pressure and standard tubing, respectively; and new mill 4 satisfies the remainder of demand for both. Total weekly cost is \$378,899.

TABLE 4.4 Tubular Products Example Optimum

Product	Mill 1, $x_{p,1}^*$	Mill 2, $x_{p,2}^*$	Mill 3, $x_{p,3}^*$	Mill 4, $x_{p,4}^*$
<u>Standard</u>				
1: $\frac{1}{2}$ in. thick	0.0	0.0	0.0	100.0
2: $\frac{1}{2}$ in. thin	0.0	0.0	630.0	0.0
3: 1 in. thick	0.0	0.0	404.8	95.2
4: 1 in. thin	0.0	0.0	980.0	0.0
5: 2 in. thick	0.0	0.0	0.0	720.0
6: 2 in. thin	0.0	0.0	0.0	240.0
7: 8 in. thick	—	0.0	0.0	75.0
8: 8 in. thin	—	0.0	0.0	22.0
<u>Pressure</u>				
9: $\frac{1}{2}$ in. thick	0.0	50.0	—	0.0
10: $\frac{1}{2}$ in. thin	0.0	22.0	—	0.0
11: 1 in. thick	0.0	214.1	—	138.9
12: 1 in. thin	55.0	0.0	—	0.0
13: 2 in. thick	0.0	125.0	—	0.0
14: 2 in. thin	0.0	0.0	—	35.0
15: 8 in. thick	—	100.0	—	0.0
16: 8 in. thin	—	10.0	—	0.0

**APPLICATION 4.4: CANADIAN FOREST PRODUCTS LIMITED (CFPL)
OPERATIONS PLANNING**

Operations planning models become more complex when there are several stages of production. Activity at each stage consumes output of the preceding stage and creates input to the next stage.

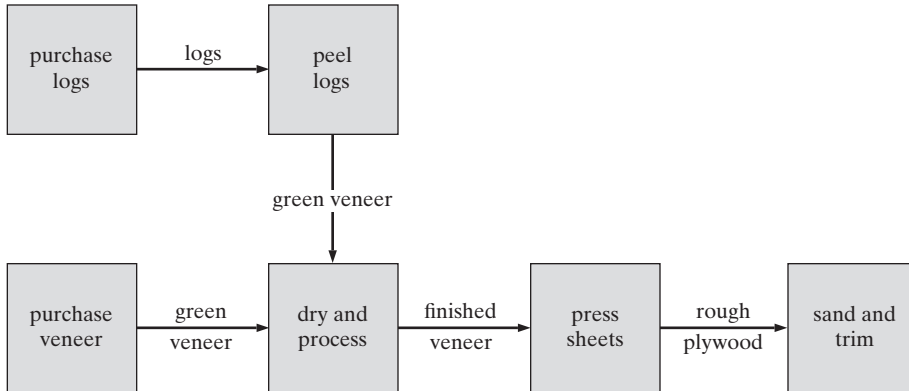


FIGURE 4.1 Plywood Processing Flow in CFPL Example

Canadian Forest Products Limited (CFPL) employed such a model to plan their production of plywood.⁴ Figure 4.1 shows the sequence of stages. Production begins by purchasing logs and peeling them into strips of thin “green” veneer. Green veneer can also be purchased directly. All green veneer is next dried, classified by quality, and in some cases improved by patching knots and gluing thin strips together. After the veneer has been cut into sheet sizes, several layers are glued and pressed to produce plywood. A final production step sands completed plywood and trims it to exact size for sale.

The objective of CFPL’s operations research analysis was to determine how to operate production facilities to maximize *contributed margin*: sales income less wood costs. Labor, maintenance, and other plant costs were assumed fixed. The principal constraint, other than limits on availability of wood and the market for various products, was the limited plant capacity to press plywood.

To have some numbers to work with, assume that logs are available from two vendors in “good” and “fair” qualities at the rate and price shown below. The table also shows the estimated yield in $\frac{1}{16}$ - and $\frac{1}{8}$ -inch green veneer of grades A, B, and C from peeling a log of the quality indicated.

	Veneer Yield (sq ft)			
	Vendor 1		Vendor 2	
	Good	Fair	Good	Fair
Available per month	200	300	100	1000
Cost per log (\$ Canadian)	340	190	490	140
A $\frac{1}{16}$ -inch green veneer (sq ft)	400	200	400	200
B $\frac{1}{16}$ -inch green veneer (sq ft)	700	500	700	500
C $\frac{1}{16}$ -inch green veneer (sq ft)	900	1300	900	1300
A $\frac{1}{8}$ -inch green veneer (sq ft)	200	100	200	100
B $\frac{1}{8}$ -inch green veneer (sq ft)	350	250	350	250
C $\frac{1}{8}$ -inch green veneer (sq ft)	450	650	450	650

⁴Based on D. B. Kotak (1976), “Application of Linear Programming to Plywood Manufacture,” *Interfaces*, 7:1, part 2, 56–68.

We can also purchase green veneer. Suppose that availabilities and purchase prices are as shown in the following table.

	$\frac{1}{16}$ -Inch Green Veneer			$\frac{1}{8}$ -Inch Green Veneer		
	A	B	C	A	B	C
Available (sq ft / month)	5,000	25,000	40,000	10,000	40,000	50,000
Cost (\$ Canadian /sq ft)	1.00	0.30	0.10	2.20	0.60	0.20

Our version of CFPL will make just 6 products—all 4- by 8-foot sheets of plywood for the U.S. market. A final table shows the composition or **Bill of Materials** of each product in veneer sheets, the available market per month, and the time required to glue and press each sheet of plywood out of a monthly capacity of 4500 hours.

	$\frac{1}{16}$ -Inch Plywood Sheets			$\frac{1}{2}$ -Inch Plywood Sheets		
	AB	AC	BC	AB	AC	BC
Front veneer	$\frac{1}{16}$ A	$\frac{1}{16}$ A	$\frac{1}{16}$ B	$\frac{1}{16}$ A	$\frac{1}{16}$ A	$\frac{1}{16}$ B
Core veneer	$\frac{1}{8}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C
				$\frac{1}{8}$ B	$\frac{1}{8}$ B	$\frac{1}{8}$ B
Back veneer	$\frac{1}{16}$ B	$\frac{1}{16}$ C	$\frac{1}{16}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C	$\frac{1}{8}$ C
				$\frac{1}{16}$ B	$\frac{1}{16}$ C	$\frac{1}{16}$ C
Market per month	1000	4000	8000	1000	5000	8000
Price (\$ Canadian)	45.00	40.00	33.00	75.00	65.00	50.00
Pressing time (hours)	0.25	0.25	0.25	0.45	0.40	0.40

CFPL Decision Variables

As usual, we begin a model for the CFPL case by choosing variables deciding how much of what to do. Index dimensions include

$q \triangleq$ log quality ($q = G$ for good, F for fair)

$v \triangleq$ log vendor number ($v = 1, 2$)

$t \triangleq$ veneer thickness ($t = \frac{1}{16}, \frac{1}{8}$)

$g \triangleq$ veneer grade ($g = A, B, C$)

To formulate the problem as a linear program, we will use four classes of (continuous) decision variables over these index dimensions:

$w_{q,v,t} \triangleq$ number of logs of quality q bought from vendor v and peeled into green veneer of thickness t per month

$x_{t,g} \triangleq$ number of square feet of thickness t , grade g green veneer purchased directly per month

$y_{t,g,g'} \triangleq$ number of sheets of thickness t veneer used as grade g' after drying and processing from grade g green veneer per month

$z_{t,g,g'} \triangleq$ number of sheets of thickness t , front veneer grade g , back veneer grade g' plywood pressed and sold per month

Notice that these variables correspond to only 4 of the 6 processing boxes in Figure 4.1. This efficiency is possible because the way we have presented the problem offers no advantage for purchasing a log and not peeling it, or pressing a sheet of plywood and not sanding or selling it. If inventories had to be modeled, so that purchased logs, for example, need not be immediately peeled, we would require many more variables.

Continuous Variables for Integer Quantities

Readers who are studying LP modeling for the first time may be perplexed about the fact that the CFPL decision variables are all treated as continuous. Don't quantities such as the number of logs and the number of sheets of plywood need to be integers? Indeed, how can CFPL's problem even be modeled as a linear program (which must have only continuous variable)?

Modeling physically integer quantities with continuous decision variables in this fashion is standard when optimal variable magnitudes are likely to be relatively large (principle [1.11](#)). If the LP-optimal number of plywood sheets sold of some type turns out to be, say, 953.2, there is little practical difficulty in rounding off to 953 sheets. After all, the costs, capacities, and other constants in the model are only estimates that contain a certain amount of error.

But we know that there is a big gain in tractability. Continuous optimization is almost always more efficient than discrete. To realize that gain without having much impact on the usability of optimal results, we choose to neglect integrality requirements.

Principle 4.6 | To gain tractability with little loss of validity, decision variables of relatively large magnitude are best modeled as continuous, even though they correspond to physically integer quantities.

Notice that this concession to tractability would be much more serious when decision variables were limited to, say, 0 and 1. If, for example, 0 means “do not build a facility” and 1 means “build it,” rounding continuous LP solutions could be much more problematic.

CFPL Objective Function

CFPL's maximum contributed margin objective is easily expressed in terms of the decision variables above. We compute

$$\max \quad - (\text{log costs}) - (\text{purchased veneer costs}) + (\text{sales income})$$

That is,

$$\begin{aligned} \max \quad & - (340w_{G,1,1/16} + 190w_{F,1,1/16} + 490w_{G,2,1/16} + 140w_{F,2,1/16}) \quad (4.6) \\ & + 340w_{G,1,1/8} + 190w_{F,1,1/8} + 490w_{G,2,1/8} + 140w_{F,2,1/8}) \\ & - (1.00x_{1/16,A} + 0.30x_{1/16,B} + 0.10x_{1/16,C} + 2.20x_{1/8,A}) \\ & + 0.60x_{1/8,B} + 0.20x_{1/8,C}) + (45z_{1/4,A,B} + 40z_{1/4,A,C} \\ & + 33z_{1/4,B,C} + 75z_{1/2,A,B} + 65z_{1/2,A,C} + 50z_{1/2,B,C}) \end{aligned}$$

CFPL Constraints

Some constraints are equally easy. Log availability limits impose

$$\begin{aligned} w_{G,1,1/16} + w_{G,1,1/8} &\leq 200, & w_{F,1,1/16} + w_{F,1,1/8} &\leq 300 \\ w_{G,2,1/16} + w_{G,2,1/8} &\leq 100, & w_{F,2,1/16} + w_{F,2,1/8} &\leq 1000 \end{aligned} \tag{4.7}$$

purchased veneer availabilities imply that

$$\begin{aligned} x_{1/16,A} &\leq 5000, & x_{1/16,B} &\leq 25,000, & x_{1/16,C} &\leq 40,000 \\ x_{1/8,A} &\leq 10,000, & x_{1/8,B} &\leq 40,000, & x_{1/8,C} &\leq 50,000 \end{aligned} \tag{4.8}$$

and market sizes constrain

$$\begin{aligned} z_{1/4,A,B} &\leq 1000, & z_{1/4,A,C} &\leq 4000, & z_{1/4,B,C} &\leq 8000 \\ z_{1/2,A,B} &\leq 1000, & z_{1/2,A,C} &\leq 5000, & z_{1/2,B,C} &\leq 8000 \end{aligned} \tag{4.9}$$

Finally, the important pressing capacity limit yields the additional constraint

$$0.25(z_{1/4,A,B} + z_{1/4,A,C} + z_{1/4,B,C}) + 0.40(z_{1/2,A,B} + z_{1/2,A,C} + z_{1/2,B,C}) \leq 4500 \tag{4.10}$$

Balance Constraints

So far we have done nothing to link log and veneer purchasing at the beginning of the process to sales at the end. In fact, we have not used the processing variables $y_{l,g,g}$, at all.

What makes operations planning models with several processing stages special is the need to provide such links through balance constraints.

Definition 4.7 | A **balance constraint** assures that in-flows equal or exceed out-flows for materials and products created by one stage of production and consumed by others.

The first family of balance constraints needed in the CFPL model involves green veneer. Assume that with trim losses, 35 square feet of green veneer is required for each 4-by 8-foot sheet of finished veneer. We then have for each thickness and grade

$$(\text{veneer from peeled logs}) + (\text{veneer purchased}) \geq 35(\text{sheets of veneer finished})$$

Assuming that careful piecing and patching can permit green veneer of one grade to be used as the next higher, and veneer of any grade can be substituted for the next lower, we obtain the following six balance constraints for various grades and thicknesses of green veneer:

$$\begin{aligned} 400w_{G,1,1/16} + 200w_{F,1,1/16} + 400w_{G,2,1/16} + 200w_{F,2,1/16} + x_{1/16,A} &\geq 35y_{1/16,A,A} + 35y_{1/16,A,B} \\ 700w_{G,1,1/16} + 500w_{F,1,1/16} + 700w_{G,2,1/16} + 500w_{F,2,1/16} + x_{1/16,B} &\geq 35y_{1/16,B,A} + 35y_{1/16,B,B} + 35y_{1/16,B,C} \\ 900w_{G,1,1/16} + 1300w_{F,1,1/16} + 900w_{G,2,1/16} + 1300w_{F,2,1/16} + x_{1/16,C} &\geq 35y_{1/16,C,B} + 35y_{1/16,C,C} \end{aligned} \tag{4.11}$$

$$\begin{aligned}
& 200w_{G,1,1/8} + 100w_{F,1,1/8} + 200w_{G,2,1/8} + 100w_{F,2,1/8} + x_{1/8,A} \\
& \geq 35y_{1/8,A,A} + 35y_{1/8,A,B} \\
& 350w_{G,1,1/8} + 250w_{F,1,1/8} + 350w_{G,2,1/8} + 250w_{F,2,1/8} + x_{1/8,B} \\
& \geq 35y_{1/8,B,A} + 35y_{1/8,B,B} + 35y_{1/8,B,C} \\
& 450w_{G,1,1/8} + 650w_{F,1,1/8} + 450w_{G,2,1/8} + 650w_{F,2,1/8} + x_{1/8,C} \\
& \geq 35y_{1/8,C,B} + 35y_{1/8,C,C}
\end{aligned}$$

Six quite similar constraints enforce balance in sheets of finished veneer passing from the drying process to pressing:

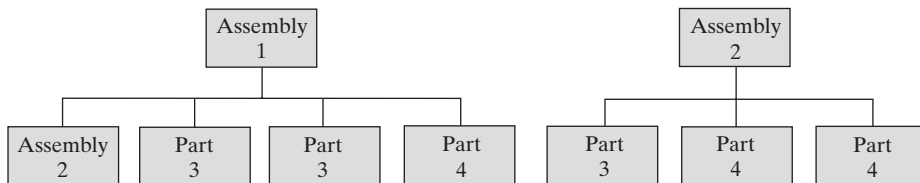
sheets finished for use at this grade = sheets consumed in pressing

We can make the constraints equalities this time because no veneer would ever be finished unless it were going to be pressed. Again detailing for two thicknesses and three grades (other than the never-used $\frac{1}{8}$ -inch, grade A finished veneer) gives

$$\begin{aligned}
& y_{1/16,A,A} + y_{1/16,B,A} \\
& = z_{1/4,A,B} + z_{1/4,A,C} + z_{1/2,A,B} + z_{1/2,A,C} \\
& y_{1/16,A,B} + y_{1/16,B,B} + y_{1/16,C,B} \\
& = z_{1/4,A,B} + z_{1/4,B,C} + z_{1/2,A,B} + z_{1/2,B,C} \\
& y_{1/16,B,C} + y_{1/16,C,C} \\
& = z_{1/4,A,C} + z_{1/4,B,C} + z_{1/2,A,C} + z_{1/2,B,C} \\
& y_{1/8,A,B} + y_{1/8,B,B} + y_{1/8,C,B} \\
& = z_{1/2,A,B} + z_{1/2,A,C} + z_{1/2,B,C} \\
& y_{1/8,B,C} + y_{1/8,C,C} \\
& = z_{1/4,A,B} + z_{1/4,A,C} + z_{1/4,B,C} + 2z_{1/2,A,B} + 2z_{1/2,A,C} + 2z_{1/2,B,C}
\end{aligned} \tag{4.12}$$

EXAMPLE 4.4: FORMULATING BALANCE CONSTRAINTS

The following figure shows the assembly structure (Bill of Materials) of two products:



Use decision variables

$x_j \triangleq$ number of parts or assemblies j produced

to formulate balance constraints for assemblies/parts $j = 2, 3, 4$.

Solution: Assembly 1 joins an assembly 2 with two part 3's and a part 4. Assembly 2 consists of one part 3 and two part 4's. Thus for $j = 2$, we require that the number of assembly 2's at least equal the number required for assembly 1's:

$$x_2 \geq x_1$$

Similarly, for parts $j = 3, 4$, we require production to at least meet requirements for the assemblies:

$$x_3 \geq 2x_1 + 1x_2$$

$$x_4 \geq 1x_1 + 2x_2$$

CFPL Application Model

Collecting (4.6) to (4.12) and adding variable-type constraints produces the full CFPL linear programming model detailed in Table 4.5. One optimal solution has the following variables nonzero:

$$\begin{aligned}
 w_{G,1,1/16}^* &= 41.3, & w_{F,1,1/16}^* &= 300.0, & w_{F,2,1/16}^* &= 155.3 \\
 w_{F,2,1/8}^* &= 844.7, & x_{1/16,C}^* &= 40,000.0, & x_{1/8,C}^* &= 50,000.0 \\
 y_{1/16,A,A}^* &= 3073.2, & y_{1/16,B,A}^* &= 7329.4, & y_{1/16,C,B}^* &= 6355.8 \\
 y_{1/16,C,C}^* &= 12,758.4, & y_{1/8,A,B}^* &= 2413.5, & y_{1/8,B,C}^* &= 6033.8 \\
 y_{1/8,C,B}^* &= 2989.1, & y_{1/8,C,C}^* &= 14,127.3, & & \\
 z_{1/4,A,B}^* &= 1000.0, & z_{1/4,A,C}^* &= 4000.0, & z_{1/4,B,C}^* &= 4355.8 \\
 z_{1/2,A,B}^* &= 1000.0, & z_{1/2,A,C}^* &= 4402.6 & &
 \end{aligned} \tag{4.13}$$

The firm should enter all markets except the one for $\frac{1}{2}$ -inch BC plywood. Total contributed margin is \$484,878 Canadian per month.

Fractions in such variables as the number of sheets of plywood sold are physically impossible. Still, the advantage of globally solving this complex optimization efficiently with linear programming far outweighs the minute inaccuracy associated with rounding the LP optimum to obtain a plan.

EXAMPLE 4.5: FORMULATING OPERATIONS PLANNING LPs

An orange juice company can sell up to 15,000 tons of juice to wholesalers at \$1500 per ton. The juice is either squeezed from oranges purchased at \$200 per ton or diluted from concentrate obtained at \$1600 per ton. Approximately 10,000 tons of juice oranges are available and each yields 0.2 ton of juice. The supply of concentrate is essentially unlimited, and each ton dilutes into 2 tons of juice. Formulate a linear program to choose an operating plan that maximizes the company's net income (sales minus cost).

Solution: We define decision variables for each of the 3 operations:

$x_1 \triangleq$ tons of oranges squeezed for juice

$x_2 \triangleq$ tons of concentrate diluted for juice

$x_3 \triangleq$ tons of juice sold

Then the required model is

$$\begin{aligned}
 \max \quad & -200x_1 - 1600x_2 + 1500x_3 && \text{(net income)} \\
 \text{s.t.} \quad & x_1 \leq 10,000 && \text{(orange availability)} \\
 & x_3 \leq 15,000 && \text{(sales limit)} \\
 & 0.2x_1 + 2x_2 = x_3 && \text{(balance)} \\
 & x_1, x_2, x_3 \geq 0
 \end{aligned}$$

Its objective function maximizes the difference of purchase cost and sales revenue. The first main constraint enforces the limit on orange availability, the second limits sales to 15,000, and the third balances production with sales.

TABLE 4.5 CFPL Application Linear Program Model

max	$-(340w_{G,1,1/16} + 190w_{F,1,1/16} + 490w_{G,2,1/16} + 140w_{F,2,1/16} + 340w_{G,1,1/8} + 190w_{F,1,1/8} + 490w_{G,2,1/8} + 140w_{F,2,1/8})$	(logs)
	$- (1.00x_{1/16,A} + 0.30x_{1/16,B} + 0.10x_{1/16,C} + 2.20x_{1/8,A} + 0.60x_{1/8,B} + 0.20x_{1/8,C})$	(vener)
	$+ (45z_{1/4,A,B} + 40z_{1/4,A,C} + 33z_{1/4,B,C} + 75z_{1/2,A,B} + 65z_{1/2,A,C} + 50z_{1/2,B,C})$	(sales)
s.t.	$w_{G,1,1/16} + w_{G,1,1/8} \leq 200, w_{F,1,1/16} + w_{F,1,1/8} \leq 300$	(log availability)
	$w_{G,2,1/16} + w_{G,2,1/8} \leq 100, w_{F,2,1/16} + w_{F,2,1/8} \leq 1000$	
	$x_{1/16,A} \leq 5000, x_{1/16,B} \leq 25,000, x_{1/16,C} \leq 40,000$	(vener)
	$x_{1/8,A} \leq 10,000, x_{1/8,B} \leq 40,000, x_{1/8,C} \leq 50,000$	(availability)
	$z_{1/4,A,B} \leq 1000, z_{1/4,A,C} \leq 4000, z_{1/4,B,C} \leq 8000$	(market)
	$z_{1/2,A,B} \leq 1000, z_{1/2,A,C} \leq 5000, z_{1/2,B,C} \leq 8000$	
	$0.25(z_{1/4,A,B} + z_{1/4,A,C} + z_{1/4,B,C}) + 0.40(z_{1/2,A,B} + z_{1/2,A,C} + z_{1/2,B,C}) \leq 4500$	(pressing)
	$400w_{G,1,1/16} + 200w_{F,1,1/16} + 400w_{G,2,1/16} + 200w_{F,2,1/16} + x_{1/16,A} \geq 35y_{1/16,A,A} + 35y_{1/16,A,B}$	(green)
	$700w_{G,1,1/16} + 500w_{F,1,1/16} + 700w_{G,2,1/16} + 500w_{F,2,1/16} + x_{1/16,B} \geq 35y_{1/16,B,A} + 35y_{1/16,B,B} + 35y_{1/16,B,C}$	(vener)
	$900w_{G,1,1/16} + 1300w_{F,1,1/16} + 900w_{G,2,1/16} + 1300w_{F,2,1/16} + x_{1/16,C} \geq 35y_{1/16,C,A} + 35y_{1/16,C,B} + 35y_{1/16,C,C}$	(balance)
	$200w_{G,1,1/8} + 100w_{F,1,1/8} + 200w_{G,2,1/8} + 100w_{F,2,1/8} + x_{1/8,A} \geq 35y_{1/8,A,A} + 35y_{1/8,A,B}$	
	$350w_{G,1,1/8} + 250w_{F,1,1/8} + 350w_{G,2,1/8} + 250w_{F,2,1/8} + x_{1/8,B} \geq 35y_{1/8,B,A} + 35y_{1/8,B,B} + 35y_{1/8,B,C}$	
	$450w_{G,1,1/8} + 650w_{F,1,1/8} + 450w_{G,2,1/8} + 650w_{F,2,1/8} + x_{1/8,C} \geq 35y_{1/8,C,A} + 35y_{1/8,C,B} + 35y_{1/8,C,C}$	
	$y_{1/16,A,A} + y_{1/16,B,A} = z_{1/4,A,B} + z_{1/4,A,C} + z_{1/2,A,B} + z_{1/2,A,C}$	(finished)
	$y_{1/16,A,B} + y_{1/16,B,B} + y_{1/16,C,B} = z_{1/4,A,B} + z_{1/4,B,C} + z_{1/2,A,B} + z_{1/2,B,C}$	(vener)
	$y_{1/16,B,C} + y_{1/16,C,C} = z_{1/4,A,C} + z_{1/4,B,C} + z_{1/2,A,C} + z_{1/2,B,C}$	(balance)
	$y_{1/8,A,B} + y_{1/8,B,B} + y_{1/8,C,B} = z_{1/2,A,B} + z_{1/2,A,C} + z_{1/2,B,C}$	
	$y_{1/8,B,C} + y_{1/8,C,C} = z_{1/4,A,B} + z_{1/4,A,C} + z_{1/4,B,C} + 2z_{1/2,A,B} + 2z_{1/2,A,C} + 2z_{1/2,B,C}$	
	all variables $w_{q,v,r}, x_{l,g}, y_{l,g,g'}, z_{l,g,g'} \geq 0$	

4.4 SHIFT SCHEDULING AND STAFF PLANNING MODELS

Operations planning models decide what work to undertake so that available resources are used efficiently. In **shift scheduling** or **staff planning models** the work is already fixed. We must now plan the resources to accomplish it. In particular, we must decide how many of what types of workers and shifts best cover all work requirements. Again, LP provides a powerful tool.

APPLICATION 4.5: OHIO NATIONAL BANK (ONB) SHIFT SCHEDULING

The Ohio National Bank (ONB) confronted such a problem in staffing its check processing center.⁵ Checks received by the bank already have account numbers and other identifying information encoded on them. Machine operators in the check processing center key the dollar amount of the check, which is then imprinted with the other information for computerized processing.

Checks arrive through the business day in volumes peaking in the early evening. Our fictitious version will assume the following arrivals (in thousands):

Hour	Arrivals	Hour	Arrivals
11:00 (11 A.M.)	10	17:00 (5 P.M.)	32
12:00 (noon)	11	18:00 (6 P.M.)	50
13:00 (1 P.M.)	15	19:00 (7 P.M.)	30
14:00 (2 P.M.)	20	20:00 (8 P.M.)	20
15:00 (3 P.M.)	25	21:00 (9 P.M.)	8
16:00 (4 P.M.)	28	—	—

Uncollected checks cost the bank money in lost interest. Thus it is essential that all checks be processed in time for collection on the next business day. ONB decided to enforce a requirement that all checks be completed by 22:00 (10 P.M.). Furthermore, the number unprocessed at any hour should not exceed 20 thousand.

Two types of employees can perform the check processing task. Full-time employees work an 8-hour shift with a 1-hour lunch break in the middle. Part-time employees work only 4 hours per day with no lunch. Both types of shifts can begin at any hour of the day, and full-time employees can be assigned an hour of overtime. Table 4.6 illustrates the possible shifts.

In our analysis we assume that full-time employees receive \$11 per hour in pay and benefits, plus an extra \$1 per hour in “night differential” for time after 6 P.M. and 150% pay for daily overtime. Part-time employees are paid \$7 per hour, plus \$1 per hour night differential after 6 P.M. Also, to keep overtime under control, we require that no more than half the full-time employees on any shift work overtime and that the total number of scheduled overtime hours not exceed 20 per day.

Naturally, full-time employees work faster than part-timers. We will assume that full-time operators process 1000 checks per hour, and part-timers only 800.

⁵Based on L. J. Krajewski, L. P. Ritzman, and P. McKenzie (1980), “Shift Scheduling in Banking Operations: A Case Application,” *Interfaces*, 10:2, 1–8.

One final complication is encoding stations. The number of machines available limits the number of employees who can work at any one time. Our center will have 35 machines.

TABLE 4.6 Possible Shifts in ONB Example^a

Start	Full-Time Shifts			Part-Time Shifts							
	11	12	13	11	12	13	14	15	16	17	18
11:00	R	—	—	R	—	—	—	—	—	—	—
12:00	R	R	—	R	R	—	—	—	—	—	—
13:00	R	R	R	R	R	R	—	—	—	—	—
14:00	R	R	R	R	R	R	R	—	—	—	—
15:00	—	R	R	—	R	R	R	R	—	—	—
16:00	R	—	R	—	—	R	R	R	R	—	—
17:00	R	R	—	—	—	—	R	R	R	R	—
18:00	RN	RN	RN	—	—	—	—	RN	RN	RN	RN
19:00	RN	RN	RN	—	—	—	—	—	RN	RN	RN
20:00	ON	RN	RN	—	—	—	—	—	—	RN	RN
21:00	—	ON	RN	—	—	—	—	—	—	—	RN

^a R, regular duty; O, possible overtime; N, night differential.

ONB Decision Variables and Objective Function

The main decisions to be made in shift scheduling models are the number of employees to work various shifts. In the ONB case we have all the possibilities in Table 4.6. For example, the full-time shift starting at 11:00 works 4 hours, then takes a lunch break, then works 4 more hours. The final 2 hours come after 6 P.M., so a night differential applies. One additional hour may also be worked in overtime.

Using the index

$$h \triangleq (\text{24-hour clock}) \text{ shift start time}$$

we define the corresponding decision variables:

$$x_h \triangleq \text{number of full-time employees beginning a shift at hour } h \\ (h = 11, \dots, 13)$$

$$y_h \triangleq \text{number of full-time employees with shift beginning at hour } h \text{ who} \\ \text{work overtime } (h = 11, 12)$$

$$z_h \triangleq \text{number of part-time employees beginning a shift at hour } h \\ (h = 11, \dots, 18)$$

We need only add up the pay for each shift to obtain a minimum (daily) cost objective function:

$$\begin{aligned} \min \quad & 90x_{11} + 91x_{12} + 92x_{13} + 18y_{11} + 18y_{12} + 28z_{11} + 28z_{12} \\ & + 28z_{13} + 28z_{14} + 29z_{15} + 30z_{16} + 31z_{17} + 32z_{18} \end{aligned} \quad (4.14)$$

For example, the coefficient on x_{13} reflects 8 regular hours at \$11 per hour plus \$4 for the 4 hours after 6 P.M., or

$$8(\$11) + 4(\$1) = \$92$$

ONB Constraints

Table 4.6 also suggests how to model the requirement that no more than 35 operators be on duty at any time. We simply constrain the sum of full-time, overtime, and part-time employees on duty in each hour.

$$\begin{array}{rcl}
 x_{11} + z_{11} & \leq 35 & (11:00 \text{ machines}) \\
 x_{11} + x_{12} + z_{11} + z_{12} & \leq 35 & (12:00 \text{ machines}) \\
 \vdots & \vdots & \vdots \\
 y_{11} + x_{12} + x_{13} + z_{17} + z_{18} & \leq 35 & (20:00 \text{ machines}) \\
 y_{12} + x_{13} + z_{18} & \leq 35 & (21:00 \text{ machines})
 \end{array} \tag{4.15}$$

There are also overtime limits. Overtime cannot exceed half of any full-time shift or total more than 20 hours per day. These limits lead us to the constraints

$$\begin{array}{rcl}
 y_{11} & \leq \frac{1}{2} x_{11} & (11\text{-shift overtime}) \\
 y_{12} & \leq \frac{1}{2} x_{12} & (12\text{-shift overtime}) \\
 y_{11} + y_{12} & \leq 20 & (\text{total overtime})
 \end{array} \tag{4.16}$$

Covering Constraints

The main element in any staff planning model is a collection of covering constraints.

Definition 4.8 **Covering constraints** in shift scheduling models assure that the shifts chosen provide enough worker output to cover requirements over each time period; that is,

$$\sum_{\text{shifts}} (\text{output/worker}) (\text{number on duty}) \geq \text{period requirement}$$

With the ONB case we have a slight complication in covering requirements. Work arrivals are specified on an hour-by-hour basis, but work completion is limited only by all checks being finished at 22:00 (10 P.M.). To model covering in such a case, we need some new decision variables reflecting the work carried over. Specifically, define

$w_h \triangleq$ uncompleted work backlog at (24-hour clock) hour h (in thousands)

Then our ONB covering constraints take the form

$$\begin{array}{rcl}
 1x_{11} + 0.8z_{11} & \geq 10 - w_{12} & (11:00 \text{ cover}) \\
 1x_{11} + 1x_{12} + 0.8z_{11} + 0.8z_{12} & \geq 11 + w_{12} - w_{13} & (12:00 \text{ cover}) \\
 \vdots & \vdots & \vdots \\
 1y_{11} + 1x_{12} + 1x_{13} + 0.8z_{17} + 0.8z_{18} & \geq 20 + w_{20} - w_{21} & (20:00 \text{ cover}) \\
 1y_{12} + 1x_{13} + 0.8z_{18} & \geq 8 + w_{21} & (21:00 \text{ cover})
 \end{array} \tag{4.17}$$

For example, the one for the 20:00 hour requires the total output of workers on duty from 20:00 to 21:00 to equal or exceed the 20 thousand checks arriving at that hour

(see the table at the beginning of Example 4.5), plus checks held over from previous hours (w_{20}), less those passed on to later hours (w_{21}).

ONB Shift Scheduling Application Model

Combining (4.14) to (4.17) with suitable variable-type constraints and upper bounds of 20 on all backlog variables w_h produces the full ONB shift scheduling linear program shown in Table 4.7. An LP optimum makes the following variables nonzero:

$$\begin{aligned} x_{12}^* &= 8.57, & x_{13}^* &= 12.86, & y_{12}^* &= 4.29, & z_{14}^* &= 13.57, & z_{16}^* &= 5.36, \\ z_{17}^* &= 7.50 & z_{18}^* &= 0.71, & w_{12}^* &= 10.00, & w_{13}^* &= 12.43, \\ w_{14}^* &= 6.00, & w_{18}^* &= 2.29, & w_{19}^* &= 20.00 & w_{20}^* &= 17.71, & w_{21}^* &= 9.71 \end{aligned}$$

That is, full-time employees carry the load early in the day, with part-time beginning at 14:00 (2:00 P.M.) Total cost is \$2836 per day.

TABLE 4.7 ONB Shift Scheduling Application LP Model

min	$90x_{11} + 91x_{12} + 92x_{13} + 18y_{11} + 18y_{12} + 28z_{11} + 28z_{12} + 28z_{13} + 28z_{14} + 29z_{15} + 30z_{16} + 31z_{17} + 32z_{18}$		(total pay)
s.t.	$x_{11} + z_{11}$	≤ 35	(11:00 machine)
	$x_{11} + x_{12} + z_{11} + z_{12}$	≤ 35	(12:00 machine)
	$x_{11} + x_{12} + x_{13} + z_{11} + z_{12} + z_{13}$	≤ 35	(13:00 machine)
	$x_{11} + x_{12} + x_{13} + z_{11} + z_{12} + z_{13} + z_{14}$	≤ 35	(14:00 machine)
	$x_{12} + x_{13} + z_{12} + z_{13} + z_{14} + z_{15}$	≤ 35	(15:00 machine)
	$x_{11} + x_{13} + z_{13} + z_{14} + z_{15} + z_{16}$	≤ 35	(16:00 machine)
	$x_{11} + x_{12} + z_{14} + z_{15} + z_{16} + z_{17}$	≤ 35	(17:00 machine)
	$x_{11} + x_{12} + x_{13} + z_{15} + z_{16} + z_{17} + z_{18}$	≤ 35	(18:00 machine)
	$x_{11} + x_{12} + x_{13} + z_{16} + z_{17} + z_{18}$	≤ 35	(19:00 machine)
	$y_{11} + x_{12} + x_{13} + z_{17} + z_{18}$	≤ 35	(20:00 machine)
	$y_{12} + x_{13} + z_{18}$	≤ 35	(21:00 machine)
	y_{11}	$\leq \frac{1}{2}x_{11}$	(11-shift overtime)
	y_{12}	$\leq \frac{1}{2}x_{12}$	(12-shift overtime)
	$y_{11} + y_{12}$	≤ 20	(total overtime)
	$1x_{11} + 0.8z_{11}$	$\geq 10 - w_{12}$	(11:00 cover)
	$1x_{11} + 1x_{12} + 0.8z_{11} + 0.8z_{12}$	$\geq 11 + w_{12} - w_{13}$	(12:00 cover)
	$1x_{11} + 1x_{12} + 1x_{13} + 0.8z_{11} + 0.8z_{12} + 0.8z_{13}$	$\geq 15 + w_{13} - w_{14}$	(13:00 cover)
	$1x_{11} + 1x_{12} + 1x_{13} + 0.8z_{11} + 0.8z_{12} + 0.8z_{13} + 0.8z_{14}$	$\geq 20 + w_{14} - w_{15}$	(14:00 cover)
	$1x_{12} + 1x_{13} + 0.8z_{12} + 0.8z_{13} + 0.8z_{14} + 0.8z_{15}$	$\geq 25 + w_{15} - w_{16}$	(15:00 cover)
	$1x_{11} + 1x_{13} + 0.8z_{13} + 0.8z_{14} + 0.8z_{15} + 0.8z_{16}$	$\geq 28 + w_{16} - w_{17}$	(16:00 cover)
	$1x_{11} + 1x_{12} + 0.8z_{14} + 0.8z_{15} + 0.8z_{16} + 0.8z_{17}$	$\geq 32 + w_{17} - w_{18}$	(17:00 cover)
	$1x_{11} + 1x_{12} + 1x_{13} + 0.8z_{15} + 0.8z_{16} + 0.8z_{17} + 0.8z_{18}$	$\geq 50 + w_{18} - w_{19}$	(18:00 cover)
	$1x_{11} + 1x_{12} + 1x_{13} + 0.8z_{16} + 0.8z_{17} + 0.8z_{18}$	$\geq 30 + w_{19} - w_{20}$	(19:00 cover)
	$1y_{11} + 1x_{12} + 1x_{13} + 0.8z_{17} + 0.8z_{18}$	$\geq 20 + w_{20} - w_{21}$	(20:00 cover)
	$1y_{12} + 1x_{13} + 0.8z_{18}$	$\geq 8 + w_{21}$	(21:00 cover)
	all variables $w_h \leq 20$		
	all variables $w_h, x_h, y_h, z_h \geq 0$		

Once again we have a fractional solution that certainly must be implemented in whole numbers of employees. Managers will need to round above LP-optimal values to obtain a satisfactory plan. Still, any loss of optimality resulting from rounding will fall well within the variability of hourly check arrivals and other data. Unless the numbers of persons working shifts are in the range of, say, 0 to 2, our LP model (4.7) is a valid approximation justified by its outstanding tractability.

EXAMPLE 4.6: FORMULATING SHIFT SCHEDULING LPS

Clerical employees of a government agency are allowed to work four 10-hour days per week in any of the following patterns:

$j = 1$	Monday–Wednesday–Thursday–Friday
$j = 2$	Monday–Tuesday–Thursday–Friday
$j = 3$	Monday–Tuesday–Wednesday–Friday

Formulate a linear program to determine the minimum number of employees needed to have at least 10 on duty Mondays, 9 in the office on Fridays, and 7 working on Tuesdays through Thursdays.

Solution: We employ decision variables

$$x_j \triangleq \text{number of employees working pattern } j$$

The required LP model is then

$$\begin{aligned}
 \min \quad & x_1 + x_2 + x_3 && \text{(total staff)} \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 10 && \text{(cover Monday)} \\
 & x_2 + x_3 \geq 7 && \text{(cover Tuesday)} \\
 & x_1 + x_3 \geq 7 && \text{(cover Wednesday)} \\
 & x_1 + x_2 \geq 7 && \text{(cover Thursday)} \\
 & x_1 + x_2 + x_3 \geq 9 && \text{(cover Friday)} \\
 & x_1, x_2, x_3 \geq 0 &&
 \end{aligned}$$

The objective minimizes total staff, and the constraints enforce the specified coverage on all working days.

4.5 TIME-PHASED MODELS

So far in this chapter we have formulated only **static models**—those where all planning is for a single period of time. Many, perhaps most, linear programs are **dynamic** or **time phased** because they address circumstances that vary over time. In this section we introduce time-phased modeling.

APPLICATION 4.6: INSTITUTIONAL FOOD SERVICES (IFS) CASH FLOW

LP models of almost any type may require time-phased decision making, but some of the most obviously time dependent involve **cash flow** management.⁶ Every business must keep track of the coming and going of its cash accounts, borrowing where necessary and investing when wise.

We illustrate the modeling issues with a fictional Institutional Food Services (IFS) company that supplies food and other products to restaurants, schools, and similar institutions. Table 4.8 shows IFS's projections of some relevant accounts over the next 8 weeks (in thousands of dollars).

$s_t \triangleq$ projected revenue in week t from cash sales to small customers

$r_t \triangleq$ projected accounts receivable revenue received in week t from large customers who buy on credit

$p_t \triangleq$ projected accounts payable to IFS's suppliers in week t

$e_t \triangleq$ projected payroll, utility, and other expenses to be paid in week t

Cash sales and accounts receivable produce immediate income to IFS's checking account. Expenses are immediate deductions. Accounts payable amounts p_t are not actually due until week $t + 3$, but they are discounted by 2% if paid early in week t .

Values in Table 4.8 vary dramatically over the period as a holiday approaches. Besides the option on accounts payable, IFS's financial officer has two additional ways of dealing with the implied cash flow difficulties. First, the company's bank has extended a \$4 million line of credit that may be drawn upon at 0.2% interest per week. However, the bank requires at least 20% of the current amount borrowed to be maintained (without earning interest) in IFS's checking account. The other option is investment of excess cash in short-term money markets. IFS can earn 0.1% interest per week on amounts invested in this way.

The financial officer wishes to minimize net total cost in interest and lost discounts while maintaining at least a \$20,000 checking account safety balance. Our task is to help him decide how to exercise the available options.

TABLE 4.8 IFS Cash Flow Example Data

Item	Projected Weekly Amount (\$ 000's) for Week:							
	1	2	3	4	5	6	7	8
Cash Sales, s_t	600	750	1200	2100	2250	180	330	540
Accounts receivable, r_t	770	1260	1400	1750	2800	4900	5250	420
Accounts payable, p_t	3200	5600	6000	480	880	1440	1600	2000
Expenses, e_t	350	400	550	940	990	350	350	410

Time-Phased Decision Variables

Time is always an index dimension in time-phased models because both input constants and decisions may be repeated in each time period. For our IFS example,

⁶Based on A. A. Robichek, D. Teichroew, and J. M. Jones (1965), "Optimal Short Term Financing Decision," *Management Science*, 12, 1–36.

time is the only index dimension. Decision variables for the three cash flow management options are (in thousands of dollars)

- $g_t \triangleq$ amount borrowed in week t against the line of credit
- $h_t \triangleq$ amount of line of credit debt paid off in week t
- $w_t \triangleq$ amount of accounts payable in week t delayed until week $t + 3$ at a loss of discounts
- $x_t \triangleq$ amount invested in short-term money markets during week t

For modeling convenience, we also define

- $y_t \triangleq$ cumulative line of credit debt in week t
- $z_t \triangleq$ cash on hand during week t

These variables could be eliminated by substituting suitable sums of the others, but many constraints are much easier to express when the extra variables are included.

Time-Phased Balance Constraints

Although separate decisions may be made in each period of a time-phased model, choices for different periods are rarely independent. Decisions in one period usually imply consequences that carry over into the next.

Such interactions among decisions for different time periods can often be modeled with balance constraints similar to those of definition [4.7].

Principle 4.9 | Time-phased models often link decisions in successive time periods with **balance constraints** of the form

$$\begin{pmatrix} \text{starting} \\ \text{level in} \\ \text{period } t \end{pmatrix} + \begin{pmatrix} \text{impacts of} \\ \text{period } t \\ \text{decisions} \end{pmatrix} = \begin{pmatrix} \text{starting} \\ \text{level in} \\ \text{period } t + 1 \end{pmatrix}$$

tracking commodities carried over from each period t to the next.

In our IFS example there are two main quantities carried over in this way: cash and debt. To develop the required balance constraints, we first enumerate the cash increments and decrements each week:

Cash Increments	Cash Decrements
Funds borrowed in week t	Borrowing paid off in week t
Investment principal from week $t - 1$	Investment in week t
Interest on investment in week $t - 1$	Interest on debt in week $t - 1$
Cash sales from week t	Expenses paid in week t
Accounts receivable for week t	Accounts payable paid with discount for week t
	Accounts payable paid without discount for week $t - 3$

Using the symbols defined above, these increments and decrements lead to the following system of balance constraints:

$$z_{t-1} + g_t - h_t + x_{t-1} - x_t + 0.001x_{t-1} - 0.002y_{t-1} + s_t - e_t + r_t - 0.98(p_t - w_t) - w_{t-3} = z_t \quad t = 1, \dots, 8 \quad (\text{cash balance})$$

(All symbols with subscripts outside the range $1, \dots, 8$ are assumed $= 0$.)

A similar constraint system tracks cumulative debt. New borrowing increases, and paying off decreases:

$$y_{t-1} + g_t - h_t = y_t \quad t = 1, \dots, 8 \quad (\text{debt balance})$$

EXAMPLE 4.7: FORMULATING BALANCE CONSTRAINTS OVER TIME

An LP model will decide

$x_q \triangleq$ thousands of snow shovels produced in quarter q

$i_q \triangleq$ thousands of snow shovels held in inventory at the end of quarter q

to meet customer demands for 11,000, 48,000, 64,000, and 15,000 shovels in quarters $q = 1, \dots, 4$, respectively. Write balance constraints in shovels for the four quarters assuming inventory at the beginning of the first quarter $= 0$.

Solution: Following principle [4.9](#), the constraints will have the form

$$(\text{beginning inventory}) + (\text{production}) = (\text{demand}) + (\text{ending inventory})$$

Now taking initial inventory $= 0$, we have

$$0 + x_1 = 11 + i_1 \quad (\text{quarter 1})$$

$$i_1 + x_2 = 48 + i_2 \quad (\text{quarter 2})$$

$$i_2 + x_3 = 64 + i_3 \quad (\text{quarter 3})$$

$$i_3 + x_4 = 15 + i_4 \quad (\text{quarter 4})$$

IFS Cash Flow Model

We are now ready to state a full linear programming model for our IFS case:

$$\begin{aligned} \min \quad & 0.002 \sum_{t=1}^8 y_t + 0.02 \sum_{t=1}^8 w_t - 0.001 \sum_{t=1}^8 x_t && (\text{net interest}) \\ \text{s.t.} \quad & z_{t-1} + g_t - h_t + x_{t-1} - x_t \\ & \quad + 0.001x_{t-1} - 0.002y_{t-1} + s_t - e_t \\ & \quad + r_t - 0.98(p_t - w_t) - w_{t-3} = z_t && t = 1, \dots, 8 \quad (\text{cash balance}) \\ & y_{t-1} + g_t - h_t = y_t && t = 1, \dots, 8 \quad (\text{debt balance}) \quad (4.18) \\ & y_t \leq 4000 && t = 1, \dots, 8 \quad (\text{credit limit}) \\ & z_t \geq 0.20y_t && t = 1, \dots, 8 \quad (\text{bank rule}) \\ & w_t \leq p_t && t = 1, \dots, 8 \quad (\text{payables limit}) \\ & z_t \geq 20 && t = 1, \dots, 8 \quad (\text{safety balance}) \\ & g_t, h_t, w_t, x_t, y_t, z_t \geq 0 && t = 1, \dots, 8 \quad (\text{variable type}) \end{aligned}$$

TABLE 4.9 IFS Cash Flow Optimal Solution

Decision Variable	Optimal Weekly Amount (\$ 000's) for Week							
	1	2	3	4	5	6	7	8
Borrowing, g_t	100.0	505.7	3394.3	0.0	442.6	0.0	0.0	0.0
Debt payment, h_t	0.0	0.0	0.0	442.5	0.0	2715.3	1284.7	0.0
Payables delayed, w_t	2077.6	3544.5	1138.5	0.0	0.0	0.0	0.0	0.0
Short-term investments, x_t	0.0	0.0	0.0	0.0	0.0	0.0	2611.7	1204.3
Cumulative debt, y_t	100.0	605.7	4000.0	3557.4	4000.0	1284.7	0.0	0.0
Cumulative cash, z_t	20.0	121.1	800.0	711.5	800.0	256.9	20.0	20.0

The objective function minimizes interest paid, plus discounts lost, less interest earned. Besides the two systems of balance requirements formulated above, constraints enforce the credit limit and the bank rule requiring that 20% of borrowed funds be kept as cash, keep delayed accounts payable within the value from Table 4.8, and ensure a continuing safety balance of \$20,000. All symbols with subscripts outside the range $1, \dots, 8$ are assumed = 0.

Table 4.9 presents an optimal solution. The corresponding optimal net interest and discounts total \$158,492 for the 8 weeks.

Time Horizons

A **time horizon** establishes the range of time periods in a time-phased model. For example, our IFS cash flow example uses a **fixed time horizon** of $1, \dots, 8$ because we model only 8 weeks.

Of course, IFS would have been operating before the current 8-week period and will continue operations after it. Thus our use of a fixed time horizon raises some special concerns about periods near the boundary.

In particular, model (4.18) assumes that all quantities outside the time horizon equal zero. Thus IFS begins week 1 with zero cash balance z_0 , zero debt y_0 , and zero short-term investments x_0 . Optimal results in Table 4.9 could change dramatically if some of these boundary values proved to be nonsensical. To obtain a more valid model, it might be necessary to estimate typical values and include them as constants in balance equations for week 1.

At the other end of the time horizon we have similar issues. Although the optimum in Table 4.9 chose not to delay accounts payable in the last few weeks, it might have been severely tempted. Payables delayed in the last 3 weeks never have to be paid because the due date ($t + 3$) falls beyond the time horizon.

Such issues do require particular attention if models with time horizons are to produce valid results.

Principle 4.10 Use of fixed time horizons, although necessary in most time-phased models, requires extra care in modeling and interpreting phenomena near both ends of the time epoch being modeled.

One way to avoid having to think about boundaries of a fixed time horizon is to employ an **infinite time horizon** model. Infinite horizon schemes “wrap around”

output states of a last time period as input conditions for the first. The result is that time goes on infinitely even though only a few periods are modeled explicitly.

Principle 4.11 Infinite horizon modeling can avoid some of the boundary difficulties with finite horizons by treating the first explicitly modeled period as coming immediately after the last.

Infinite horizon modeling of our IFS example would treat $t = 1$ as the period immediately after week $t = 8$. Then, for example, the debt balance constraint for $t = 1$ would read

$$y_8 + g_1 - h_1 = y_1$$

EXAMPLE 4.8: MODELING WITH TIME HORIZONS

Return to the snow shovel problem of Example 4.7, and write balance constraints for the four quarters under each of the following assumptions about inventory at time horizon boundaries.

- (a) The time horizon is a fixed four quarters, with beginning inventory in the first quarter of 9000 shovels.
- (b) The time horizon is infinite, with quarter 1 following quarter 4.

Solution:

- (a) With initial inventory = 9, the required balance constraints are

$$9 + x_1 = 11 + i_1 \quad (\text{quarter 1})$$

$$i_1 + x_2 = 48 + i_2 \quad (\text{quarter 2})$$

$$i_2 + x_3 = 64 + i_3 \quad (\text{quarter 3})$$

$$i_3 + x_4 = 15 + i_4 \quad (\text{quarter 4})$$

- (b) With inventory wrapped around from the last to the first quarter, the balance constraints are

$$i_4 + x_1 = 11 + i_1 \quad (\text{quarter 1})$$

$$i_1 + x_2 = 48 + i_2 \quad (\text{quarter 2})$$

$$i_2 + x_3 = 64 + i_3 \quad (\text{quarter 3})$$

$$i_3 + x_4 = 15 + i_4 \quad (\text{quarter 4})$$

4.6 MODELS WITH LINEARIZABLE NONLINEAR OBJECTIVES

Because LP models possess all the tractable features explored in Chapter 3, a linear programming model of a problem is almost always preferable to a nonlinear one of equal validity (principle [2.31](#) of Section 2.4). Nonlinearity is often unavoidable, but some frequently occurring nonlinear objective functions are exceptions.

We introduce in this section those minimax, maximin, and min deviation objective functions which though nonlinear at first glance can be modeled with a linear objective function and linear constraints. Interested readers may also wish to refer to the related discussion of separable nonlinear programming in Section 17.9.

APPLICATION 4.7: HIGHWAY PATROL

We begin with a real allocation problem addressed by the Highway Patrol of a southern state.⁷ The Patrol wished to divide the effort of its on-duty officers among highway segments in each territory to maximize speeding reduction.

The first two lines of Table 4.10 illustrate the types of data available. Highway segments in our fictitious version are indexed by

$$j \triangleq \text{highway segment number } (j = 1, \dots, 8)$$

with 25 officers per week to allocate. Analysts were able to estimate for each segment:

$u_j \triangleq$ upper bound on the number of officers assigned to segment j per week

$r_j \triangleq$ reduction potential for suppressing speeding on segment j per officer assigned

A high reduction potential indicates a segment where a patrol would be especially effective. In the real application, reduction potentials were obtained by directly measuring segment traffic speeds with and without an officer on patrol.

TABLE 4.10 Highway Patrol Example Data and Solutions

	Values by Highway Segment, j							
	1	2	3	4	5	6	7	8
Upper bound, u_j	4	8	5	7	6	5	6	4
Reduction potential, r_j	11	3	4	14	2	19	10	13
Maxisum optimum, x_j^*	4.00	0.00	0.00	7.00	0.00	5.00	5.00	4.00
Maximin optimum, x_j^*	1.09	4.00	3.00	0.86	6.00	4.85	1.20	4.00

Maxisum Highway Patrol Application Model

It is obvious that decision variables in our Highway Patrol allocation example should be

$$x_j \triangleq \text{number of officers per week assigned to patrol segment } j$$

Then a straightforward linear programming model is

$$\begin{aligned}
 \max \quad & \sum_{j=1}^8 r_j x_j && \text{(total reduction)} \\
 \text{s.t.} \quad & \sum_{j=1}^8 x_j \leq 25 && \text{(officers available)} \\
 & x_j \leq u_j \quad j = 1, \dots, 8 && \text{(upper bounds)} \\
 & x_j \geq 0 \quad j = 1, \dots, 8 && \text{(nonnegativity)}
 \end{aligned} \tag{4.19}$$

⁷Based on D. T. Phillips and G. L. Hogg (1979), "The Algorithm That Converged Too Fast," *Interfaces*, 9:5, 90–93.

The objective function is a **maxisum** because it seeks to maximize the sum of reductions in different segments (the analog in a minimize model is called **minisum**). Main constraints restrict solutions to the 25 available officers and enforce upper bounds. The third line of Table 4.10 shows an optimal solution to maxisum model (4.19) that yields a total speed reduction of 339.

Minimax and Maximin Objective Functions

Notice that all but one of the maxisum optimal values in Table 4.10 are either zero or upper bound u_j . A little contemplation will reveal that this must always happen in a maxisum model with constraints as simple as those of (4.19).

Sometimes we would prefer a minimax or a maximin objective to spread the allocation more evenly.

Definition 4.12 **Minimax** (minimize the maximum) or **maximin** (maximize the minimum) objective functions model cases where success is measured by worst rather than total performance.

Instead of optimizing total output, we focus on the model element with the least satisfactory result.

Nonlinear Maximin Highway Patrol Application Model

Adopting the maximin approach in our Highway Patrol example yields the model

$$\begin{aligned}
 \max \quad & f(x_1, \dots, x_8) \triangleq \min\{r_j x_j : j = 1, \dots, 8\} && \text{(maximin reduction)} \\
 \text{s.t.} \quad & \sum_{j=1}^8 x_j \leq 25 && \text{(officers available)} \\
 & x_j \leq u_j \quad j = 1, \dots, 8 && \text{(upper bounds)} \\
 & x_j \geq 0 \quad j = 1, \dots, 8 && \text{(nonnegativity)}
 \end{aligned} \tag{4.20}$$

The objective now maximizes the least reduction among all highway segments.

Notice that (4.20) is a nonlinear program (definition [2.14](#)). Constraints remain linear, but the objective function is no longer a weighted sum of the decision variables. Still, this NLP may provide more valid results than maxisum LP (4.19) because speed reduction is addressed on every highway segment. The final line of Table 4.10 shows that the optimal allocation in this maximin model is much more uniform across segments. The specified optimum yields a reduction of at least 12 on every segment.

Linearizing Minimax and Maximin Objective Functions

With a model as simple as (4.20), nonlinearity may not produce much loss of tractability. Happily, we need not sacrifice tractability even in much more complicated cases. By a suitable modification of nonlinear form (4.20), we can formulate an exactly equivalent linear program.

The idea is simply to introduce a new continuous variable

$$f \triangleq \text{objective function value}$$

and maximize f subject to a system of linear constraints keeping f no more than any term of the minimum.

Principle 4.13 | A minimax or maximin objective function can be modeled linearly by introducing a new decision variable f to represent the objective function value, and then minimizing f subject to $f \geq$ each max element in a minimax, or maximizing f subject to $f \leq$ each min element in a maximin.

Linearized Maximin Highway Patrol Example Model

Applying principle [4.13](#) to the maximin version of our Highway Patrol model yields linear program

$$\begin{aligned}
 \max \quad & f && \text{(maximin reduction)} \\
 \text{s.t.} \quad & f \leq r_j x_j && j = 1, \dots, 8 \quad f \leq \text{each term} \\
 & \sum_{j=1}^8 x_j \leq 25 && \text{(officers available)} \quad (4.21) \\
 & x_j \leq u_j && j = 1, \dots, 8 \quad \text{(upper bounds)} \\
 & x_j \geq 0 && j = 1, \dots, 8 \quad \text{(nonnegativity)}
 \end{aligned}$$

Unrestricted variable f is now the only term of the objective function, which makes the objective trivially linear. New linear constraints keep f less than or equal to all terms $r_j x_j$.

Transformation [4.13](#) works because any optimal solution in (4.21) must have

$$f^* = \min \{r_j x_j^* : j = 1, \dots, 8\}$$

The new system of constraints keeps

$$f^* \leq \min \{r_j x_j^* : j = 1, \dots, 8\}$$

and an f strictly less than the minimum $r_j x_j^*$ can be increased to improve the objective.

EXAMPLE 4.9: MODELING MINIMAX OBJECTIVE FUNCTIONS

In terms of decision variables $x_1, x_2,$ and $x_3,$ the production times required on a company’s two assembly lines are

$$3x_1 + 2x_2 + 1x_3 \quad \text{and} \quad 1x_1 + 5x_2$$

Assuming that all other constraints are linear, formulate an objective function and extra constraints needed to minimize the maximum time on the two lines as a linear program.

Solution: Following principle [4.13](#) for a minimax case, we introduce new unrestricted variable f and employ objective function

$$\min f$$

To make this minimize the maximum, we also add constraints

$$\begin{aligned}
 f &\geq 3x_1 + 2x_2 + 1x_3 \\
 f &\geq 1x_1 + 5x_2
 \end{aligned}$$

APPLICATION 4.8: VIRGINIA PRESTRESS (VP) LOCATION

To see another common nonlinear objective that can be linearized, we consider the location problem confronted by a firm we will call Virginia Prestress (VP).⁸ VP was planning for production of a new product: concrete utility poles. That production required a new concrete casting area to make poles and a storage area for finished products.

Figure 4.2 presents the implied facilities location problem for our fictitious case. The two new facilities will interact with each other and with three existing operations: the concrete batching facility, where premixed concrete is prepared, the steel area, where reinforcing steel is manufactured, and the shipping gate, where finished poles are processed out of the plant. A coordinate system quantifies the locations of all three existing facilities, and the adjoining table displays material handling costs of expected traffic between facilities. For example, each foot of distance between the pole storage area and the shipping gate adds \$0.40 in crane activity. We must choose locations for the new facilities to minimize total material handling cost.

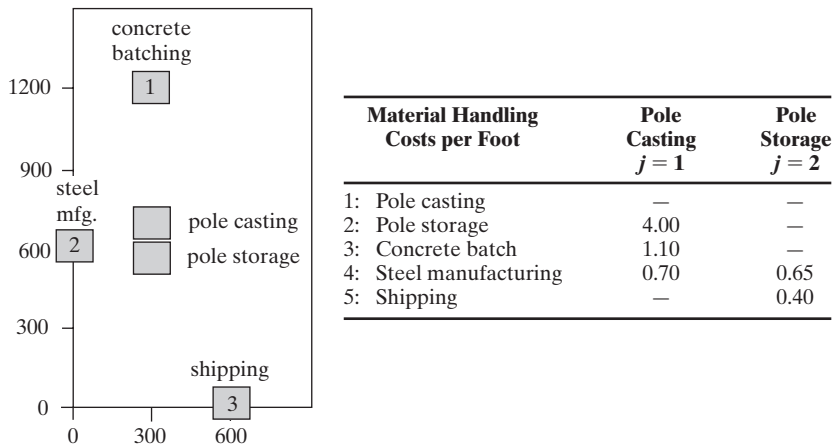


FIGURE 4.2 Virginia Prestress (VP) Location Example

Nonlinear VP Location Model

It should be clear that the main decisions to be made in this VP case are

$x_j \triangleq$ x -coordinate of new facility j 's location

$y_j \triangleq$ y -coordinate of new facility j 's location

We want to choose $x_1, y_1, x_2,$ and y_2 to minimize the sum of implied distances to other facilities times unit material handling costs. Using values and coordinates in Figure 4.2, a model is

$$\begin{aligned} \min \quad & 4.00d(x_1, y_1, x_2, y_2) + 1.10d(x_1, y_1, 300, 1200) \\ & + 0.70d(x_1, y_1, 0, 600) + 0.65d(x_2, y_2, 0, 600) \quad (\text{handling cost}) \quad (4.22) \\ & + 0.40d(x_2, y_2, 600, 0) \end{aligned}$$

⁸Based on R. F. Love and L. Yerex (1976), "An Application of a Facilities Location Model in the Prestressed Concrete Industry," *Interfaces*, 6:4, 45–49.

where

$$d(x_j, y_j, x_k, y_k) \triangleq \text{distance from } (x_j, y_j) \text{ to } (x_k, y_k)$$

No constraints are required.

If we measure distance in the straight-line or **Euclidean** way depicted in Figure 4.3(a), model (4.22) is unavoidably nonlinear.⁹ However, it is often more appropriate in facilities design to calculate distance in the **rectilinear** manner of Figure 4.3(b). Material movements tend to follow aisles and other paths aligned with either the x or the y axis. Thus travel distance is best modeled as

$$d(x_j, y_j, x_k, y_k) \triangleq |x_j - x_k| + |y_j - y_k| \quad (4.23)$$

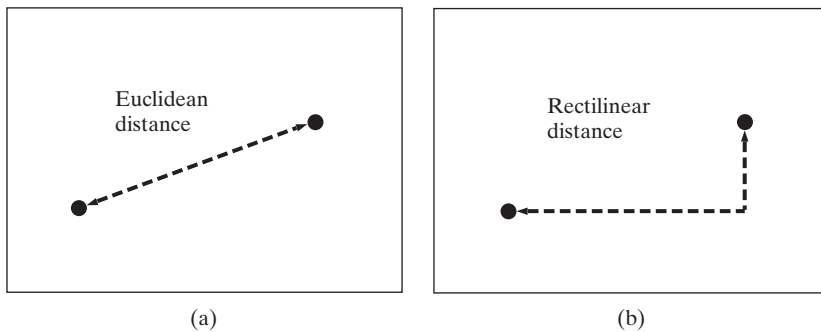


FIGURE 4.3 Euclidean versus Rectilinear Distance

Min Deviation Objective Functions

With rectilinear distance measure (4.23), mathematical program (4.22) takes a min deviation form.

Definition 4.14 | **Min deviation** objective functions model cases where the goal is to minimize positive-weighted sums of absolute differences between pairs of model quantities.

Here we seek to minimize the cost-weighted sum of location coordinate differences.

Linearizing Min Deviation Objective Functions

Any min deviation objective involving positive-weighted absolute differences of linear functions can be modeled linearly. We need only introduce new deviation variables expressing the required differences.

⁹See, for example, the DClub model of Section 3.1.

Principle 4.15 Positive-weighted terms $|p(\mathbf{x}) - q(\mathbf{x})|$ of a min deviation objective function involving differences of linear functions $p(\mathbf{x})$ and $q(\mathbf{x})$ can be modeled linearly by (1) introducing new nonnegative **deviation variables** s^+ and s^- , (2) adding new constraints

$$p(\mathbf{x}) - q(\mathbf{x}) = s^+ - s^-$$

and (3) substituting $s^+ + s^-$ for the absolute differences in the objective function.

Linearized VP Location Model

With distance rectilinear, our VP location model (4.22) becomes

$$\begin{aligned} \min \quad & 4.00|x_1 - x_2| + 4.00|y_1 - y_2| + 1.10|x_1 - 300| \\ & + 1.10|y_1 - 1200| + 0.70|x_1 - 0| + 0.70|y_1 - 600| \\ & + 0.65|x_2 - 0| + 0.65|y_2 - 600| + 0.40|x_2 - 600| \\ & + 0.40|y_2 - 0| \end{aligned}$$

To apply principle [4.15](#), we introduce a pair of deviation variables for each objective function term $i = 1, \dots, 10$:

$s_i^+ \triangleq$ positive difference in absolute value term i

$s_i^- \triangleq$ negative difference in absolute value term i

Then we may solve VP's location problem with linear program

$$\begin{aligned} \min \quad & 4.00(s_1^+ + s_1^-) + 4.00(s_2^+ + s_2^-) + 1.10(s_3^+ + s_3^-) \\ & + 1.10(s_4^+ + s_4^-) + 0.70(s_5^+ + s_5^-) + 0.70(s_6^+ + s_6^-) \\ & + 0.65(s_7^+ + s_7^-) + 0.65(s_8^+ + s_8^-) + 0.40(s_9^+ + s_9^-) \\ & + 0.40(s_{10}^+ + s_{10}^-) \\ \text{s.t.} \quad & x_1 - x_2 = s_1^+ - s_1^- \quad (\text{term 1}) \\ & y_1 - y_2 = s_2^+ - s_2^- \quad (\text{term 2}) \\ & x_1 - 300 = s_3^+ - s_3^- \quad (\text{term 3}) \\ & y_1 - 1200 = s_4^+ - s_4^- \quad (\text{term 4}) \\ & x_1 - 0 = s_5^+ - s_5^- \quad (\text{term 5}) \\ & y_1 - 600 = s_6^+ - s_6^- \quad (\text{term 6}) \\ & x_2 - 0 = s_7^+ - s_7^- \quad (\text{term 7}) \\ & y_2 - 600 = s_8^+ - s_8^- \quad (\text{term 8}) \\ & x_2 - 600 = s_9^+ - s_9^- \quad (\text{term 9}) \\ & y_2 - 0 = s_{10}^+ - s_{10}^- \quad (\text{term 10}) \\ & s_i^+, s_i^- \geq 0 \quad i = 1, \dots, 10 \end{aligned} \tag{4.24}$$

Notice that new (linear) constraints express each absolute difference term of the min deviation objective function as the difference of corresponding deviation

variables subject to nonnegativity constraints. Then the objective function minimizes the weighted sum (not the difference) of those variable pairs.

Here an optimal solution locates the new facilities next to each other at

$$\begin{aligned} x_1^* &= x_2^* = 300 \\ y_1^* &= y_2^* = 600 \end{aligned}$$

Corresponding optimal values for deviation variables are

$$\begin{aligned} s_1^{+*} &= 0, & s_2^{+*} &= 0, & s_3^{+*} &= 0, & s_4^{+*} &= 0, & s_5^{+*} &= 300 \\ s_1^{-*} &= 0, & s_2^{-*} &= 0, & s_3^{-*} &= 0, & s_4^{-*} &= 600, & s_5^{-*} &= 0 \\ s_6^{+*} &= 0, & s_7^{+*} &= 300, & s_8^{+*} &= 0, & s_9^{+*} &= 0, & s_{10}^{+*} &= 600 \\ s_6^{-*} &= 0, & s_7^{-*} &= 0, & s_8^{-*} &= 0, & s_9^{-*} &= 300, & s_{10}^{-*} &= 0 \end{aligned}$$

Why does transformation [4.15](#) work? Because it can never be optimal for any term to have both $s_i^+ > 0$ and $s_i^- > 0$. For example, at optimality the fourth term of our VP model (4.24) has the corresponding constraint

$$y_1^* - 1200 = 600 - 1200 = -600 = s_4^+ - s_4^-$$

Many choices of s_4^+ and s_4^- will satisfy the constraint, but the objective function prefers the one with the smallest sum, that is,

$$s_4^{+*} = 0, \quad s_4^{-*} = 600$$

With at most one member of each deviation variable pair being positive in an optimal solution, their difference will always exactly equal the required absolute difference.

EXAMPLE 4.10: MODELING MIN DEVIATION OBJECTIVE FUNCTIONS

In terms of design parameters $x_1, x_2,$ and x_3 the speed and weight of a proposed vehicle can be expressed as

$$4x_1 - x_2 + 7x_3 \quad \text{and} \quad 9x_1 - 10x_2 + x_3$$

respectively. Assuming that all other constraints are linear, formulate an objective function and extra constraints needed in a linear program to find the design with speed as close as possible to 100, and weight as close as possible to 150.

Solution: Following principle [4.15](#), we define deviation variables s_1^+ and s_1^- for speed, together with s_2^+ and s_2^- for weight. Then we minimize the deviation of speed and weight from their desired values with objective function

$$\min \quad s_1^+ + s_1^- + s_2^+ + s_2^- \quad (\text{total deviation})$$

and special constraints

$$4x_1 - x_2 + 7x_3 - 100 = s_1^+ - s_1^- \quad (\text{speed})$$

$$9x_1 - 10x_2 + x_3 - 150 = s_2^+ - s_2^- \quad (\text{weight})$$

$$s_1^+, s_1^-, s_2^+, s_2^- \geq 0$$

4.7 STOCHASTIC PROGRAMMING

Section 1.6 introduced the distinction between deterministic and stochastic optimization models. For convenience and tractability, **deterministic models** assume all model parameters are known with certainty even though they are truly only estimates of the values that will arise in real application. **Stochastic models** explicitly include parameters known only in probability, that is, **random variables** for which a probability distribution of possible parameter **realizations** is known but variability of possible values must be modeled to validly choose best values for the decision variables of the optimization.

The overwhelming majority of this book addresses only the deterministic form, leaving stochastic models to other sources. Still, a small collection of stochastic cases fit neatly into the paradigms of deterministic optimization, and those can be included. We introduce the LP version of one of them here – **two-stage stochastic programming** also known as **stochastic programming with recourse**.

Definition 4.16 | An optimization model is a two-stage stochastic program if the decisions to be made can be divided into two phases or **stages**. The first considers the decisions to be made before the values of any modeled random variables are known, and the second provides response or **recourse** after stochastic parameter values have been realized.

A small fictional example will illustrate.

APPLICATION 4.9: QUICK AID (QA)

The Quick Aid (QA) division of the Emergency Management Agency is establishing a network to rapidly distribute vitally needed first-aid and food supplies in survival kits sized for single individuals when some part of the U.S. Gulf coast has been struck by a hurricane. Kits will be inventoried in advance at one or more of 3 depot sites leased within existing commercial warehouses. Then, as soon as a storm hits, needed kits will be transported rapidly to any or all of the 4 coastal regions where major populations have been dislocated.

QA wishes to find a minimum total cost plan for dealing with the 4 storm emergencies typically occurring per year. Kits inventories for up to a total of 1 million victims will be held in depots then shipped as needed to impacted regions. Cost (per kit) for annual holding inventory at each potential depot site, and for shipping from each site to each potentially stricken region have been estimated as shown in the following:

Depot	Holding Cost c_i	Transportation Costs d_{ij}			
		$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	7.20	2.25	3.45	5.52	5.00
$i = 2$	20.00	5.95	2.14	3.64	4.15
$i = 3$	8.00	5.90	3.95	4.10	4.00

The number of kits that will need to be shipped to any region depends on the scope of the storm emergency. Careful analysis storm history over the past 20 years has

identified $n = 8$ **scenarios** s according to the regions of the storm focus. The fraction of storms fitting in each p_s , and the associated requirements (in 000s) $r_j^{(s)}$ have been estimated as shown in the following:

Scenario	Fraction	Regional Requirements			
		$j = 1$	$j = 2$	$j = 3$	$j = 4$
$s = 1$ (R1 only)	0.10	100	10	—	—
$s = 2$ (R2 only)	0.02	10	720	20	—
$s = 3$ (R3 only)	0.16	—	16	270	11
$s = 4$ (R4 only)	0.14	—	—	20	77
$s = 5$ (R1 & R2)	0.08	90	675	20	—
$s = 6$ (R2 & R3)	0.20	10	675	220	11
$s = 7$ (R3 & R4)	0.24	—	10	220	69
$s = 8$ (All)	0.06	90	675	220	11
	Average	24.8	249.9	158.0	32.0

EXAMPLE 4.11: RECOGNIZING 2-STAGE STOCHASTIC SETTINGS

Consider a beachfront vendor with dispensing tanks for 2 types of drinks. Each evening the vendor must decide how many tanks to pre-order for the next morning filled with lemonade, and how many filled with coffee. One tank will also be left empty for filling the next morning, but no other ordering is possible. If the new day proves sunny, which happens 70% of the time, there will be demand for 3 tanks of lemonade and 1 of coffee. If it is cold, which happens 30% of the time, demands will be for 1 tank of lemonade and 3 of coffee. The vendor would like to maximize the expected total tanks of liquid sold.

- (a) Explain how this vendor’s decisions can be viewed as a two-stage stochastic program with recourse.
- (b) Identify what scenarios will be considered in Stage 2.

Solution:

- (a) This is a two-stage stochastic programming setting, because a choice on the mix of the first 3 tanks must be made when next day’s weather is known only in probability. Then, disposition of the 4th tank is left to recourse after the weather has been realized.
- (b) The two scenarios are 1 = sun and 2 = cold.

Deterministic Model of QA Example

To begin, consider formulating a deterministic model of the QA example more in the spirit of others in this chapter. Using decision variables $x_i \triangleq$ the number of kits (in thousands) inventoried at depot i and $z_{ij} \triangleq$ the number of kits (in thousands)

shipped from depot i to region j in the event of an emergency, we can formulate the task deterministically as

$$\begin{aligned}
 \min \quad & \sum_{i=1}^3 c_i x_i + 4 \sum_{i=1}^3 \sum_{j=1}^4 d_{ij} z_{ij} && \text{(total annual cost)} \\
 \text{s.t.} \quad & \sum_{i=1}^3 x_i \leq 1000 && \text{(up to 1 million)} \\
 & \sum_{i=1}^3 z_{ij} = r_j^{avg} && j = 1, \dots, 4 \quad \text{(average demand)} \\
 & \sum_{j=1}^4 z_{ij} \leq x_i && i = 1, \dots, 3 \quad \text{(depot inventory)} \\
 & x_i \geq 0, \quad i = 1, \dots, 3 \\
 & z_{ij} \geq 0, \quad i = 1, \dots, 3, j = 1, \dots, 4
 \end{aligned}$$

where the r_j^{avg} parameters are the scenario-fraction-weighted average requirements for regions r in the scenario table above. The constant 4 in the objective function reflects planning for 4 events per year.

Such a model plans inventories and shipping for an average of the possible futures rather than addressing recourse to particular scenarios. An optimal solution is

$$\begin{aligned}
 x_i^* &= (274.7, 0.0, 190.0) \\
 z_{ij}^* &= \begin{pmatrix} 24.8 & 249.9 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 158.0 & 32.0 \end{pmatrix} \\
 cost &= \$10,271 \text{ thousand}
 \end{aligned}$$

Notice that the chosen total inventory $274.7 + 190.0 = 464.4$ is not enough to fulfill even region $j = 2$ needs given in the above tables for scenarios $s = 2, 5, 6$ or 8 . Implementing this “optimal” plan could easily lead to a major crisis.

Principle 4.17 | Deterministic optimization for an average outcome in settings with high variability of requirements may lead to optimal solutions with high risk.

Stochastic Programming with Recourse

Stochastic programming LPs with recourse explicitly consider decisions in both optimization stages.

Definition 4.18 | Two-stage stochastic programs refine modeling of uncertain futures by including both the impacts of Stage 1 choices before the future is known, and separate choices explicitly addressing recourse for each possible future scenario.

Figure 4.4 illustrates the idea.

The first set of variables \mathbf{x} , objective function term $\mathbf{c}\mathbf{x}$, and constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ model Stage 1 before stochastic outcomes have been observed. Then for each scenario s , there is another set of recourse variables $\mathbf{z}^{(s)}$ and constraints

$$\mathbf{T}_s \mathbf{x} + \mathbf{R}_s \mathbf{z}^{(s)} \leq \mathbf{r}^{(s)}$$

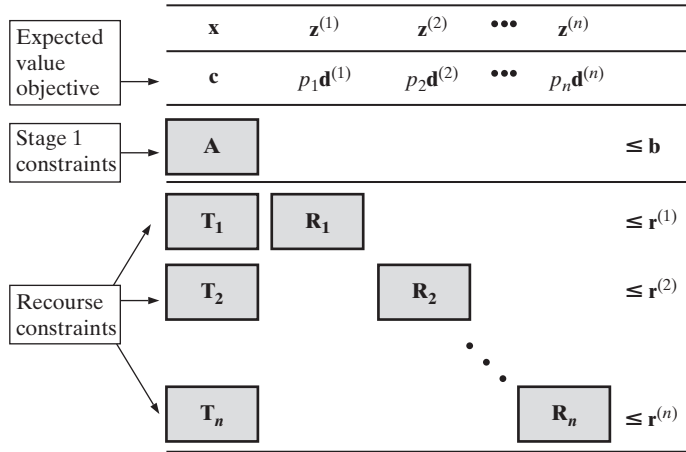


FIGURE 4.4 Extensive Format of 2-Stage Stochastic Programs

with objective function contribution $p_s d^{(s)} z^{(s)}$. Here the p_s are the (nonnegative, sum = 1) probabilities of particular scenarios; including them in the objective makes the complete sum the expected value of possible outcomes. Each T_s transforms Stage 1 variables into how they are reflected in scenario constraints, and together with $R_s z^{(s)} \leq r^{(s)}$ imposes applicable limitations on recourse decisions for the scenario.

Many objective functions can be used in stochastic programs, but expected values are the most common.

Principle 4.19 | Although other possibilities are sometimes used, typical stochastic programming models employ objective functions computing the expected value of choices from both decision stages. This is accomplished by weighted objective function terms for each scenario by the probability of it occurring.

Stochastic Programming Modeling of the QA Application

LP (4.25) is the 2-stage stochastic programming model of QA hurricane response Application 4.9.

$$\begin{aligned}
 \min \quad & \sum_{i=1}^3 c_i x_i + 4 \cdot \sum_{s=1}^8 p_s \left(\sum_{i=1}^3 \sum_{j=1}^4 d_{ij} z_{ij}^s \right) && \text{(expected annual cost)} \\
 \text{s.t.} \quad & \sum_{i=1}^3 x_i \leq 1000 && \text{(up to 1 million)} \\
 & -x_i + \sum_{j=1}^4 z_{ij}^s \leq 0 \quad s = 1, \dots, 8; i = 1, \dots, 3 && \text{(supply)} \\
 & \sum_{i=1}^3 z_{ij}^s = r_j^s \quad s = 1, \dots, 8; j = 1, \dots, 4 && \text{(demand)} \\
 & x_i \geq 0, \quad i = 1, \dots, 3 \\
 & z_{ij}^s \geq 0, \quad i = 1, \dots, 3; \quad j = 1, \dots, 4; \quad s = 1, \dots, 8
 \end{aligned} \tag{4.25}$$

In the schema of Figure 4.4, the Stage 1 rows are a single \leq inequality with

$$\mathbf{A} = [+1 \ +1 \ +1] \quad \text{and} \quad b = 1000$$

Recourse systems for each scenario s use arrays

$$\mathbf{T}_s = \begin{bmatrix} -1 & & & \\ & -1 & & \\ & & -1 & \\ \hline & & & \end{bmatrix} \quad \mathbf{R}_s = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & & & & \\ & & & & 1 & 1 & 1 & 1 \\ \hline & & & & & & & & 1 & 1 & 1 & 1 \\ 1 & & & & 1 & & & & & & & \\ & 1 & & & & 1 & & & & 1 & & \\ & & 1 & & & & 1 & & & & 1 & \\ & & & & & & & 1 & & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & & 1 & & \end{array} \right] \quad \mathbf{r}^{(s)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline r_1^s \\ r_2^s \\ r_3^s \\ r_4^s \end{bmatrix}$$

with the first 3 rows \leq inequalities and the last 4 equalities.

An optimal solution has expected value \$14,192.1 thousand. Decision variables at non-zero optimal values in Stage 1 are $\bar{x}_1 = 715$, $\bar{x}_3 = 281$. Then optimal shipping from the two used depots by scenario is

s	Ship Depot $i = 1$ to Region		Ship Depot $i = 3$ to Region		
	$j = 1$	$j = 2$	$j = 2$	$j = 3$	$j = 4$
1	100	10	0	0	0
2	10	697	23	20	0
3	0	16	0	270	11
4	0	0	0	20	77
5	90	217	58	20	0
6	10	617	0	220	11
7	0	10	0	220	69
8	90	617	58	220	11

EXAMPLE 4.12: FORMULATION OF 2-STAGE BEACH VENDOR
EXAMPLE 4.11

Return to the beach vendor’s problem of Example 4.11, and assume that numbers of tanks can be taken as continuous rather than integer.

- (a) Define decision variables for Stage 1 of the application.
- (b) Define decision variables for both scenarios recourse in Stage 2.
- (c) Formulate a full 2-stage stochastic LP in terms of your variables in parts (a) and (b) that maximizes expected total sales.
- (d) Identify an optimal solution in your model of (c) by inspection.

Solution:

(a) Stage 1 decisions are $x_1 \triangleq$ the number of lemonade tanks pre-ordered, and $x_2 \triangleq$ the number of coffee.

(b) Stage 2 decisions for scenarios $s = 1$, sunny and $s = 2$, cold are $w_1^s \triangleq$ the number of lemonade tanks ordered after weather is known, and $w_2^s \triangleq$ the number of coffee tanks ordered after weather is known. For simplicity we also include intermediate variables $z_1^s \triangleq$ the number of tanks of lemonaid sold on type s days, and $z_2^s \triangleq$ the number of tanks of coffee sold.

(c) A suitable model is

$$\begin{aligned}
 \max \quad & .70(z_1^1 + z_2^1) + .30(z_1^2 + z_2^2) \\
 \text{s.t.} \quad & x_1 + x_2 = 3 \\
 & x_1 + w_1^1 \geq z_1^1 & x_1 + w_1^2 \geq z_1^2 \\
 & x_2 + w_2^1 \geq z_2^1 & x_2 + w_2^2 \geq z_2^2 \\
 & w_1^1 + w_2^1 = 1 & w_1^2 + w_2^2 = 1 \\
 & z_1^1 \leq 3 & z_1^2 \leq 1 \\
 & z_2^1 \leq 1 & z_2^2 \leq 3 \\
 & \text{all variables} \geq 0
 \end{aligned}$$

(d) An optimal choice of Stage 1 variables is $\bar{x}_1 = 2$ and $\bar{x}_2 = 2$. Then, if the day proves sunny, choosing $\bar{w}_1^1 = 1$, $\bar{w}_2^1 = 0$ yields maximal sales $\bar{z}_1^1 = 3$ and $\bar{z}_2^1 = 1$. If the day proves cold, choosing $\bar{w}_1^2 = 0$, $\bar{w}_2^2 = 1$ yields sales $\bar{z}_1^2 = 0$, and $\bar{z}_2^2 = 2$. These lead to solution expected value = $.7(3 + 1) + .3(1 + 2) = 3.7$.

Extensive Form versus Large-Scale Techniques

The 2-stage stochastic programming format of Figure 4.4 is called the **extensive** form of the model because all scenarios and their associated recourse are included explicitly. Many applications involve far too many scenarios to make explicit in this way. Furthermore, many of them have no significant effect of the ultimate optimal solution and expected value. That is why most such applications employ the large-scale techniques of Chapter 13.

Principle 4.20 | The extensive form of Stochastic Programs grows prohibitively large if too many scenarios are possible. In such cases, large-scale techniques like those in Chapter 13 can be employed to achieve tractable models.

EXERCISES

4-1 Bisco’s new sugar-free, fat-free chocolate squares are so popular that the company cannot keep up with demand. Regional demands shown in the following table total 2000 cases per week, but Bisco can produce only 60% of that number.

	NE	SE	MW	W
Demand	620	490	510	380
Profit	1.60	1.40	1.90	1.20

The table also shows the different profit levels per case experienced in the regions due to competition and consumer tastes. Bisco wants to find a maximum profit plan that fulfills between 50 and 70% of each region’s demand.

- (a) Formulate an allocation LP to choose an optimal distribution plan.
- (b) Enter and solve your model with the class optimization software.

4-2 A small engineering consulting firm has 3 senior designers available to work on the firm’s 4 current projects over the next 2 weeks. Each designer has 80 hours to split among the projects, and the following table shows the manager’s scoring (0 = nil to 100 = perfect) of the capability of each designer to contribute to each project, along with his estimate of the hours that each project will require.

Designer	Project			
	1	2	3	4
1	90	80	10	50
2	60	70	50	65
3	70	40	80	85
Required	70	50	85	35

The manager wants to assign designers to maximize total capability.

- (a) Formulate an allocation LP to choose an optimal work assignment.
- (b) Enter and solve your model with the class optimization software.

4-3 Cattle feed can be mixed from oats, corn, alfalfa, and peanut hulls. The following table shows the current cost per ton (in dollars) of each of

these ingredients, together with the percentage of recommended daily allowances for protein, fat, and fiber that a serving of it fulfills.

	Oats	Corn	Alfalfa	Hulls
% Protein	60	80	55	40
% Fat	50	70	40	100
% Fiber	90	30	60	80
Cost	200	150	100	75

We want to find a minimum cost way to produce feed that satisfies at least 60% of the daily allowance for protein and fiber while not exceeding 60% of the fat allowance.

- (a) Formulate a blending LP to choose an optimal feed mix.
- (b) Which of the constraints of your model are composition constraints? Explain.
- (c) Enter and solve your model with the class optimization software.

4-4 Several forms of gasoline are produced during the petroleum refining process, and a last step combines them to obtain market products with specified quality measures. Suppose 4 different gasolines are available, with values for the 2 indexes of quality being 99 and 210, 70 and 335, 78 and 280, and 91 and 265, respectively. Using corresponding costs per barrel of \$48, \$43, \$58, and \$46, we would like to choose a minimum cost blend with a first quality index between 85 and 90 and a second index between 270 and 280.

- (a) Formulate a blending LP to choose an optimal gasoline blend.
- (b) Which of the constraints of your model are composition constraints? Explain.
- (c) Enter and solve your model with the class optimization software.

4-5 Ronnie Runner distilleries blends $i = 1, \dots, m$ scotch whiskeys to create its $j = 1, \dots, n$ products with properties $k = 1, \dots, p$. Unblended whiskey i measures $a_{i,k}$ on scale k . Express each of the following as linear constraint(s) in these parameters and the nonnegative decision variables $x_{i,j} \triangleq$ barrels of whiskey i used in product j . Assume that the properties combine in proportion to volume and

that the total production of any blend is free to vary with the optimization.

- ✓ (a) Property $k = 11$ should fall between 45 and 48 in all products.
- (b) Product $j = 14$ must have all properties $k = 5, \dots, 9$ between 90 and 95.
- ✓ (c) Product $j = 26$ must measure at least 116 on property $k = 15$.
- (d) No product should measure more than 87 on property $k = 8$.
- ✓ (e) Products 6 through 11 should combine input whiskey 1 with others in at most the ratio 3:7.
- (f) Input whiskeys 4 and 7 should be in ratio 2:3 for all blends.
- ✓ (g) At least one-third of all output must come from inputs $i = 3, \dots, 6$.
- (h) No more than 5% of all output can come from input $i = 13$.

4-6 Problems are often modeled as linear programs even though some decision variables represent quantities such as the number of units processed or the number of times an alternative is used that must be integer in a physical implementation. Briefly justify this practice.

4-7 A metalworking shop needs to cut at least 37 large disks and 211 small ones from sheet metal rectangles of a standard size. Three cutting patterns are available. One yields 2 large disks with 34% waste, the second gives 5 small disks with 22% waste, and the last produces 1 large and 3 small disks with 27% waste. The shop seeks a minimum waste way to fulfill its requirements.

- ✓ (a) Formulate an operations management LP to choose an optimal cutting plan.
- ✓ (b) Enter and solve your model with the class optimization software.

4-8 Classic Candles handmakes three models of elegant Christmas candles. Santa models require 0.10 day of molding, 0.35 day of decorating, and 0.08 day of packaging and produce \$16 of profit per unit sold. Corresponding values for Christmas trees are 0.10, 0.15, 0.03, and \$9, while those of gingerbread houses are 0.25, 0.40, 0.05, and \$27. Classic wants to maximize profit on what it makes over the next 20 working days with its 1 molder, 3 decorators, and 1 packager, assuming that everything made can be sold.

- (a) Formulate an operations management LP to choose an optimal production plan.
- ✓ (b) Enter and solve your model with the class optimization software.

4-9 Wobbly Office Equipment (WOE) makes two models of tables for libraries and other university facilities. Both models use the same tabletops, but model A has 4 short (18-inch) legs and model B has 4 longer ones (30-inch). It takes 0.10 labor hour to cut and shape a short leg from stock, 0.15 labor hour to do the same for a long leg, and 0.50 labor hour to produce a tabletop. An additional 0.30 labor hour is needed to attach the set of legs for either model after all parts are available. Estimated profit is \$30 for each model A sold and \$45 for each model B. Plenty of top material is on hand, but WOE wants to decide how to use the available 500 feet of leg stock and 80 labor hours to maximize profit, assuming that everything made can be sold.

- ✓ (a) Formulate an operations management LP to choose an optimal plan using the decision variables $x_1 \triangleq$ number of model A's assembled and sold, $x_2 \triangleq$ number of model B's assembled and sold, $x_3 \triangleq$ number of short legs manufactured, $x_4 \triangleq$ number of long legs manufactured, and $x_5 \triangleq$ number of tabletops manufactured.
- ✓ (b) Which of the constraints of your model are balance constraints? Explain.
- ✓ (c) Enter and solve your model with the class optimization software.

4-10 Perfect Stack builds standard and extralong wooden palettes for a variety of manufacturers. Each model consists of 3 heavy separators of length equal to the palette. The standard model has 5 cross pieces above and 5 below the separators and requires 0.25 hour to assemble. The extralong version has 9 similar cross pieces on top and bottom and consumes 0.30 hour to assemble. The supply of wood is essentially unlimited, but it requires 0.005 hour to fabricate a standard separator, 0.007 hour to fabricate an extralong separator, and 0.002 hour to fabricate a cross piece. Assuming that it can sell as many standard models as can be made at \$5 profit each and as many extralongs at \$7 profit, Perfect wants to decide what to produce with the available 200 hours of assembly time and 40 hours of fabrication.

- (a) Formulate an operations management LP to choose an optimal plan using the decision variables $x_1 \triangleq$ number of standard palettes assembled and sold, $x_2 \triangleq$ number of extralongs assembled and sold, $x_3 \triangleq$ number of standard separators manufactured, $x_4 \triangleq$ number of extralong separators manufactured, and $x_5 \triangleq$ number of cross pieces manufactured.
- (b) Which of the constraints of your model are balance constraints? Explain.
- (c) Enter and solve your model with the class optimization software.

4-11 The figure below shows the Bill of Materials buildup to 2 finished products of bicycle rack manufacturer Hang Up (HU). For example, Product 2 uses one Assembly 3 and four Part 6's. Each Assembly 3, in turn, is made up of 2 Part 5's and 1 Part 7. All Products, Assemblies and Parts are produced at HU's plant.

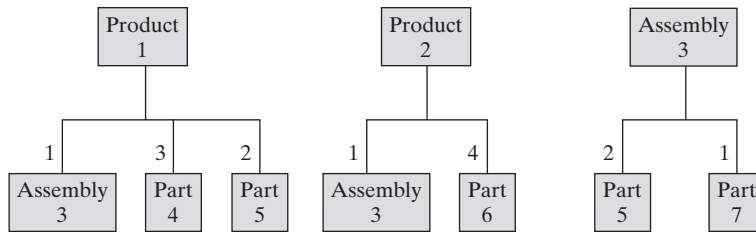


Figure for Exercise 4-11

Using decision variables $x_j \geq 0$, $j = 1, \dots, 7$ for the numbers of objects j (Products, Assemblies, or Parts) produced per week, formulate all material balance constraints required in an LP to assure enough of each object has been manufactured to meet given weekly demands d_1 and d_2 for finished products 1 and 2.

4-12 Goings Engine produces diesel engines and assemblies $i = 1, \dots, m$ at its plants $p = 1, \dots, n$. There is some end demand $d_{i,p}$ for the various engines and assemblies, with the rest used in Goings production. The number of subassemblies i required to produce each assembly k is $a_{i,k}$.

- (a) Write a system of linear constraints specifying that the number of each engine and assembly must balance (with zero inventories) across the company using the parameters above and the nonnegative decision variable $x_{i,p} \triangleq$ number of assemblies i produced at plant p .
- (b) Write a system of linear constraints specifying that the number of each engine and assembly must balance (with zero inventories) at each plant using the foregoing parameters and the nonnegative decision variable $x_{i,p,q} \triangleq$ number of assemblies i produced at plant p for use at plant q .

4-13 The River City Police Department uses work shifts in which officers work 5 of the 7 days of the week with 2 successive days off. For example, a shift might work Sunday through Thursday and then have Friday and Saturday off. A total of 6 officers must be on duty Monday, Tuesday, Wednesday, and Thursday; 10 are required on Friday and Saturday; and 8 are needed on Sunday. River City wants to meet these staffing needs with the minimum total number of officers.

- (a) Formulate a shift scheduling LP to select an optimal staffing plan.
- (b) Which of the constraints of your model are covering constraints? Explain.
- (c) Enter and solve your model with the class optimization software.

4-14 Mama's Kitchen serves from 5:30 A.M. each morning until 1:30 P.M. Tables are set and cleared by busers working 4-hour shifts beginning on the hour from 5 A.M. through 10 A.M. Most are college students who hate to get up in the morning, so Mama's pays \$7 per hour for the 5, 6, and 7 A.M. shifts, and \$6 per hour for all others. The manager seeks a minimum cost staffing plan that will have 2 busers on duty for the hour beginning

at 5 A.M., plus 3, 5, 5, 3, 2, 4, 6 and 3 on duty for the hours to follow.

- (a) Formulate a shift scheduling LP to select an optimal staffing plan.
- (b) Which of the constraints of your model are covering constraints? Explain.
- (c) Enter and solve your model with the class optimization software.

4-15 The MacKensie's daughter will begin college 4 years from today. Her parents want to invest \$10,000 at the beginning of each of the 4 years to accumulate a fund that can help pay the cost. Each year they expect to have available both certificates of deposit returning 5% after 1 year and ones returning 12% after 2 years. This year, they also have an opportunity to make a special investment that would return 21% after 4 years. The MacKensies want to choose investments to maximize their college fund assuming that all funds are reinvested at maturity.

- (a) Formulate a time-phased LP to choose an optimal investment plan.
- (b) Which of the constraints in your model are balance constraints? Explain.
- (c) What is the time horizon of your model? Explain.
- (d) Enter and solve your model with the class optimization software.

4-16 The Big Gear (BG) transmission company buys and distributes replacement transmissions for large, 18-wheeler trucks. For the next 4 months, the company anticipates demands of 100, 130, 95, and 300 units, respectively. During the first month, units can be purchased from BG's supplier at a cost of \$12K each, but thereafter the price goes to \$14K per transmission. One order will be placed at the beginning of each month, and goods arrive immediately. Units can be held in the Big Gear warehouse at a cost of \$1.2K per unit held per month, but there is no starting inventory. Naturally, the company wants to meet demand over the finite 4-month horizon at minimum total cost.

- (a) Briefly justify why appropriate (nonnegative) decision variables for an LP model of BG's problem are numbers of units x_t purchased each month and inventory h_t held during each month t .

- (b) Clearly, BG can only implement solutions involving integer numbers of transmissions. Briefly explain why it still makes sense to represent the numbers of transmissions as continuous variables in a model of their challenge.
- (c) Formulate BG's problem as a time-phased finite-horizon linear program over the decision variables of part (a). Assume all demand occurs on the last day of the month. Be sure to annotate each objective function and constraint with a few words indicating its meaning.
- (d) Enter and solve your model with class optimization software.

4-17 Seasons Greetings (SG) manufacturers artificial holiday trees decorated with embedded multi-colored lights suitable for homes or stores, tree sales vary seasonally, with 1 thousand demanded in the first quarter of the year, 5 thousand in the second, 10 thousand in the third, and 7 thousand in the fourth. Corresponding profits to the company also vary with \$50 per tree gained on sales in the high-demand third and fourth quarters, and \$35 per tree in slower-demand first and second quarters. Still, SG can manufacture only 5 thousand in any quarter due to the limits of its production facility, so not all demand can be met. Inventory can be produced and held to exploit varying profit levels, but holding costs SG \$12 per quarter per tree.

- (a) Define index sets and symbolic parameters naming and indexing the model constants described above.
- (b) Formulate a linear program to compute an optimal (max net income) sales and inventory plan for SG using the parameters of part (a) and assuming perpetual inventory. Use decision variables s_q = sales in quarter q and h_q = units held in quarter q , for $q = 1, \dots, 4$. Be sure to briefly annotate the objective function and constraints (both main and variable-type) to indicate their meaning.
- (c) Enter and solve your model with class optimization software.

4-18 Down Hill Ski (DHS) manufactures high speed racing skis for the most adventurous of skiers. Ski sales vary seasonally, with 7 thousand pairs demanded in the first quarter of the year,

2 thousand in the second, 1 thousand in the third, and 10 thousand in the fourth. Corresponding profits to the company also vary with \$500 per pair gained on sales in the high-demand first and fourth quarters, and \$350 per pair in the other, slower-demand quarters. Still, DHS will be manufacturing only 4 thousand pairs per quarter due to the limits of its production facility, so not all demand can be met. Inventory can be produced and held to exploit varying profit levels, but holding costs DHS \$40 per quarter per pair.

- Define symbolic parameters naming and indexing the constants described above.
- Formulate a linear program to compute an optimal (max net income) sales and inventory plan for SG using the parameters of part (a) and assuming perpetual inventory. Use decision variables s_q = sales in quarter q and h_q = units held in quarter q , for $q = 1, \dots, 4$. Be sure to briefly annotate the objective function and constraints (both main and variable-type) to indicate their meaning. Be sure to briefly annotate the objective function and constraints (main and variable-type) with a few words indicating their meaning.

(c) Enter and solve your model with class optimization software.

4-19 Ace Green Windows (AGW) manufactures environmentally efficient windows as replacements of those in existing homes. It has just received a contract for the next 6 months, requiring 100, 250, 190, 140, 220, and 110 units in months 1 through 6, respectively. Production costs for windows vary with time due to the specialized materials. AGW estimates production will cost \$250, \$450, \$350, \$400, \$520, and \$500 per unit in periods 1 through 6, respectively. To take advantage of cost variations, AGW may produce more windows than the contract requires in some months and hold up to 375 of them in inventory for later months. But holding inventory costs \$30 per window per month. Assume there is no beginning inventory.

- Define symbolic parameters naming and indexing the constants described above.
- Formulate a time-phased linear program to compute an optimal (minimum total cost) finite-horizon, manufacturing and inventory plan for the next 6 months

using the parameters of part (a). Use decision variables x_t = units produced in month t and h_t = units held in month t , for $t = 1, \dots, 6$. Briefly annotate the objective function and constraints (main and variable-type) with a few words indicating their meaning.

(c) Enter and solve your model with class optimization software.

4-20 Global Minimum manufactures bikini swimming suits. Their business is highly seasonal, with expected demands being 2800, 500, 100, and 850 dozen suits over the four quarters of next year. The company can produce 1200 dozen suits per quarter, but inventories must be built up to meet larger demands at a holding cost of \$15 per dozen per quarter. Global wants to meet demand while minimizing this inventory cost.

- Formulate a time-phased LP to choose an optimal production plan assuming an infinite time horizon.
- Which of the constraints in your model are balance constraints? Explain.
- Explain why your model has an infinite horizon even though it details only four quarters.

(d) Enter and solve your model with the class optimization software.

4-21 A company manufactures parts $i = 1, \dots, m$ in weeks $t = 1, \dots, n$, where each unit of part i requires $a_{i,k}$ units of production resource $k = 1, \dots, q$ and has value v_i . Production resource capacities b_k cannot be exceeded in any period and part demands $d_{i,t}$ must be met. Express each of the following as linear constraint(s) in these parameters and the non-negative decision variables $x_{i,t} \triangleq$ number of units of i produced in week t and $z_{i,t} \triangleq$ inventory of product i held at the end of period t . Initial inventories all = 0.

- No production capacity can ever be exceeded.
- The total value of held inventories should never exceed 200.
- Quantities of each part i available after week 1 should balance with demand and accumulated inventory.
- Quantities of each part i available after weeks $2, \dots, n - 1$ should balance with demand and accumulated inventory.

4-22 The following table shows observed electrical power consumption at several different levels of a factory's operation.

Level	2	3	5	7
Power	1	3	3	5

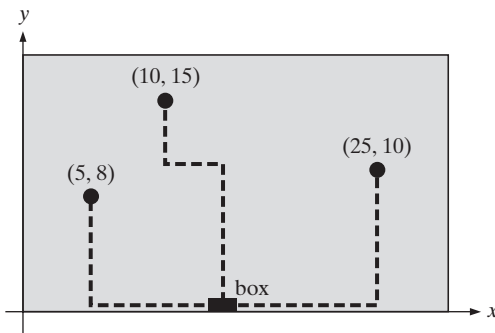
Engineers want to fit the estimating relationship

$$\text{power} = \beta_0 + \beta_1 \text{level}$$

to these data in a way that minimizes the sum of the absolute deviations between predicted and observed power requirements. Both parameters β_0 and β_1 should be nonnegative.

- ✓ (a) Formulate a linearized nonlinear LP to choose optimal parameter values. (*Hint:* β_0 and β_1 are among the decision variables.)
- ✓ (b) Enter and solve your model with the class optimization software.
- ✓ (c) Determine from your solution in part (b) the absolute deviation at each observed point.

4-23 The following figure shows the ceiling locations of 3 sensors in a new factory relative to a coordinate system (in feet) with origin at the lower left. A control box will be located along the long (lower in the figure) wall with fiber-optic cables running rectilinearly to each sensor. Designers want to place the box to minimize the cable required.



- (a) Formulate a linearized nonlinear LP to choose an optimal placement.
- ✓ (b) Enter and solve your model with the class optimization software.

(c) How much cable would be required to implement your solution of part (b)?

✓ **4-24** Repeat Exercise 4-22, this time choosing a fit that minimizes the maximum deviation between observed and predicted power requirements.

4-25 Repeat Exercise 4-23, this time minimizing the length of the longest cable.

4-26 The American Edwards Laboratories (AEL)¹⁰ manufactures artificial human heart valves from pig hearts. One of the things making planning complex is that the size of pig hearts is highly variable, depending on breed, age when slaughtered, feed mix, and so on. The following (fictitious) table shows the fraction of hearts from suppliers $j = 1, \dots, 5$ yielding each of the valve sizes $i = 1, \dots, 7$, along with the maximum quantity available from each supplier per week and the unit cost of hearts obtained.

Size	Supplier j				
	1	2	3	4	5
1	0.4	0.1	—	—	—
2	0.4	0.2	—	—	—
3	0.2	0.3	0.4	0.2	—
4	—	0.2	0.3	0.2	—
5	—	0.2	0.3	0.2	0.2
6	—	—	—	0.2	0.3
7	—	—	—	0.2	0.5
Availability	500	330	150	650	300
Cost	2.5	3.2	3.0	2.1	3.9

AEL wants to decide how to purchase hearts to meet weekly requirements of 20 size 1, 30 size 2, 120 size 3, 200 size 4, 150 size 5, 60 size 6, and 45 size 7 valves at minimum total cost.

✓ (a) Formulate an LP model of this heart purchase planning problem using the decision variable ($j = 1, \dots, 5$)

$$x_j \triangleq \text{number of hearts purchased weekly from supplier } j$$

✓ (b) Enter and solve your model with the class optimization software.

¹⁰Based on S. S. Hilal and W. Erikson (1981), "Matching Supplies to Save Lives: Linear Programming the Production of Heart Valves," *Interfaces*, 11:6, 48–55.

4-27 Midville Manufacturing assembles heavy duty materials handling carts to meet demand of 500 units in the first quarter of each year, 1200 in the second, 1000 in the third, and 300 in the fourth. Elementary components, which consist of wheels, steering yokes, and carrying platforms, are first assembled separately. Then each steering yoke is equipped with 4 wheels to form the front-end subassembly. Finally, front-end subassemblies are combined with a carrying platform and 8 additional wheels at the rear to complete the cart. Using $j = 1$ for steering yokes, $j = 2$ for wheels, $j = 3$ for platforms, $j = 4$ for front-end assemblies, and $j = 5$ for finished carts, the following table shows the estimated value of each element (in dollars) and the factory hours required to assemble it.

	Element j				
	1	2	3	4	5
Value	120	40	75	400	700
Time	0.06	0.07	0.04	0.12	0.32

Components, subassemblies, and finished carts produced in any quarter may be used or shipped in the same quarter or held over as inventory (including from the fourth quarter to the first) at 5% per quarter interest on the held value. Midville seeks a plan that minimizes these holding costs while conforming to the factory production capacity, 1150 hours per quarter.

- (a) Formulate an LP model to choose a production plan using the decision variables ($j = 1, \dots, 5; q = 1, \dots, 4$)

$x_{j,q} \triangleq$ number of units of element j produced in quarter q

$h_{j,q} \triangleq$ number of units of element j held in inventory from quarter q to the next

Your model should include a system of main constraints for production capacity and an additional system for each element j to enforce material balance.

- (b) Enter and solve your model with the class optimization software.

4-28 A construction contractor has undertaken a job with 7 major tasks. Some of the tasks can begin at any time, but others have predecessors that must be completed first. The following table shows those predecessor task numbers, together with the minimum and maximum time (in days) allowed for each task, and the total cost that would be associated with accomplishing each task in its minimum and maximum times (more time usually saves expense).

j	Min.	Max.	Cost	Cost	Predecessor Tasks
	Time	Time	Min.	Max.	
1	6	12	1600	1000	None
2	8	16	2400	1800	None
3	16	24	2900	2000	2
4	14	20	1900	1300	1,2
5	4	16	3800	2000	3
6	12	16	2900	2200	3
7	2	12	1300	800	4

The contractor seeks a way to complete all work in 40 days at least total cost, assuming that the cost of each task is linearly interpolated for times between the minimum and maximum.

- (a) Formulate an LP model of this time/cost planning problem using the decision variables ($j = 1, \dots, 7$)

$s_j \triangleq$ start time of task j (in days)

$t_j \triangleq$ days to complete task j

Your model should have an objective function summing interpolated cost and main constraints to enforce precedence relationships and the time limit.

- (b) Enter and solve your model with the class optimization software.

4-29 Import Books, Incorporated (IBI)¹¹ stocks several thousand titles in its main warehouse. The titles can be categorized by sales volume, with $i = 1$ requiring a stored inventory of 0 to 20 books, $i = 2$ requiring 21 to 40, $i = 3$ requiring 41 to 100, and $i = 4$ requiring 101 to 200. The number of titles in category i is b_i . Each title is stored in a separate bin, and each bin has at most

¹¹Based on R. J. Paul and R. C. Thomas (1977), "An Integrated Distribution, Warehousing and Inventory Control System for Imported Books," *Operational Research Quarterly*, 28, 629–640.

one title. IBI has 500 bins with space for up to 100 books and 2000 larger ones with space for up to 200 books. The bins for 100 can also be subdivided to create either two bins for 40 books or three bins for 20. Costs $c_{i,j}$ for storing a category i title in a size j bin have been estimated by accounting for the material handling cost of accessing the bin and the wasted space if a bin is underutilized. Here $j = 1$ refers to bins for 20, $j = 2$ to bins for 40, $j = 3$ to bins for 100, and $j = 4$ to bins for 200. Formulate an LP model to find a minimum cost allocation of titles to bins using the decision variable ($i = 1, \dots, 4; j = i, \dots, 4$)

$$x_{i,j} \triangleq \begin{array}{l} \text{number of titles of category} \\ i \text{ allocated to of size } j \end{array}$$

4-30 Radiation therapy planning for cancer treatment begins with computer images of several body tissues. A tumorous **target** is identified along with surrounding **healthy tissues**. The treatment goal is to get maximize the total radiation received by a target tumor $t = 0$ while avoiding damage to surrounding tissues $t = 1, \dots, T$ by limiting the radiation allowed to fall on them. Radiation is provided from a large **accelerator** that can shoot beams from multiple angles $j = 1, \dots, J$ around the patient's body so as to spread the danger to healthy tissues while focusing on the tumor. The accelerator beam is relatively large, often approximately 10cm square. That is why **Intensity Modulated Radiation Therapy (IMRT)** adds precision to plans by treating each beam j as if made up of many **beamlets** $k = 1, \dots, K_j$ with independently controllable **intensities** (roughly exposure times) $x_{j,k} \geq 0$.

It is important that both tumor and healthy tissue dose be spread fairly evenly across their respective volumes. This is modeled by dividing each tissue volume t into a larger number of mini-volumes called **voxels** $i = 1, \dots, I_t$. Then the impact of particular beamlets (j, k) on any voxel (t, i) can be estimated as $a_{j,k,t,i}$ per unit beamlet intensity. The total dose received at any voxel (t, i) can be assumed to be the sum of these contributions across all beamlets, that is, $\sum_{j=1}^J \sum_{k=1}^{K_j} a_{j,k,t,i} x_{j,k}$. Then such total voxel

doses within healthy tissues $t = 1, \dots, T$ must be limited to a specified maximum safe doses b_t .

Using the indicies and symbols defined above, formulate a linear program over decision variables $x_{j,k}$ to compute a treatment plan maximizing the average dose to target voxels while satisfying upper limits for all voxels of each surrounding healthy tissue. Be sure to explain and justify the meaning of the objective and all constraints.

4-31 Dairy cows¹² in most countries calve on a regular annual basis. Their milk output varies over the year accordingly, with a peak reached a few months after calving followed by a decline to almost zero in the tenth month. Knowing these facts, farmers in an agricultural cooperative are trying to plan calving months $c = 1, \dots, 12$ to make it easier to meet seasonal milk demands r_d pounds in months $d = 1, \dots, 12$. Any milk produced beyond these demands must be sold on the bulk market at b per pound below the regular price. The annual cost m_c of maintaining a cow calving in month c varies significantly over the year because low-cost grazing is available only in certain seasons. From scientific studies the farmers can estimate the yield $p_{d,c}$ pounds in demand month d per cow calving in month c . Formulate an LP model to determine a minimum total cost calving schedule using the decision variables ($c, d = 1, \dots, 12$)

$$x_c \triangleq \begin{array}{l} \text{number of cows calving in month } c \\ y_d \triangleq \text{pounds of excess milk} \\ \text{produced in demand month } d \end{array}$$

4-32 Blue Bell¹³ is planning its monthly production of a particular type of men's jeans. Demands d_i are know for the $i = 1, \dots, 75$ different fabric parts needed to make all the combinations of waist and inseam sizes being produced. Such parts are cut from fabric laid out on cutting tables in 60 to 70 layers. A predefined set of markers (cutting patterns) $m = 1, \dots, 350$ define how various parts may be cut. Each use of marker m yields $a_{i,m}$ copies of part i per layer and wastes w_m square yards of fabric in areas between the usable

¹²Based on L. Killen and M. Keane (1978), "A Linear Programming Model of Seasonality in Milk Production," *Journal of the Operational Research Society*, 29, 625-631.

¹³Based on J. R. Edwards, H. M Wagner, and W. P. Wood (1985), "Blue Bell Trims Its Inventory," *Interfaces*, 15:1, 34-52.

parts. Formulate an LP model to choose a minimum total waste cutting plan using the decision variable ($m = 1, \dots, 350; p = 60, \dots, 70$)

$x_{m,p} \triangleq$ number of times marker pattern m is cut in a layup of p layers

4-33 To assess the impact on the U.S. coal market of different pollution control strategies, the Environmental Protection Agency (EPA)¹⁴ wants to determine, for assumed control regimes, how much coal from supplies s_i in different mining regions $i = 1, \dots, 24$ will be extracted, how much will then be processed into various deliverable coal types $m = 1, \dots, 8$, and how much of each deliverable product will be sent to meet consumer demands $d_{m,j}$ in regions $j = 1, \dots, 113$. Demands are expressed in Btu, with each ton of raw coal mined at i for processing into type m yielding $a_{i,m}$ Btu. Including the economic burden of pollution controls and transportation, the cost per ton of raw coal mined at i for processing into type m and use at j can be estimated at $c_{i,m,j}$. Formulate an LP model to determine how coal would be mined, processed, and distributed if the market seeks to minimize total cost. Use the decision variable ($i = 1, \dots, 24; m = 1, \dots, 8; j = 1, \dots, 113$)

$x_{i,m,j} \triangleq$ tons of coal mined at i for processing into m and use at j

4-34 Quantas Airways Ltd.¹⁵ must schedule its hundreds of reservation salesclerks around the clock to have at least r_t on duty during each 1-hour period starting at (24-hour) clock hour $t = 0, \dots, 23$. A shift beginning at time t extends for 9 hours with 1 hour out for lunch in the fourth, fifth, or sixth hours of the shift. Shifts beginning at hour t cost the company c_t per day, including wages and night-hour premiums. Formulate an LP model to compute a minimum total cost daily shift schedule using the decision variables ($t = 0, \dots, 23; i = t + 4, \dots, t + 6$)

$x_i \triangleq$ number of clerks working a shift starting at hour t

$y_{i,t} \triangleq$ number of clerks working a shift starting at hour t who take lunch during hour i

4-35 An Indian reservation irrigation project¹⁶ must decide how much water to release through the gate at the top of its main canal in each of the upcoming 4-hour periods $t = 1, \dots, 18$. Ideal canal outflows, r_t , are known for each time period, and the total outflow over all 18 periods should equal or exceed the sum of these quantities. However, period-to-period deviations may be needed to avoid flooding. The initial canal storage is 120 units, and the net effect of releases and outflows should never leave more than u units stored after any period. Within these limits, managers would like to minimize the total absolute deviation between desired demands r_t and actual outflows. Formulate an LP model of this irrigation control problem using the decision variables ($t = 1, \dots, 18$)

$x_t \triangleq$ gate release during period t

$s_t \triangleq$ amount of water stored in the canal at the end of period t

$w_t \triangleq$ canal outflow during period t

$d_1^+ \triangleq$ over satisfaction of demand in period t

$d_1^- \triangleq$ undersatisfaction of demand in period t

4-36 Major shopping mall developer Homart¹⁷ is selecting the tenant mix for its next facility. Stores of product types $i = 1, \dots, 20$ are being considered for arrangement into the new mall's sectors $j = 1, \dots, 5$. Each sector will have 150 thousand square feet, and an allowance of c_i per square foot will be set aside for finishing of areas allocated to stores of type i . From prior experience, Homart can estimate the present worth $p_{i,j}$ of revenues from a type i store located in sector j and the required floor space a_i (in thousands

¹⁴Based on C. Bullard and R. Engelbrecht-Wiggans (1988), "Intelligent Data Compression in a Coal Model," *Operations Research*, 38, 521–531.

¹⁵Based on A. Gaballa and W. Pearce (1979), "Telephone Sales Manpower Planning at Quantas," *Interfaces*, 9:3, 1–9.

¹⁶Based on B. J. Boman and R. W. Hill (1989), "LP Operation Model for On-Demand Canal Systems," *Journal of Irrigation and Drainage Engineering*, 115, 687–700.

¹⁷Based on J. C. Bean, C. E. Noon, S. M. Ryan, and G. J. Salton (1988), "Selecting Tenants in a Shopping Mall," *Interfaces*, 18:2, 1–9.

of square feet). They seek a tenant mix that will maximize total present worth while having between \underline{n}_i and \bar{n}_i stores of each type i totaling between \underline{f}_j and \bar{f}_j thousand square feet, and not exceeding the budget b for finishing allowances. Formulate an LP model of this tenant mix problem using the decision variable

$$x_{i,j} \triangleq \text{number of stores of type } i \text{ included in sector } j$$

4-37 Once the configuration of molds is fixed, the planning of production of aluminum ingots¹⁸ reduces to allocating the time of furnaces $j = 1, \dots, n$ among alloys $i = 1, \dots, m$ and ingot sizes $s = 1, \dots, p$. Yields $a_{j,s}$ of ingots of size s producible from furnace j during the entire planning period can be estimated. Planning should meet demands $d_{i,s}$ for ingots of alloy i and size s , but this may require misapplying some ingots (i.e., trimming some larger ingots sizes s' greater than s to meet demands for size s of the same alloy). Misapplications result in a trim-loss cost $c_{i,s',s}$ for each ingot cut down from size s' to s . Managers want to find a feasible plan that minimizes the total cost of these misapplications. Formulate an LP model of this ingot production planning problem using decision variables ($i = 1, \dots, m; j = 1, \dots, n; s = 1, \dots, p, s' > s$)

$$x_{i,j,s} \triangleq \text{fraction of time on furnace } j \text{ dedicated to making ingots of alloy } i, \text{ size } s$$

$$y_{i,s',s} \triangleq \text{number of ingots of alloy } i, \text{ size } s', \text{ misapplied to meet demand for size } s$$

4-38 S&S operates its large supermarkets¹⁹ on a 24-hour per day basis using only part-time cashiers working shifts of 2 to 5 hours per day. All shifts start on the hour. The required number r_h of cashiers on duty at a given store is known for (24-clock) hours $h = 0, \dots, 23$, and managers can also estimate the number of employees b_l willing

to work shifts of lengths $l = 2, \dots, 5$. They seek a shift schedule that meets requirements at minimum total cashier hours worked. Formulate an LP model of this shift scheduling problem using the decision variable ($h = 0, \dots, 23; l = 2, \dots, 5$)

$$x_{h,l} \triangleq \text{number of cashiers starting an } l\text{-hour shift at hour } h$$

Neglect the fact that the numbers working each shift must physically be integers.

4-39 The transmitted gray-scale value $g_{i,j}$ of pixels $i = 1, \dots, m, j = 1, \dots, n$, in a digital space satellite photo²⁰ is distorted by both the usual random noise and a known problem with the video camera that effectively multiplies the value for pixel (i, j) by a blurring factor $b_{i,j}$. Engineers want to restore the image by estimating correct values for each pixel in a way that minimizes the total absolute deviation between predicted (after blurring) and observed gray-scale numbers. Formulate an LP model of this image restoration problem using decision variables ($i = 1, \dots, m; j = 1, \dots, n$)

$$x_{i,j} \triangleq \text{correct value for pixel } (i, j)$$

$$d_{i,j}^+ \triangleq \text{positive deviation of predicted over observed value for pixel } (i, j)$$

$$d_{i,j}^- \triangleq \text{negative deviation of predicted below observed value for pixel } (i, j)$$

4-40 The Hanshin expressway²¹ serves the Osaka–Kobe area of Japan. Due to heavy congestion, the number of vehicles entering at each ramp $j = 1, \dots, 38$ of the expressway is controlled by a system that reevaluates the situation every 5 minutes based on current queue lengths q_j at each ramp and estimated number of new entry-seeking arrivals d_j over the next 5 minutes. The system enforces end-of-period queue-length limits u_j and total traffic capacities b_i on 500-meter segments $i = 1, \dots, 23$ of the highway.

¹⁸Based on M. R. Bowers, L. A. Kaplan, and T. L. Hooker (1995), “A Two-Phase Model for Planning the Production of Aluminum Ingots,” *European Journal of Operational Research*, 81, 105–114.

¹⁹Based on E. Melachrinoudis and M. Olafsson (1992), “A Scheduling System for Supermarket Cashiers,” *Computers and Industrial Engineering*, 23, 121–124.

²⁰Based on R. V. Digumarthi, P. Payton, and E. Barrett (1991), “Linear Programming Solutions of Problems in Logical Inference and Space-Variant Image Restoration,” *Image Understanding and the Man–Machine Interface III*, SPIE Vol. 1472, 128–136.

²¹Based on T. Yoshino, T. Sasaki, and T. Hasegawa (1995), “The Traffic-Control System on the Hanshin Expressway,” *Interfaces*, 25:1, 94–108.

Traffic entering at j affects only down-stream segments, and a part may exit before reaching any given i . Prior engineering studies have captured this behavior in estimated fractions $f_{i,j}$ of vehicles entering at j that persist to consume capacity at i . The system seeks the feasible control policy that permits the maximum total number of entry-seeking vehicles into the expressway during the next time period. Formulate an LP model of this traffic control problem using the decision variables ($j = 1, \dots, 38$)

$x_j \triangleq$ number of vehicles allowed to enter at ramp j during the time period

4-41 Industrial engineers are planning the layout of cells $i = 1, \dots, 18$ in a rectangular manufacturing facility of $x = 1000$ by $y = 200$ feet with a 6-foot-wide, two-way conveyor system along the $y = 0$ boundary²². It has already been decided that cells will be sequenced along the conveyor in the same order as their i subscripts, but the exact geometry of the cells remains to be fixed. Analysis of cell loadings has produced lower limits x_j and y_j on the x and y dimensions of each cell. Engineers have also specified minimum cell perimeters p_i as a surrogate for area (which would lead to nonlinear optimization). Conveyor traffic will enter and exit at input and output stations located at cell x -midpoints along the conveyor, and one-way traffic flows from cell i to cell j are estimated at $f_{i,j}$. The IEs seek a feasible design that minimizes total travel (flow times distance) on the conveyor. Formulate an LP model of this layout problem using the decision variables ($i, j = 1, \dots, 18$)

$x_j \triangleq$ left x -coordinate of cell j

$y_i \triangleq$ y -depth of cell j

$d_{i,j}^+ \triangleq$ positive x -distance between I/O stations of cells i and j

$d_{i,j}^- \triangleq$ negative x -distance between I/O stations of cells i and j

4-42 Swift Chemical Company²³ mines phosphate rock, collects it in inventory piles $i = 1, \dots, 8$, and blends it to meet contracts with customers $k = 1, \dots, 25$ at profit p_{ik} per ton. The critical measure of phosphate content in rock is its BPL. Piles correspond to different average BPL contents b_i per ton, asset value a_i per ton, contract net profit $r_{i,k}$ per ton, starting inventory h_j , and expected quantity q_i to arrive from mines vary accordingly. Each contract includes a minimum s_k and a maximum \bar{s}_k number of tons to be shipped, along with a minimum \underline{p}_k and a maximum \bar{p}_k average BPL content. Managers want to schedule blending and sales to maximize total profit plus total ending inventory asset value. Formulate an LP model of this phosphate planning problem using the decision variables ($i = 1, \dots, 8; k = 1, \dots, 25$)

$x_{i,k} \triangleq$ tons of rock from pile i included in shipment for contract k

$h_i \triangleq$ ending inventory in pile i

4-43 Any convex 3-dimensional object (i.e., a body such that the line segment between any two points in its volume falls entirely within the volume) with flat sides can be described as the set of points (x, y, z) satisfying a series of linear constraints²⁴. For example, a 3-by 5-by 9-meter box with one corner at the origin can be modeled as

$$\{(x, y, z): 0 \leq x \leq 3, 0 \leq y \leq 5, 0 \leq z \leq 9\}$$

Suppose that a stationary object is described in this way by constraints

$$a_i x + b_i y + c_i z \leq d_i \quad i = 1, \dots, 19$$

and that a link of a robot arm is described at its initial position by the constraints

$$p_j x + q_j y + r_j z \leq s_j \quad j = 1, \dots, 12$$

The object and the link do not intersect at that initial position, but the link is in motion. Its location is being translated from the initial location by growing a step $\alpha > 0$ in direction $(\Delta x, \Delta y, \Delta z)$. Formulate an LP in terms of decision variables x ,

²²Based on A. Langevin, B. Montreuil, and D. Riopel (1994), "Spine Layout Problem," *International Journal of Production Research*, 32, 429–442.

²³Based on J. M. Reddy (1975), "A Model to Schedule Sales Optimally Blended from Scarce Resources," *Interfaces*, 5:1, 97–107.

²⁴Based on R. Gallerini and A. Sciomachen (1993), "On Using LP to Collision Detection between a Manipular Arm and Surrounding Obstacles," *European Journal of Operational Research*, 63, 343–350.

y , z , and α to find the smallest step (if any) that will produce a collision between the object and the link, and indicate how the LP would detect the case where no collision will occur.

4-44 The principal export of Iceland²⁵ is fish which are very perishable and subject to high day-to-day variation in the size of the catch available for processing. Each day processing begins at any packing plant with estimates b_f of the kilograms of raw fish species $f = 1, \dots, 10$ that will be available for processing for market $m = 1, \dots, 20$. The marketable volume (in kilograms) of product m in any day is at most u_m . Each kilogram of raw fish f processed for market m yields $a_{f,m}$ kilograms of the final product and produces a gross profit of $p_{f,m}$ in sales minus costs other than labor. Processing fish f into product m requires $h_{f,m,i}$ hours of worker time at stations $i = 1$ (filleting), $i = 2$ (packing), and $i = 3$ (freezing). A total of q_i hours of workers can be obtained for workstation i at an average wage of c_i per hour. The daily plan should maximize total gross profit minus wage cost. Formulate an LP model to compute an optimal fish processing plan using the decision variables ($f = 1, \dots, 10$; $m = 1, \dots, 20$; $i = 1, \dots, 3$)

- $x_{f,m} \triangleq$ kilograms of raw fish f processed for market m
- $y_i \triangleq$ worker hours at workstation i

4-45 The U.S. Air Force (USAF)²⁶ must procure aircraft types $i = 1, \dots, 10$ and associated munition types $j = 1, \dots, 25$ to meet anticipated sortie requirements against target types $k = 1, \dots, 15$ in weather conditions classes $\ell = 1, \dots, 8$. Targets k are assigned a value r_k , and $t_{k,\ell}$ are anticipated under weather condition ℓ in the assumed war scenario. An aircraft type i using munitions type j under weather conditions ℓ has probability $p_{i,j,k,\ell}$ of killing a type k target each sortie, and a load of $b_{i,j,k,\ell}$ munitions j is required. A total of $s_{i,j,k,\ell}$ such sorties could be

flown by each available type i aircraft during the assumed war. Currently, there are a_i aircraft of type i available, and new ones can be procured at $\$c_i$ billion per unit. Similarly, the current inventory of munitions type j is m_j and new ones can be procured at $\$d_j$ billion per unit. The USAF wants to buy planes and munitions to maximize the total expected value of targets it could kill if the assumed war scenario happened subject to a current-year procurement budget of \$100 billion. Formulate an LP model of this weapons procurement problem using the decision variables ($i = 1, \dots, 10$; $j = 1, \dots, 25$; $k = 1, \dots, 15$; $\ell = 1, \dots, 8$)

- $x_{i,j,k,\ell} \triangleq$ number of sorties flown by aircraft type i with munitions type j against target type k under weather conditions class ℓ
- $y_i \triangleq$ number of new type i aircraft procured
- $z_j \triangleq$ number of new type j munitions procured

4-46 North American Van Lines²⁷ maintains a fleet of several thousand truck tractors, each of which is owned by one of its contract truckers. Tractors can be anywhere in the range $i = 0, \dots, 9$ years old. Every 4-week planning period $t = 1, \dots, 13$, tractors may be purchased new at price p , sold to contractors at price s_i , traded in to manufacturers at allowance a_i , and repurchased from contractors at price r_i . Only new units can be sold or traded as age $i = 0$, and no more total units of any age can be traded in any period than are purchased new in the same period. The number of tractors in the fleet inventory (i.e., with contractors) during each period t must fall between a minimum l_t and a maximum u_t to meet seasonal moving demands without accumulating excess capacity. Managers wish to find a plan that maximizes their net total income from buying, selling, trading, and repurchasing trucks. Formulate an LP model of this fleet

²⁵Based on P. Jennson (1988), "Daily Production Planning in Fish Processing Firms," *European Journal of Operational Research*, 36, 410-415.

²⁶Based on R. J. Might (1987), "Decision Support for Aircraft and Munitions Procurement," *Interfaces* 17:5, 55-63.

²⁷Based on D. Avrmovich, T. M. Cook, G. D. Langston, and F. Sutherland (1982), "A Decision Support System for Fleet Management: A Linear Programming Approach," *Interfaces*, 12:3, 1-9.

management problem using the decision variables ($i = 0, \dots, 9; t = 1, \dots, 3$)

- $w_t \triangleq$ number of new tractors purchased in period t
- $x_{i,j} \triangleq$ number of tractors of age i sold to contractors in period t
- $y_{i,j} \triangleq$ number of tractors of age i traded in period t
- $z_{i,j} \triangleq$ number of tractors of age i repurchased from contractors in period t
- $f_{i,j} \triangleq$ number of tractors of age i in the fleet at the beginning of period t

Assume that vehicles enter and leave the fleet only through sales to and repurchases from contractors. Also assume that the fleet ages 1 year as inventory is passed from period 13 to period 1 and that no 9-year-old vehicles can be carried over to the new year.

4-47 The College County Election Board (CCEB) is planning for election machines needed for the coming national election in the county's 4 voting precincts. Each machine can be expected to service 100 voters on election day, but the challenge is that the number of voters at each precinct is unpredictable. Precincts 1 and 2 in the more upscale, highly educated part of the county experience fairly consistent voting, but precinct 3, and especially university district precinct 4 are highly volatile. The table below shows the numbers of citizens (in hundreds) whom history suggests would vote in each precinct, depending upon whether the overall election is a Low, Medium, or High turnout one.

Precinct Number	Turnout (hundreds)		
	Low	Medium	High
1	5	6	7
2	4	7	8
3	2	6	10
4	2	8	15
Prob	0.25	0.35	0.40

The table also shows to probability of the coming election being of each type.

The CCEB must choose how many machines to pre-position at each precinct on election day, and how many more to keep in a warehouse

from which they can be distributed to precincts after the shape of the election is revealed in the early hours of voting. Machines cost \$5 thousand each to obtain and program plus an additional \$0.5 thousand to relocate any from warehouse to a polling place. The Board wants to stay within its budget of \$150 thousand while minimizing the expected total number of voters who have long waits because there are not enough machines at their precinct to meet the demand.

- ✓ (a) Explain how CCEB's planning task can be modeled as a Two-Stage Stochastic Program with recourse (definition 4.18)? Specifically, what will be decided in Stage 1, what scenarios s need to be treated in Stage 2, and what recourse decisions are available?
- ✓ (b) Assuming it is satisfactory to treat numbers of voting machines as continuous, formulate a Stochastic Linear Program in Extensive Format (Figure 4.4) to compute a minimum expected-value plan for CCEB. Use Stage 1 decision variables $x_p \triangleq$ the number of voting machines (hundreds) initially positioned at precincts $p = 1, \dots, 4$ or at the warehouse $p = 5$. Combine with Stage 2 decision variables $y_p^{(s)} \triangleq$ the number of machines (hundreds) shipped from the warehouse to precinct p under scenario s , and $w_p^{(s)} \triangleq$ the number (hundreds) of voters at precinct p who will experience long waits due to a shortage of machines under scenario s .
- ✓ (c) Use class optimization software to solve your model of (b) and explain what decisions prove optimal.

4-48 Although it is only August, the Big View (BV) electronics company is placing orders now for holiday shopping season sales of a new super-large, flat-screen TV. Orders will be delivered from the overseas manufacturer to the company's sites in 3 regional shopping malls. Due to the global nature of orders, this will be BV's only buying opportunity until after the first of next year. The product is very new and different from competitors, which makes BV uncertain how well it will sell. The following table

shows projected sales (if stock is available) at the 3 malls under 5 possible demand scenarios, along with the probability of each:

Mall	Demand Scenario				
	1	2	3	4	5
1	200	400	500	600	800
2	320	490	600	475	900
3	550	250	400	550	650
Prob	0.10	0.20	0.40	0.20	0.10

BV will pay \$500 per TV purchased and shipped to the 3 malls. Extra units may be obtained for one mall from another’s excess at a transfer cost of \$150. Whatever the source, each TV sold during the season will bring \$800. TV’s still available at the end of the season in any malls will all be sold at the clearance price of \$300. BV wishes to find a plan that maximizes the total expected net profit over all the units purchased in August.

- (a) Explain how BV’s planning task can be modeled as a Two-Stage Stochastic Program with recourse (definition 4.18)? Specifically, what will be decided in Stage 1, what scenarios s need to be treated in Stage 2, and what recourse decisions are available?
- (b) Assuming it is satisfactory to treat numbers of TVs as continuous, formulate a

Stochastic Linear Program in Extensive Format (Figure 4.4) to compute a maximum expected net-profit plan for BV. Use Stage 1 decision variables $x_m \triangleq$ the number TVs purchased for mall m . Combine with Stage 2 decision variables $w_{m,n}^{(s)} \triangleq$ the number of units transferred from mall m to mall n under scenario s , $y_m^{(s)} \triangleq$ the number of TVs sold in mall m under scenario s , and $z_m^{(s)} \triangleq$ the number of overstock units sold at discount from mall m after the season.

- (c) Use class optimization software to solve your model of (b) and explain what decisions prove optimal.

4-49 The Zoom automobile company²⁸ is planning capacities and configurations of 3 plants to produce 4 new models being introduced for the coming market cycle. The first part of the following table shows the configuration options available at each plant, along with the implied capacities (in thousands of cars) and (fixed) changeover costs (in \$ million). Option 1 of each plant is the current facility configuration making changeover cost = 0. Different configurations at the plants imply different mixes of models that can be manufactured there. Part two of the table adds the marginal production cost per thousand units of the products eligible for manufacture at the site given the chosen configuration.

Configuration	Plant $j = 1$		Plant $j = 2$			Plant $j = 3$		
	$k = 1$	$k = 2$	$k = 1$	$k = 2$	$k = 3$	$k = 1$	$k = 2$	$k = 3$
Capacity ($\triangleq u_{jk}$)	37	45	44	50	60	25	47	59
Changeover ($\triangleq f_{jk}$)	0	12	0	10	22	0	15	29
Product	Production Costs ($\triangleq c_{ijk}$ in \$ 000 if Allowed)							
$i = 1$	20	18	22	20	—	—	—	16
$i = 2$	—	—	—	21	18	—	—	19
$i = 3$	—	30	—	—	27	33	—	—
$i = 4$	—	—	34	—	36	32	31	22

²⁸Based on G. D. Eppen, R. K. Martin and L. Schrage (1989), “A Scenario Approach to Capacity Planning,” Operations Research, 37, 517–527.

Exactly one configuration must be chosen for each plant well in advance, but demands and market prices that will arise for units produced are uncertain until later. The next table shows 4 scenarios

for available demands and prices applicable for the entire market cycle. Estimated probabilities for the scenarios are $p^{(s)} = 0.15, 0.30, 0.35,$ and 0.20 respectively.

Product	Quantity Demanded ($\triangleq d_i^s$ in 000s)				Market Price ($\triangleq r_i^s$ \$ 000)			
	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 1$	$s = 2$	$s = 3$	$s = 4$
$i = 1$	30	25	20	17	30	25	22	21
$i = 2$	24	20	22	35	27	22	23	33
$i = 3$	17	16	21	14	33	30	35	27
$i = 4$	52	36	30	40	45	30	28	33

- (a) Explain why Zoom’s capacity planning task can be formulated as a two-stage mixed-integer stochastic program with recourse (definition [4.18]) taking production capacity decisions in Stage 1 and determining implied scenario production and sales in Stage 2.
- (b) Using parameters defined above and decision variables $y_{jk} \triangleq 1$ if plant j is placed in configuration k ($= 0$ otherwise), and $x_{ijk}^{(s)} \geq 0 \triangleq$ the quantity (in thousands) of models i produced and sold from plant j if it is placed in configuration k under scenario s , formulate a two-stage mixed-integer stochastic program in extensive form (Figure 4.4) to choose a maximum expected profit plan for Zoom.
- (c) Use class optimization software to solve your model of (b) and describe the plan that results.

4-50 The Regional Power Alliance (RPA) is a profit-making operator of electric power generators $g = 1, \dots, G$. Each generator g can be utilized or left idle in any of the $t = 1, \dots, T$ months over which it plans its operation. If generator g is utilized, there is a fixed setup cost of f_g each time, and the power it produces must fall between W_g^{min} and W_g^{max} (in megawatt hours (MWh)), with variable cost of c_g per MWh. Decisions about which generators to utilize are made at the beginning of the planning horizon which take into account maintenance and other needs.

RPA planning must address 2 broad markets for its power. The first is “base load” power sold under agreements selected from an available list $k = 1, \dots, K$ at the beginning of the planning

horizon. Such contracts set a price d_k per MWh, and requirements to deliver $r_{k,t}$ MWh in periods t . These base load contracts must be accepted or declined on an all-or-nothing basis.

The second “peak load” part of RPA’s sales comes from offering power not committed to base-load customers in short-term energy markets. The difficulty is that there is uncertainty of what selling price v_t will be available in the market when period t arrives. To account for this uncertainty, RPA has constructed a series of scenarios $s = 1, \dots, S$ having probability $p^{(s)}$ with assumed prices $v_t^{(s)}$ per MWh in time t .

RPA seeks a production and sales plan to maximize its expected total profit over the planning horizon.

- (a) Explain why RPA’s problem can be formulated as a two-stage mixed-integer stochastic program with Stage 1 deciding which generators to utilize and which base-load contracts to adopt, then Stage 2 adding the issue of peak-load power sales in different scenarios.
- (b) Justify why the needed decision variables in your model can be defined as $x_{gt} \triangleq 1$ generator g is operated in period t ($= 0$ otherwise), $y_k \triangleq 1$ if contract k is accepted ($= 0$ otherwise), $w_{gt} \geq 0 \triangleq$ the total power produced (MWh) by generator g in period t , and $z_t^{(s)} \geq 0 \triangleq$ the amount of power (MWh) sold as peak load in period t under scenario s .
- (c) Using the above parameters and decision variables, formulate and justify an expected-value objective function tracking fixed cost of open generators, plus variable cost

of base- and peak-load generation, versus income from fulfilling adopted power agreements, and expected income from sales in the peak-load market.

- (d) Using the above parameters and decision variables, formulate and justify a system of constraints requiring total power produced (both base-load and peak-load markets) from each open generator g and each period t falls between W_g^{min} and W_g^{max} .
- (e) Using the above parameters and decision variables, formulate and justify a system of constraints for each scenario and time period assuring power produced across all open generators fulfills both selected power agreements and peak-load market sales for the scenario.
- (f) Complete an extensive-form statement of the full model, including variable-type constraints.

REFERENCES

- Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.
- Birge, John R. and Francois Louveaux (2010), *Introduction to Stochastic Programming*, Springer, New York, New York.
- Chvátal, Vašek (1980), *Linear Programming*, W.H. Freeman, San Francisco, California.
- Hillier, Fredrick S. and Gerald J. Lieberman (2001), *Introduction to Operations Research*, McGraw-Hill, Boston.
- Taha, Hamdy (2011), *Operations Research - An Introduction*, Prentice-Hall, Upper Saddle River, New Jersey.
- Winston, Wayne L. (2003), *Operations Research - Applications and Algorithms*, Duxbury Press, Belmont California.

Simplex Search for Linear Programming

Having sampled in Chapter 4 some of the enormous variety of linear programming models, it is time to focus on the powerful algorithms that make them tractable. In this chapter we develop a special form of improving search called the **simplex** method. Simplex is the most widely used of all optimization algorithms, although the newer interior-point methods of Chapter 7 are proving strong competitors. Both exploit the special properties of linear programs to produce highly efficient algorithms capable of globally optimizing huge models. Our treatment assumes reader familiarity with the search fundamentals in Chapter 3.

5.1 LP OPTIMAL SOLUTIONS AND STANDARD FORM

We begin our investigation of linear programming algorithms with some observations and conventions that enormously simplify the task of designing efficient computational procedures.

APPLICATION 5.1: TOP BRASS TROPHY

As usual, it will help to have a tiny example at hand. We illustrate with the case of the (fictional) Top Brass Trophy Company, which makes large championship trophies for youth athletic leagues. At the moment they are planning production for fall sports: football and soccer. Each football trophy has a wood base, an engraved plaque, a large brass football on top, and returns \$12 in profit. Soccer trophies are similar except that a brass soccer ball is on top, and the unit profit is only \$9. Since the football has an asymmetric shape, its base requires 4 board feet of wood; the soccer base requires only 2 board feet. At the moment there are 1000 brass footballs in stock, 1500 soccer balls, 1750 plaques, and 4800 board feet of wood. What trophies should be produced from these supplies to maximize total profit assuming that all that are made can be sold?

The decisions to be made in this problem are

$x_1 \triangleq$ number of football trophies to produce

$x_2 \triangleq$ number of soccer trophies to produce

In terms of these decision variables, we can model the problem

$$\begin{aligned}
 \max \quad & 12x_1 + 9x_2 && \text{(profit)} \\
 \text{s.t.} \quad & x_1 &\leq 1000 & \text{(footballs)} \\
 & x_2 &\leq 1500 & \text{(soccer balls)} \\
 & x_1 + x_2 &\leq 1750 & \text{(plaques)} \\
 & 4x_1 + 2x_2 &\leq 4800 & \text{(wood)} \\
 & x_1, x_2 &\geq 0 &
 \end{aligned} \tag{5.1}$$

The objective seeks to maximize total profit, and the main constraints enforce limits on footballs, soccer balls, plaques, and wood, respectively.

Figure 5.1 solves the problem graphically. An optimal solution occurs at $x_1^* = 650$ and $x_2^* = 1100$ with a total profit of \$17,700.

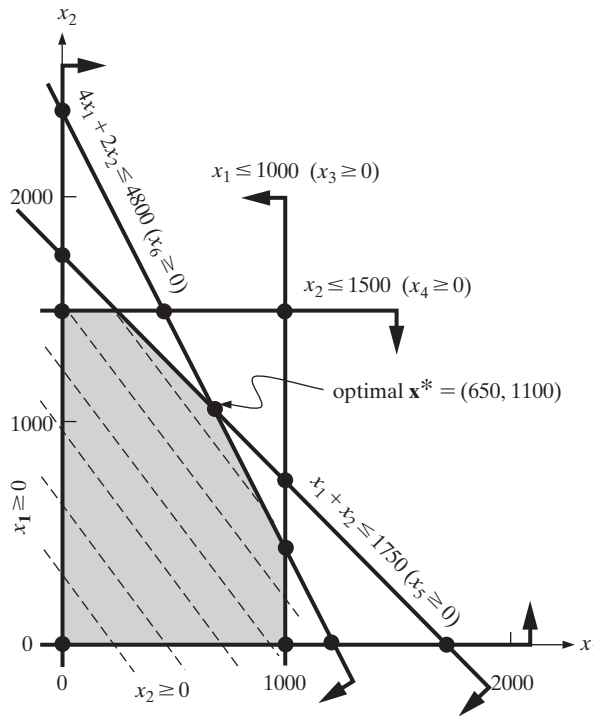


FIGURE 5.1 Graphical Solution of the Top Brass Trophy Example

Model (5.1) is a linear program (definition [4.1](#)) because the single objective function and all constraints involve only weighted sums of the decision variables. Also, all variables are continuous.

Global Optima in Linear Programs

By applying some of the observations in Chapter 3, we can begin to appreciate the elegant model tractability implied by LP definition [4.1](#) (also [2.29](#)). Feasible sets defined by continuous variables and linear constraints are convex (principle [3.32](#)). Combined with linear objective functions, LPs have all the properties convenient for improving search, and principle [3.33](#) applies.

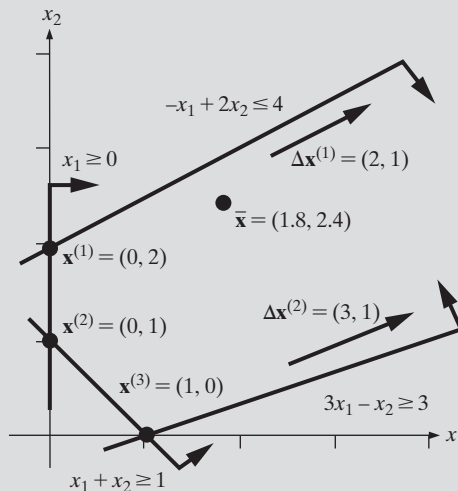
Principle 5.1 Every local optimum for a linear program is a global optimum.

Even more than being convex, feasible sets of linear programs are **polyhedral**. Primer 3 highlights some mathematical properties of polyhedral sets that will prove critical to some of our investigations of LP algorithms.

PRIMER 3: EXTREME POINTS AND DIRECTIONS OF POLYHEDRAL SETS

We have seen in Section 3.4 that a feasible set F is **convex** if for every two points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in F$ the line segment between them $\{(1 - \lambda)\mathbf{x}^{(1)} + \lambda\mathbf{x}^{(2)} \text{ with } 0 \leq \lambda \leq 1\}$ is entirely contained in F . The set is **polyhedral** if it is defined by a set of linear inequalities, or equivalently, if it is the feasible set of a linear program. Every polyhedral set is convex.

The following figure illustrates a polyhedral (and thus convex) feasible set in 2 dimensions:



(Continued)

Polyhedral sets can be characterized by certain distinguished elements:

- A point \mathbf{x} of a polyhedral set F is an **extreme point** if it does not lie within the line segment between any 2 other members of F . All of $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$ in the figure above are extreme. Clearly, any line segment within which any of them lies must have a least one end point outside F .
- A direction $\Delta\mathbf{x}$ is a **direction** of F if for some $\mathbf{x} \in F$, $\mathbf{x} + \mu\Delta\mathbf{x}$ belongs to F for all $\mu \geq 0$. Such a $\Delta\mathbf{x}$ is an **extreme direction** of F if it cannot be written as a combination $\mu_1\mathbf{d}^{(1)} + \mu_2\mathbf{d}^{(2)}$ of other directions $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$ of F with $\mu_1, \mu_2 > 0$. Directions $\Delta\mathbf{x}^{(1)}$ and $\Delta\mathbf{x}^{(2)}$ illustrate for the polyhedral set in the above figure. Adding any positive multiple of either (or both) to a point in F remains feasible for arbitrarily large stepsizes. The positive linear combination $\mu_1\Delta\mathbf{x}^{(1)} + \mu_2\Delta\mathbf{x}^{(2)}$ of the two would also be a direction, but not extreme.

Normally we are entitled to expect every nonempty polyhedral set to have at least one extreme point. Still, there is a technical exception. A **line** is a doubly unbounded set of the form $\{\bar{\mathbf{x}} + \lambda\Delta\mathbf{x}\}$ for $-\infty \leq \lambda \leq +\infty$. A nonempty LP-feasible (i.e. polyhedral) set has at least one extreme point if and only if it contains no line. That is, there must be extreme points unless the polyhedral set is so loosely constrained that a direction can be pursued forever with either positive or negative steps without losing feasibility—highly unlikely for practical LP models.

Elements of a polyhedral set can be completely characterized by its extreme points and extreme directions. Let P index the extreme points $\mathbf{x}^{(j)}$ of given polyhedral set F , and D the extreme directions $\Delta\mathbf{x}^{(k)}$ of F . Then any $\bar{\mathbf{x}}$ belongs to polyhedral convex set F if and only if, the solution can be expressed as

$$(*) \quad \bar{\mathbf{x}} = \sum_{j \in P} \lambda_j \mathbf{x}^{(j)} + \sum_{k \in D} \mu_k \Delta\mathbf{x}^{(k)}$$

where $\lambda_j \geq 0$ for all $j \in P$, $\mu_k \geq 0$ for all $k \in D$ and $\sum_j \lambda_j = 1$. That is, every point in F can be written as a sum = 1 nonnegative weighted sum of extreme points plus a nonnegative weighted sum of extreme directions.

Point $\bar{\mathbf{x}} = (1.8, 2.4)$ is illustrated in the above figure. In terms of $(*)$, it can be expressed as

$$\bar{\mathbf{x}} = 0.5\mathbf{x}^{(1)} + 0.5\mathbf{x}^{(2)} + 0.9\Delta\mathbf{x}^{(1)} = 0.5 \begin{pmatrix} 0 \\ 2 \end{pmatrix} + 0.5 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 0.9 \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.8 \\ 2.4 \end{pmatrix}$$

Interior, Boundary, and Extreme Points

One useful tool is a classification of the points in an LP feasible region. Not all have equal interest.

We first distinguish between points in the interior of the feasible region versus those on the boundary. Precise definitions follow from the fact that the boundary of an LP's feasible set is characterized by active inequality constraints (those satisfied as equality).

Definition 5.2 A feasible solution to a linear program is a **boundary point** if at least one inequality constraint that can be strict for some feasible solutions is satisfied as equality at the given point, and an **interior point** if no such inequalities are active.

The collection of feasible points identified in [5.2] as those for which all inequality constraints that can be strictly satisfied are strict is formally known as the **relative interior** of the feasible region. The small example below illustrates how equality constraints can be encompassed in this relative sense.

All points in feasible set F have $x_3 = 2$; it can never be a strict inequality. Thus the relative interior of the feasible region is the shaded area within the boundaries of the 2-dimensional object shown. Point $(1, 1, 2)$ is interior even though it satisfies the $x_3 = 2$ constraint as equality. Points $(0, 0, 2)$, $(0, 3, 2)$, and $(3, 0, 2)$ are the 3 extreme-points of F , and $(1, 0, 2)$ is a nonextreme boundary point.

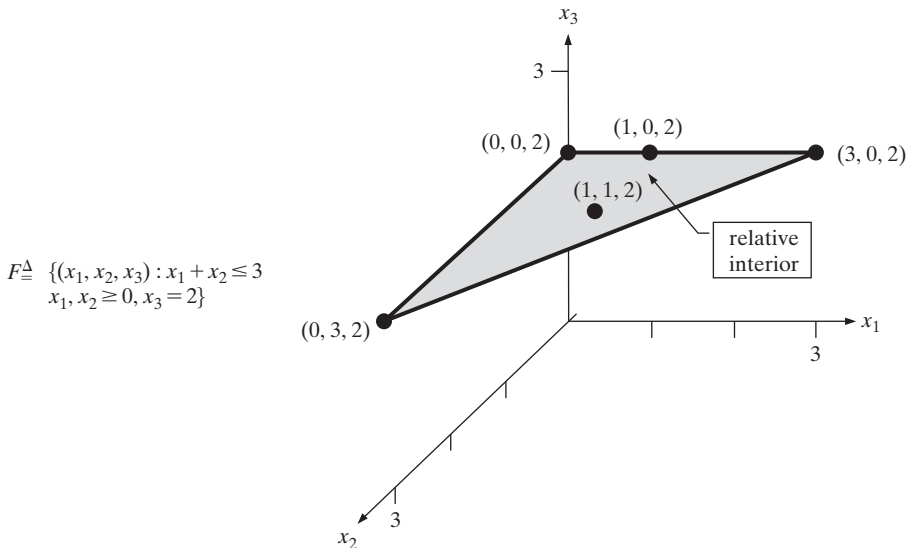


Figure 5.2 labels some specific points in the Top Brass Trophy feasible set. Each of the inequalities has some feasible points that satisfy it as a strict inequality, and solution $\mathbf{x}^{(7)}$ is interior because none of the inequalities is active. All of $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(6)}$ are boundary points because at least one inequality is active at each. For example, inequalities $x_1 \leq 1000$ and $x_2 \geq 0$ are satisfied as equalities at solution $\mathbf{x}^{(1)} = (1000, 0)$, and $x_1 \geq 0$ is active at $\mathbf{x}^{(6)}$. Solutions $\mathbf{x}^{(8)}$ and $\mathbf{x}^{(9)}$ constitute neither boundary nor interior points because both are infeasible.

Extreme points (also called corner points) such as $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(5)}$ of Figure 5.2 are special boundary points so-named because they “stick out.” Primer 3 gives a formal definition. If a point is to form a “corner” of a feasible set, it cannot fall in the middle of any line segment of the set.

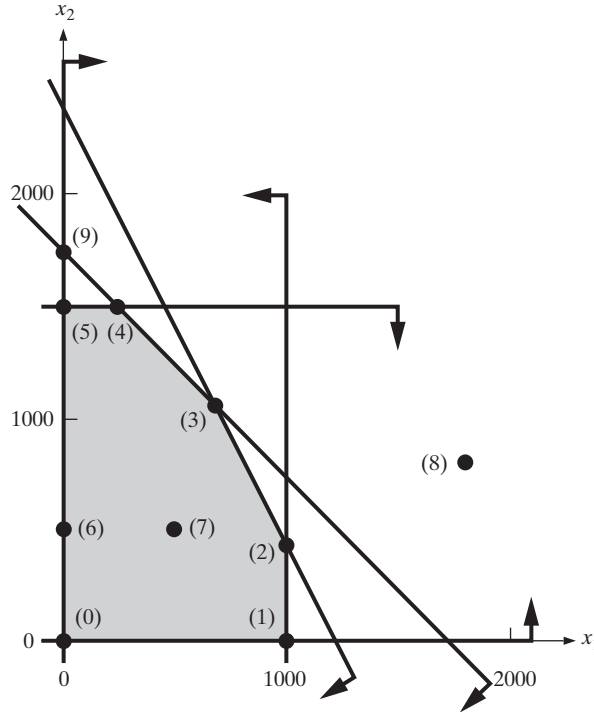
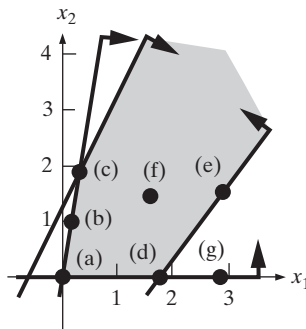


FIGURE 5.2 Interior, Boundary and Extreme Points of the Top Brass Example

Notice that points $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(5)}$ in Figure 5.2 all fulfill this requirement. They can form endpoints of line segments within the set, but never midpoints. Contrast with nonextreme feasible solutions $\mathbf{x}^{(6)} = (0, 500)$ and $\mathbf{x}^{(7)} = (500, 500)$. The line segment from $\mathbf{x}^{(0)}$ to $\mathbf{x}^{(5)}$ passes through $\mathbf{x}^{(6)}$, and the segment from $\mathbf{x}^{(6)}$ to $\mathbf{x} = (501, 500)$ includes $\mathbf{x}^{(7)}$.

EXAMPLE 5.1: CLASSIFYING FEASIBLE POINTS

Classify the labeled solutions as interior, boundary, and/or extreme points of the following LP feasible region:



Solution: Points (a), (c), and (d) are both extreme points and boundary points, because they do not lie along the line segment between any two other feasible points. Points (b) and (e) are boundary points that are not extreme; each makes at least one constraint active, but these points do fall within a line segment joining others. Point (f) is interior because no constraint is active. Point (g) is neither interior nor boundary (nor extreme) because it is infeasible.

Optimal Points in Linear Programs

In order to structure efficient improving search algorithms for LP, we need to know a bit more about the kinds of feasible points that can be optimal.

Principle 5.3 Unless the objective function is constant (same for all solutions), every optimal solution to an LP will occur at a boundary point of its feasible set.

For example, the optimal solution to our Top Brass example (see Figure 5.1) occurs at boundary (and extreme point) solution $\mathbf{x}^* = (650, 1100)$.

To gain insight about why LP optima over non constant objectives always occur on the boundary, consider the case where all constraints are inequalities that may be strict at some feasible points. Then we can make at least a small move in any direction from an interior point where none of the constraints are active without losing feasibility. Also, with a (non constant) linear objective function, the objective function coefficient vector $\Delta \mathbf{x} = \mathbf{c}$ is always an improving direction for a maximize problem (principle [3.21](#)), and $\Delta \mathbf{x} = -\mathbf{c}$ improves for a minimize problem (principle [3.22](#)).

Matters become more complex with equality constraints present, but the result is the same for every LP model with a non constant objective. No interior point can be optimal.

What about **unique optimal solutions**? Intuition suggests that if a feasible point is to be the only optimal solution to a linear program, it must somehow “stick out” farther than other feasible points. This is true.

Principle 5.4 If a linear program has a unique optimal solution, that optimum must occur at an extreme point of the feasible region.

For a more mathematical argument, consider an optimal solution \mathbf{x}^* to a maximizing LP over objective function $\mathbf{c} \cdot \mathbf{x}$. If \mathbf{x}^* is not an extreme point of the feasible set, then (Primer 3) it must be the weighted average of two other feasible solutions $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. That is,

$$\begin{aligned}\mathbf{x}^* &= (1 - \lambda)\mathbf{x}^{(1)} + \lambda\mathbf{x}^{(2)} \text{ where } 0 < \lambda < 1 \text{ and} \\ \mathbf{c} \cdot \mathbf{x}^* &= (1 - \lambda)\mathbf{c} \cdot \mathbf{x}^{(1)} + \lambda\mathbf{c} \cdot \mathbf{x}^{(2)}\end{aligned}$$

If the objective function values at the two end points differ, one must be higher than their average $\mathbf{c} \cdot \mathbf{x}^*$, and \mathbf{x}^* cannot be optimal. If the objective function values at the end points are equal, then all solutions along the line between them achieve the optimal value; there are multiple optimal solutions, and \mathbf{x}^* cannot be unique. We conclude that an LP solution can be a unique optimum only if it is an extreme point of the feasible set.

Of course, the LP may be infeasible or unbounded, and there may be nonextreme alternative optimal solutions along the boundary. But if a linear program has any optimal solutions at all, it is easy to see at least one will occur at an extreme.

Principle 5.5 | If a linear program has any optimal solution, it has one at an extreme point of its feasible region.

To see why, think of forming the set of all optimal points for a given LP over maximizing objective function $\mathbf{c} \cdot \mathbf{x}^*$. This is just another polyhedral set over all the model's constraints plus one more that $\mathbf{c} \cdot \mathbf{x} = v^*$, where $v^* \triangleq$ the optimal solution value. Assuming the given LP feasible set meets the no line technical requirement of Primer 3, its optimal subset also meets the requirement and has an extreme point solution. Will that solution also be extreme in the underlying LP? It has to be, because if it did lay along a feasible line segment, at least one of the end points would have to be outside the optimal set, and their weighted average could not be optimal.

Principle 5.5 is fundamental because of the flexibility it provides in designing improving search algorithms for linear programming. Knowing that there will be an optimal solution at an extreme point if there is any optimal solution at all, we are free to restrict our search to extreme-point solutions. The simplex algorithm, which is the main topic of this chapter, does exactly that.

EXAMPLE 5.2: IDENTIFYING OPTIMAL POINTS

Indicate which of the labeled points in Example 5.1 can be optimal or uniquely optimal for any objective function.

Solution: Following principle 5.4, extreme points (a), (c), and (d) can be optimal or uniquely optimal. Boundary points (b) and (e) can also be optimal. Still, neither can be uniquely optimal because any objective making either optimal will also have extreme-point optimal solutions. Interior point (f) cannot be optimal for any non-constant objective function; feasible improvement is always possible. Point (g) is not even feasible.

LP Standard Form

A model can be a linear program even if variables are subject to a variety of (continuous) variable-type constraints, main constraints are mixtures of inequalities and equalities, and expressions are nested through several levels of parentheses on both sides of \geq , \leq , and $=$ signs. Still, it will be much easier to discuss solution methods if we settle on an LP standard form.

Definition 5.6 Linear programs in **standard form** (1) have only equality main constraints; (2) have only nonnegative variables, and (3) have objective function and main constraints simplified so that variables appear at most once, on the left-hand side, and any constant term (possibly zero) appears on the right-hand side.

Converting Inequalities to Nonnegativities with Slack Variables

We will clearly have to do some rearranging to fit every “raw” linear program into standard form. For example, Top Brass model (5.1) includes inequalities among its main constraints. Standard form [5.6] allows inequalities only in the form of nonnegativity variable-type constraints.

To accomplish the needed transformation we introduce new **slack variables** in each main inequality that consumes the difference between left- and right-hand sides.

Principle 5.7 Main inequality constraints of a given linear program can be converted into nonnegativities by adding distinct, nonnegative, zero-cost slack variables in every such \leq inequality and subtracting such slack variables in every main \geq .

Applying principle [5.7] to the 4 main inequalities of Top Brass model (5.1), we add slacks x_3, \dots, x_6 . The result is the standard-form model

$$\begin{array}{ll} \max & 12x_1 + 9x_2 \\ \text{s.t.} & x_1 + x_3 = 1000 \\ & x_2 + x_4 = 1500 \\ & x_1 + x_2 + x_5 = 1750 \\ & 4x_1 + 2x_2 + x_6 = 4800 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{array}$$

Notice that a different slack was used in each constraint, nonnegativity limits apply to each slack variable, and the slacks do not appear in the objective function. All slack variables carry plus signs in this example (compare with Example 5.3 below) because all the modified inequalities were of the \leq form.

Even though our standard form now has 6 variables versus an original 2, we have not really changed the model. Labels in Figure 5.1 show that each original inequality constraint corresponds to some slack’s nonnegativity in standard form. For example, the final main constraint

$$4x_1 + 2x_2 \leq 4800$$

becomes nonnegativity on slack variable x_6 . Under standard-form equality

$$4x_1 + 2x_2 + x_6 = 4800$$

the inequality holds exactly when

$$x_6 = 4800 - 4x_1 - 2x_2 \geq 0$$

Also, with their coefficients in the objective function = 0, slack variables have no impact on cost.

Why bother with this modified version of a perfectly good linear program? To see, we must look back to the development of feasible move directions in Section 3.5. Most of the complexity in dealing with feasible directions relates to keeping track of active inequality constraints (ones satisfied as equalities). Equality constraints are always active, but inequalities may be active at one moment in a search and inactive at the next. Introducing slack variables as in principle [5.7] does not eliminate any inequalities, but it does convert them to the simplest possible form which will simplify analysis.

EXAMPLE 5.3: INTRODUCING SLACK VARIABLES

Introduce slack variables and simplify to place each of the following linear programs in standard form.

- (a) $\min \quad 9w_1 + 6w_2$
 s.t. $2w_1 + w_2 \geq 10$
 $w_1 \leq 50$
 $w_1 + w_2 = 40$
 $100 \geq w_1 + 2w_2 \geq 15$
 $w_1, w_2 \geq 0$
- (b) $\max \quad 15(2x_1 + 8x_2) - 4x_3$
 s.t. $2(10 - x_1) + x_2 + 5(9 - x_3) \geq 10$
 $x_1 + 2x_3 \leq x_3$
 $2x_2 + 18x_3 = 50$
 $x_1, x_2, x_3 \geq 0$

Solution:

(a) We introduce slack variables w_3, w_4, w_5, w_6 as in principle [5.7] to convert the 4 main inequalities to nonnegativities. The result is the standard-form linear program

$$\begin{aligned} \min \quad & 9w_1 + 6w_2 \\ \text{s.t.} \quad & 2w_1 + w_2 - w_3 = 10 \\ & w_1 + w_4 = 50 \\ & w_1 + w_2 = 40 \\ & w_1 + 2w_2 + w_5 = 100 \\ & w_1 + 2w_2 - w_6 = 15 \\ & w_1, w_2, w_3, w_4, w_5, w_6 \geq 0 \end{aligned}$$

Notice that the last two main constraints, which were written together in the original model, are separated in standard form. Slacks are added in \leq inequalities and subtracted in \geq forms. No slack is required in equality constraints.

(b) We begin by simplifying to collect variable terms on the left-hand side and constants on the right [(3) of definition 5.6].

$$\begin{array}{llll} \max & 30x_1 + 120x_2 - 4x_3 & & \\ \text{s.t.} & -2x_1 + x_2 - 5x_3 & \geq & -55 \\ & x_1 + x_3 & \leq & 0 \\ & 2x_2 + 18x_3 & = & 50 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Now slacks are introduced according to principle 5.7 to complete the standard form

$$\begin{array}{llll} \max & 30x_1 + 120x_2 - 4x_3 & & \\ \text{s.t.} & -2x_1 + x_2 - 5x_3 & -x_4 = & -55 \\ & x_1 + x_3 & +x_5 = & 0 \\ & 2x_2 + 18x_3 & = & 50 \\ & x_1, x_2, x_3, x_4, x_5 & \geq & 0 \end{array}$$

Converting Nonpositive and Unrestricted Variables to Nonnegative

Like most applied linear programs, the Top Brass Trophy example employs only nonnegative decision variables. Condition (2) of standard format 5.6 is fulfilled automatically.

Still, LPs do occasionally involve variables such as net income or temperature that can feasibly take on negative values. Such variables may be **nonpositive** (i.e., subject to ≤ 0 variable-type constraints) or **unrestricted** by sign. The latter are often designated **URS**, meaning “unrestricted sign.”

We can convert linear programs with nonpositive and unrestricted variables to standard form simply by changes of variables. For nonpositive variables, the change substitutes the negative.

Principle 5.8 Nonpositive variables in linear programs can be eliminated by substituting new variables equal to their negatives.

For example, in a model with original decision variables x_1, \dots, x_{10} and nonpositive-type restriction

$$x_7 \leq 0$$

we would introduce the new variable

$$x_{11} = -x_7$$

and substitute $-x_{11}$ everywhere that x_7 appears. In particular, the nonpositivity constraint

$$x_7 \leq 0 \text{ becomes } -x_{11} \leq 0 \text{ or } x_{11} \geq 0$$

The handling of unrestricted variables is slightly less obvious. How can non-negative variables model a quantity that can take on any sign? The answer is to introduce two new nonnegative variables and consider their difference.

Principle 5.9 Unrestricted (or URS) variables in linear programs can be eliminated by substituting the difference of two new nonnegative variables.

For example, a model with variables y_1, \dots, y_7 and

$$y_1 \text{ URS}$$

can be placed in standard form by introducing two new variables $y_8, y_9 \geq 0$. Everywhere y_1 appears in the given model we then substitute

$$y_1 = y_8 - y_9$$

EXAMPLE 5.4: CONVERTING NONPOSITIVE AND UNRESTRICTED VARIABLES

Make suitable variable changes to convert the following linear program to standard form:

$$\begin{array}{ll} \min & -9w_1 + 4w_2 + 16w_3 - 11w_4 \\ \text{s.t.} & w_1 + w_2 + w_3 + w_4 = 100 \\ & 3w_1 - w_2 + 6w_3 - 2w_4 = 200 \\ & w_1 \geq 0, w_2 \leq 0 \end{array}$$

Solution: Variable w_1 is already nonnegative, but we must substitute for nonpositive variable w_2 and for URS variables w_3 and w_4 which are subject to no variable-type constraints. Following principles [5.8](#) and [5.9](#), we employ

$$\begin{aligned} w_2 &= -w_5 \\ w_3 &= w_6 - w_7 \\ w_4 &= w_8 - w_9 \end{aligned}$$

After simplification the resulting standard-form linear program is

$$\begin{array}{ll} \min & -9w_1 - 4w_5 + 16w_6 - 16w_7 - 11w_8 + 11w_9 \\ \text{s.t.} & w_1 - w_5 + w_6 - w_7 + w_8 - w_9 = 100 \\ & 3w_1 - w_5 + 6w_6 - 6w_7 - 2w_8 + 2w_9 = 200 \\ & w_1, w_5, w_6, w_7, w_8, w_9 \geq 0 \end{array}$$

Notice that $w_2, w_3,$ and w_4 have been eliminated from the model completely.

Standard Notation for LPs

Once a linear program has been placed in standard form, its key elements are so neatly sorted out that we can begin to think of the model as a collection of coefficients. Widely used notation gives all of these elements standard names.

Definition 5.10 | Standard notation for linear programs is

- $x_j \triangleq$ j th decision variable
- $c_j \triangleq$ cost or objective function coefficient of x_j
- $a_{i,j} \triangleq$ constraint coefficient of x_j in the i th main constraint
- $b_i \triangleq$ right-hand-side (or RHS) constant term of main constraint i
- $m \triangleq$ number of main (equality) constraints
- $n \triangleq$ number of decision variables

Then, LP standard form of every linear program has the generic format

$$\begin{array}{ll} \min \text{ (or max) } & \sum_{j=1}^n c_j x_j \\ \text{s.t.} & \sum_{j=1}^n a_{i,j} x_j = b_i \quad \text{for all } i = 1, 2, \dots, m \\ & x_j \geq 0 \quad \text{for all } j = 1, 2, \dots, n \end{array}$$

It will often be convenient to write standard-form linear programs in an even more compact way using notions of vectors and matrices. Primer 4 reviews some of the main ideas of matrix arithmetic for those who need it.

Definition 5.11 | In the usual matrix notation, **LP standard form** is

$$\begin{array}{ll} \min \text{ (or max) } & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

PRIMER 4: MATRICES, MATRIX ARITHMETIC, AND TRANSPOSES

Matrices are 2-dimensional arrays of numbers, with the **dimension** of the array being described in terms of its number of rows and columns. For example,

$$\mathbf{Q} = \begin{pmatrix} 2 & 0 & -\frac{7}{5} \\ 0 & -1.2 & 3 \end{pmatrix} \quad \text{and} \quad \mathbf{R} = \begin{pmatrix} 12 & -2 & \frac{7}{5} \\ 1 & 0 & -2 \end{pmatrix}$$

are 2 by 3 matrices (2 rows and 3 columns). See Primer 1 for the related 1-dimensional notion of vectors.

(Continued)

In this book matrices are always denoted by uppercase (capital) boldface symbols (e.g., \mathbf{A} , \mathbf{R} , Σ). Their entries are indicated by corresponding lowercase italic symbols ($a_{i,j}$, $r_{2,6}$, $\sigma_{3,9}$) having one subscript for the row index and one for the column. Thus matrix \mathbf{Q} above has $q_{1,2} = 0$ and $q_{2,3} = 3$.

Just as with vectors, matrices of like dimension are added, subtracted, and multiplied by a scalar in component by component fashion. Thus for \mathbf{Q} and \mathbf{R} above,

$$\mathbf{Q} + \mathbf{R} = \begin{pmatrix} 14 & -2 & 0 \\ 1 & -1.2 & 1 \end{pmatrix} \quad \text{and} \quad -.3\mathbf{R} = \begin{pmatrix} -3.6 & 0.6 & -0.42 \\ -0.3 & 0 & 0.6 \end{pmatrix}$$

Also like vectors, multiplication of one matrix by another is defined in a somewhat nonintuitive way convenient for expressing linear combinations. Matrices \mathbf{P} and \mathbf{A} can be multiplied as $\mathbf{D} = \mathbf{PA}$ if the number of columns in \mathbf{P} is the same as the number of rows in \mathbf{A} . Then the i, j component of the result is defined as the dot product of row i of \mathbf{P} and column j of \mathbf{A} (i.e., $d_{i,j} = \sum_k p_{i,k} a_{k,j}$). For example, with

$$\mathbf{P} = \begin{pmatrix} 1 & 3 \\ -1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{A} = \begin{pmatrix} 5 & -1 & 0 \\ 2 & 9 & 4 \end{pmatrix}, \quad \mathbf{PA} = \begin{pmatrix} 11 & 26 & 12 \\ -1 & 19 & 8 \end{pmatrix}$$

Notice that the order of multiplication of matrices matters. Product \mathbf{AP} is not defined for the \mathbf{P} and \mathbf{A} above because the number of columns of \mathbf{A} does not equal the number of rows of \mathbf{P} . Even if matrices admit multiplication in both orders, the results can be different.

Multiplication of a matrix by a vector is defined as if the vector were a row or column matrix, whichever is appropriate. Thus if $\mathbf{v} = (-1, 4)$, $\mathbf{x} = (2, 1, 2)$, and \mathbf{A} is as above,

$$\mathbf{vA} = (-1, 4) \begin{pmatrix} 5 & -1 & 0 \\ 2 & 9 & 4 \end{pmatrix} = (3, 37, 16)$$

$$\mathbf{Ax} = \begin{pmatrix} 5 & -1 & 0 \\ 2 & 9 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 9 \\ 21 \end{pmatrix}$$

In the first computation \mathbf{v} was treated as a row matrix because it premultiplies matrix \mathbf{A} . In the second computation \mathbf{x} plays the role of a column matrix because it postmultiplies \mathbf{A} .

In some cases, it is also convenient to work with the **transpose** of matrix \mathbf{M} , denoted \mathbf{M}^T , where rows and columns have been interchanged. For example,

$$\mathbf{Q} = \begin{pmatrix} 2 & -\frac{1}{3} & 7 \\ 0 & 6 & 13 \end{pmatrix} \quad \text{has transpose} \quad \mathbf{Q}^T = \begin{pmatrix} 2 & 0 \\ -\frac{1}{3} & 6 \\ 7 & 13 \end{pmatrix}$$

A matrix is **symmetric** if transposing leaves it unchanged, and **nonsymmetric** otherwise. Thus with

$$\mathbf{R} = \begin{pmatrix} 3 & -1 \\ 6 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{S} = \begin{pmatrix} 3 & 1 & 8 \\ 1 & -11 & 0 \\ 8 & 0 & 25 \end{pmatrix}$$

\mathbf{R} is nonsymmetric, because $\mathbf{R} \neq \mathbf{R}^T$, but $\mathbf{S} = \mathbf{S}^T$ is symmetric.

When matrices can be multiplied the transpose of the product is the product of the transposes after the sequence of multiplication is reversed. For example, with the \mathbf{Q} and \mathbf{R} above,

$$\begin{aligned} (\mathbf{RQ})^T &= \left(\begin{pmatrix} 3 & -1 \\ 6 & 0 \end{pmatrix} \begin{pmatrix} 2 & -\frac{1}{3} & 7 \\ 0 & 6 & 13 \end{pmatrix} \right)^T = \begin{pmatrix} 6 & -7 & 8 \\ 12 & -2 & 42 \end{pmatrix}^T \\ &= \begin{pmatrix} 6 & 12 \\ -7 & -2 \\ 8 & 42 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ -\frac{1}{3} & 6 \\ 7 & 13 \end{pmatrix} \begin{pmatrix} 3 & 6 \\ -1 & 0 \end{pmatrix} = \mathbf{Q}^T \mathbf{R}^T \end{aligned}$$

Cost or objective function vector \mathbf{c} presents all objective function coefficients c_j , **constraint matrix \mathbf{A}** displays all main constraint coefficients $a_{i,j}$, and **right-hand-side (RHS) vector \mathbf{b}** shows constant terms b_i of the constraints. As usual, \mathbf{x} is the vector of decision variables x_j .

We can illustrate with the Top Brass Trophy model and its $m = 4$ main constraints. In standard form the Top Brass decision vector has the form $\mathbf{x} = (x_1, x_2, \dots, x_6)$, with $n = 6$ components. Corresponding coefficient arrays are n -vector \mathbf{c} , m by n matrix \mathbf{A} , and m -vector \mathbf{b} :

$$\begin{aligned} \mathbf{c} &= (12 \quad 9 \quad 0 \quad 0 \quad 0 \quad 0) \\ \mathbf{A} &= \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 4 & 2 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1000 \\ 1500 \\ 1750 \\ 4800 \end{pmatrix} \end{aligned}$$

EXAMPLE 5.5: USING STANDARD LP MATRIX NOTATION

Return to the standard-form linear program of Example 5.3(b). Identify the m , n , \mathbf{A} , \mathbf{b} , and \mathbf{c} of standard matrix representation [\[5.11\]](#).

Solution: The standard form model of Example 5.3(b) has $m = 3$ main constraints and $n = 5$ variables. Corresponding coefficient arrays are

$$\begin{aligned} \mathbf{c} &= (30 \quad 120 \quad -4 \quad 0 \quad 0) \\ \mathbf{A} &= \begin{pmatrix} -2 & 1 & -5 & -1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 2 & 18 & 0 & 0 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} -55 \\ 0 \\ 50 \end{pmatrix} \end{aligned}$$

5.2 EXTREME-POINT SEARCH AND BASIC SOLUTIONS

One can fashion efficient algorithms for linear programs that pass through the interior of the feasible set (see Chapter 7). Still, since there is an extreme-point optimal solution if there is any at all (definition [5.6](#)), we are free to limit our search to extreme points of an LP's feasible region.

Simplex is just such an extreme-point search algorithm. In this section we develop some underlying concepts that lead to the full algorithm in Section 5.3.

Determining Extreme Points with Active Constraints

To begin, we need a more convenient characterization of extreme points. Extreme points are boundary points, and every part of the boundary of an LP feasible space is formed by one or more active constraints. What makes extreme-point solutions special is that enough constraints are active to determine a point completely.

Principle 5.12 Every extreme-point solution to a linear program is determined by a set of inequality constraints simultaneously active only at that solution. In standard form all are nonnegativity constraints on original and slack variables taking value = 0.

For example, extreme-point solution $\mathbf{x}^{(5)}$ in Top Brass Figure 5.2 is determined completely by the two constraints active there, $x_1 \geq 0$ and $x_2 \leq 1500$. Contrast this with point $\mathbf{x}^{(6)}$, where only $x_1 \geq 0$ is active. This is not enough to determine a point because many solutions have $x_1 = 0$.

To see more fully how active inequalities relate to extreme points, we need more dimensions. Figure 5.3 depicts the search of a 3-dimensional case maximizing x_3 . The boundary area where each main inequality is active now forms one of the 2-dimensional faces labeled *A–L*. Extreme points are determined as in principle [5.8](#) by collections of three active constraints.

Inequalities *I* and *J*, plus nonnegativity $x_3 \geq 0$, determine $\mathbf{x}^{(0)}$. Notice, however, that some extreme points can be determined in several ways. For example, any 3 of the inequalities for surfaces *C*, *D*, *G*, and *J* determine the extreme point labeled $\mathbf{x}^{(2)}$.

EXAMPLE 5.6: DETERMINING EXTREME POINTS

List all sets of 3 active constraints determining points $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$, and $\mathbf{x}^{(5)}$ in Figure 5.3.

Solution: Constraints *B*, *C*, *G*, *H*, and *I* are all active at $\mathbf{x}^{(3)}$. Any 3 of these 5 determine the point. Solution $\mathbf{x}^{(4)}$ is determined uniquely by constraints *A*, *B*, and *C*. Extreme point $\mathbf{x}^{(5)}$ is determined uniquely by inequalities *A* and *C*, plus $x_2 \geq 0$.

Adjacent Extreme Points and Edges

Improving search algorithms move from one solution to another nearby. We can define neighboring or adjacent extreme points in terms of their determining sets of active constraints.

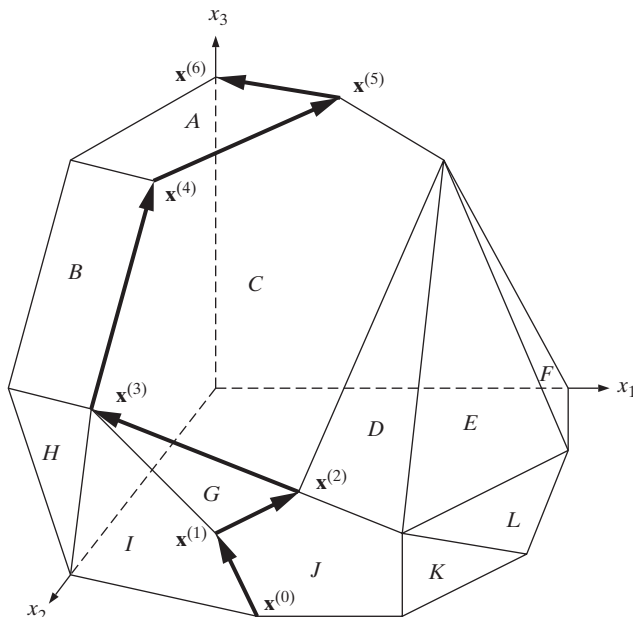


FIGURE 5.3 Adjacent Extreme Point Search to Max x_3 in Three Dimensions

Definition 5.13 | Extreme points of an LP-feasible space are **adjacent** if they are determined by active constraint sets differing in only one element.

Again Figure 5.3 illustrates. Extreme points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are adjacent because the first is determined by active inequalities $G, I,$ and $J,$ while $D, G,$ and J provide a determining set for the second. These two lists have all but one member in common.

Contrast with $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(4)}$, which are not adjacent. Point $\mathbf{x}^{(4)}$ is determined by active inequalities $A, B,$ and $C.$ Of these, only C is also active at $\mathbf{x}^{(2)}$; no choice of a determining set for $\mathbf{x}^{(2)}$ can satisfy definition [5.13](#).

Line segments joining adjacent extreme points in Figure 5.3 are called edges.

Definition 5.14 | An **edge** of the feasible region for a linear program is a 1-dimensional set of feasible points along a line determined by a collection of active constraints.

Adjacent extreme points are joined by an edge determined by the active constraints the extreme points have in common. For example, inequalities for surfaces $G, I,$ and J determine the extreme point labeled $\mathbf{x}^{(1)}$. Keeping just G and J active produces the edge joining $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. Similarly, if I and G are kept active, we obtain the edge between adjacent extreme points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(3)}$.

EXAMPLE 5.7: IDENTIFYING EDGES AND ADJACENT EXTREME POINTS

Consider the linear program with feasible set delimited by

$$-2x_1 + 3x_2 \leq 6 \quad (5.2)$$

$$-x_1 + x_2 \leq 1 \quad (5.3)$$

$$x_1 \geq 0 \quad (5.4)$$

$$x_2 \geq 0 \quad (5.5)$$

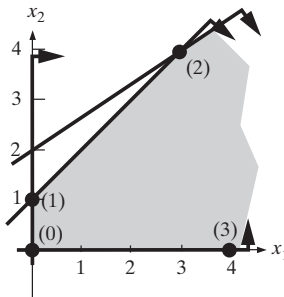
(a) Sketch the feasible region and identify points $\mathbf{x}^{(0)} = (0, 0)$, $\mathbf{x}^{(1)} = (0, 1)$, $\mathbf{x}^{(2)} = (3, 4)$, and $\mathbf{x}^{(3)} = (4, 0)$.

(b) Determine which pairs of those points are adjacent extreme points.

(c) Determine which pairs of the points are joined by an edge.

Solution:

(a) The feasible region is as follows:



(b) Applying principle [5.12], extreme points must be determined by active constraints. Solutions $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$, and $\mathbf{x}^{(2)}$ are thus extreme points determined by active constraints (5.5)–(5.6), (5.4)–(5.5), and (5.3)–(5.4), respectively. Point $\mathbf{x}^{(3)}$ is not an extreme point.

In two dimensions all-but-one condition [5.12] makes extreme points adjacent if they have one active constraint in common. Thus $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ are adjacent, as are $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, but not $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(2)}$.

(c) Each adjacent pair of part (b) is joined by an edge determined by their common active constraint. For example, $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are joined by the edge along line

$$-x_1 + x_2 = 1$$

Although $\mathbf{x}^{(3)}$ is not an extreme point, the edge defined by line

$$x_2 = 0$$

joins it to $\mathbf{x}^{(0)}$.

Basic Solutions

We saw in Section 5.1 how LP standard form encodes every model as

$$\begin{aligned} \min \text{ (or max) } & \sum_{j=1}^n c_j x_j \\ \text{s.t.} & \sum_{j=1}^n a_{i,j} x_j = b_i \quad \text{for all } i = 1, 2, \dots, m \\ & x_j \geq 0 \quad \text{for all } j = 1, 2, \dots, n \end{aligned}$$

One effect is to reduce all inequalities to nonnegativity constraints.

It follows from principle [5.12](#) that extreme-point solutions to linear programs in standard form are determined by sets of active nonnegativity constraints. Enough variables must be fixed to $= 0$ (making the corresponding nonnegativity constraints active) to uniquely determine all other components of a solution.

Basic solutions are produced in just this way.

Definition 5.15 A **basic solution** to a linear program in standard form is one obtained by fixing just enough variables to $= 0$ that the model's equality constraints can be solved uniquely for the remaining variable values. Those variables fixed are called **nonbasic** and the ones obtained by solving the equalities are termed **basic**.

We can illustrate with the standard form of our Top Brass example (Section 5.1):

$$\begin{aligned} \max & 12x_1 + 9x_2 \\ \text{s.t.} & + x_1 + x_3 = 1000 \\ & + x_2 + x_4 = 1500 \\ & + x_1 + x_2 + x_5 = 1750 \\ & + 4x_1 + 2x_2 + x_6 = 4800 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned} \tag{5.6}$$

Figure 5.4 again shows the feasible set, with constraints labeled by the applicable nonnegativity constraint of standard form.

One basic solution is obtained by choosing x_1, x_2, x_3, x_4 to be basic and x_5, x_6 to be nonbasic. Fixing $x_5 = x_6 = 0$ leaves the equality system

$$\begin{aligned} & + x_1 + x_3 = 1000 \\ & + x_2 + x_4 = 1500 \\ & + x_1 + x_2 + (0) = 1750 \\ & + 4x_1 + 2x_2 + (0) = 4800 \end{aligned} \tag{5.7}$$

The unique solution is $x_1 = 650$, $x_2 = 1100$, $x_3 = 350$, and $x_4 = 400$. Thus the full basic solution is $\mathbf{x} = (650, 1100, 350, 400, 0, 0)$.

Notice that this standard-form solution corresponds to extreme point $\mathbf{x}^{(3)}$ in Figure 5.4. This should come as no surprise, because $\mathbf{x}^{(3)}$ is defined by making active the inequalities corresponding to $x_5 \geq 0$ and $x_6 \geq 0$. That is, the point is defined by setting $x_5 = x_6 = 0$.

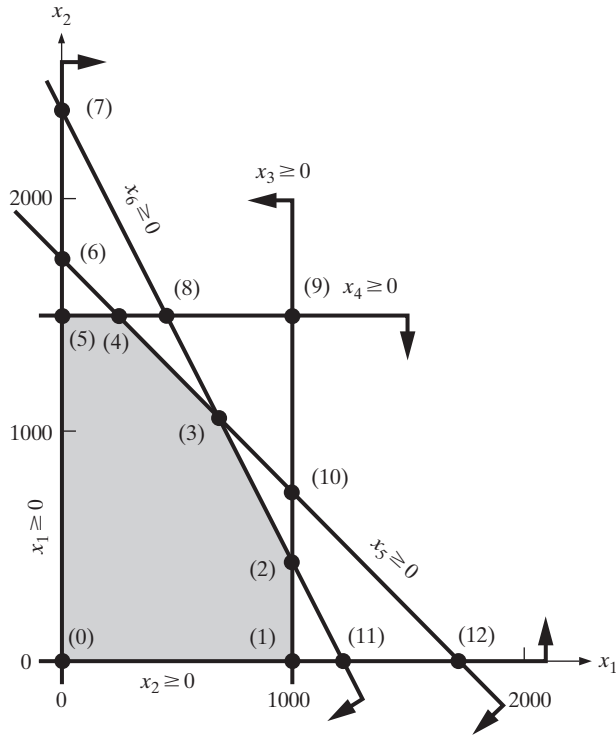


FIGURE 5.4 Basic Solutions of the Top Brass Example

EXAMPLE 5.8: COMPUTING BASIC SOLUTIONS

Suppose that a linear program in standard form has the constraints

$$4x_1 - x_2 + x_3 = 1$$

$$3x_1 + 2x_2 - 2x_3 = 9$$

$$x_1, x_2, x_3 \geq 0$$

Compute the basic solution corresponding to x_1 and x_2 basic.

Solution: The only nonbasic variable will be x_3 . Setting it to zero as in definition 5.15 gives

$$4x_1 - x_2 + (0) = 1$$

$$3x_1 + 2x_2 - 2(0) = 9$$

The remaining 2 equations in 2 unknowns have a unique solution $x_1 = 1, x_2 = 3$. Thus the full basic solution is $\mathbf{x} = (1, 3, 0)$.

Existence of Basic Solutions

It is tempting to believe that we can form basic solutions by setting any collection of nonbasic variables to $= 0$ (i.e., by making any collection of nonnegativity constraints active). Not so! For example, fixing only $x_4 = 0$ in the Top Brass standard-form model leaves the equation system

$$\begin{array}{rcccccc} + x_1 & & + x_3 & & & = 1000 \\ & + x_2 & & + (0) & & = 1500 \\ + x_1 & + x_2 & & & + x_5 & = 1750 \\ +4x_1 & + 2x_2 & & & & + x_6 = 4800 \end{array}$$

There remain 5 unknowns in only 4 equations. Geometrically, this simply reflects the fact that only making active the inequality corresponding to $x_4 \geq 0$ in Figure 5.4 does not fully determine a point.

We are no better off if we fix $x_2 = x_4 = 0$. The resulting system

$$\begin{array}{rcccccc} + x_1 & & + x_3 & & & = 1000 \\ & + (0) & & + (0) & & = 1500 \\ + x_1 & + (0) & & & + x_5 & = 1750 \\ +4x_1 & + 2(0) & & & & + x_6 = 4800 \end{array} \quad (5.8)$$

now has 4 equations in 4 unknowns, but it has no solutions at all (look carefully at the second equation). This establishes algebraically what is apparent in Figure 5.4. Making active the inequalities corresponding to $x_2 \geq 0$ and $x_4 \geq 0$ does not determine a point.

To characterize when basic solutions do exist, we must draw on the algebra of systems of simultaneous equations. Primer 5 provides a quick review. One condition assuring a system with a unique solution gives basic solutions their name:

Principle 5.16 | A basic solution exists if and only if the columns of equality constraints corresponding to basic variables from a basis, that is, a largest possible linearly independent collection.

Systems (5.7) and (5.8) illustrate the possibilities. We can verify that the constraint columns for the basic variables x_1, x_2, x_3 , and x_4 in (5.7) form a basis by checking that the corresponding matrix is nonsingular, that is,

$$\det \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 4 & 2 & 0 & 0 \end{pmatrix} = -2 \neq 0$$

It follows that the equations have a unique solution.

On the other hand, the columns for basic variables x_1, x_3, x_5 , and x_6 in (5.8) do not produce a unique solution because they are linearly dependent; the column for x_1 is easily expressed as

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 4 \end{pmatrix} = 1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + 4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

EXAMPLE 5.9: CHECKING EXISTENCE OF BASIC SOLUTIONS

The following are the constraints of a linear program in standard form:

$$\begin{aligned} 4x_1 - 8x_2 - x_3 &= 15 \\ x_1 - 2x_2 &= 10 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Check whether basic solutions exist for each of the following possible sets of basic variables: (a) x_1, x_2 ; (b) x_1, x_3 ; (c) x_1 ; (e) x_1, x_2, x_3 .

Solution:

(a) Column vectors for x_1 and x_2 form a linearly dependent set because

$$\begin{pmatrix} 4 \\ 1 \end{pmatrix} = -\frac{1}{2} \begin{pmatrix} -8 \\ -2 \end{pmatrix} \quad \text{or equivalently,} \quad \det \begin{pmatrix} 4 & -8 \\ 1 & -2 \end{pmatrix} = 0$$

Thus basic solutions do not exist.

(b) Basic solutions do exist because column vectors for x_1 and x_3 form a basis. One way to check is to verify that the corresponding matrix is nonsingular, that is

$$\det \begin{pmatrix} 4 & -1 \\ 1 & 0 \end{pmatrix} = 1 \neq 0$$

(c) Column vector (4,1) for x_1 is linearly independent because it is nonzero. However, we have already seen in part (b) that a larger linearly independent set of columns is possible. Thus x_1 alone does not determine a basic solution.

(d) Basic solutions do exist because column vectors x_2 and x_3 form a basis. One proof is

$$\det \begin{pmatrix} -8 & -1 \\ -2 & 0 \end{pmatrix} = -2 \neq 0$$

(e) Columns for this set of variables cannot form a basis because no more than two 2-vectors can be linearly independent. Thus basic solutions do not exist.

PRIMER 5: SIMULTANEOUS EQUATIONS, SINGULARITY, AND BASES

A system of m **simultaneous linear equations** in m unknowns may have a unique solution, no solutions, or an infinite number of solutions. For example, the $m = 3$ instances

$$\begin{array}{rcl} 3x_1 + x_2 - 7x_3 = 17 & 2y_1 - y_2 - 5y_3 = 3 & 2z_1 - z_2 - 5z_3 = -3 \\ 4x_1 + 5x_2 = 1 & -4y_1 + 8y_3 = 0 & -4z_1 + 8z_3 = 4 \\ -2x_1 + 11x_3 = -24 & -6y_1 - y_2 + 11y_3 = -2 & -6z_1 - z_2 + 11z_3 = 11 \end{array}$$

have unique solution $\mathbf{x} = (1, 0, -2)$, no solution \mathbf{y} , and infinitely many solutions \mathbf{z} , respectively.

Whether a system falls within the first, unique-solution case depends entirely on the variable coefficient structure of its left-hand side. The x -system above continues to have a unique solution if the right-hand side $(17, 1, 24)$ is replaced by any other 3-vector. Notice that the same is not true of the two nonunique cases; the y and z systems above have identical coefficients on the left-hand side.

A square matrix is **singular** if its determinant = 0 and **nonsingular** otherwise. In these terms, systems of m linear equations in m unknowns have unique solutions if and only if the corresponding matrix of left-hand-side variable coefficients is nonsingular. Here the **determinant** of a square matrix \mathbf{D} is the scalar quantity computed recursively as

$$\det(\mathbf{D}) \triangleq \sum_j (-1)^{(j-1)} d_{1,j} \det(\mathbf{D}_j) \quad \text{with} \quad \det(d_{1,j}) \triangleq d_{1,j}$$

and \mathbf{D}_j the matrix obtained from \mathbf{D} by deleting row 1 and column j . Thus the x -system above has a unique solution, and the others do not, because corresponding left-hand-side coefficient matrices

$$\mathbf{N} \triangleq \begin{pmatrix} 3 & 1 & -7 \\ 1 & 5 & 0 \\ -2 & 0 & 11 \end{pmatrix} \quad \text{and} \quad \mathbf{S} \triangleq \begin{pmatrix} 2 & -1 & -5 \\ -4 & 0 & 8 \\ -6 & -1 & 11 \end{pmatrix}$$

are nonsingular and singular, respectively. That is,

$$\begin{aligned} \det(\mathbf{N}) &= 3 \det \begin{pmatrix} 5 & 0 \\ 0 & 11 \end{pmatrix} - 1 \det \begin{pmatrix} 1 & 0 \\ -2 & 11 \end{pmatrix} - 7 \det \begin{pmatrix} 1 & 5 \\ -2 & 0 \end{pmatrix} \\ &= 3(55 - 0) - 1(11 - 0) - 7(0 + 10) = 84 \neq 0 \end{aligned}$$

and $\det(\mathbf{S}) = 0$.

It is often convenient to draw also on the completely equivalent characterization of when square systems of linear equations have unique solutions, which come from treating the columns of left-hand-side coefficients as

(Continued)

vectors. For example, we could deal with coefficients of the y and z systems above as vectors

$$\mathbf{s}^{(1)} \triangleq \begin{pmatrix} 2 \\ -4 \\ -6 \end{pmatrix}, \quad \mathbf{s}^{(2)} \triangleq \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix}, \quad \text{and} \quad \mathbf{s}^{(3)} \triangleq \begin{pmatrix} -5 \\ 8 \\ 11 \end{pmatrix}$$

A **linear combination** of vectors is simply a weighted sum. Weights may be positive, negative, or zero. For example, weights $\frac{1}{2}$ and -3 applied to vectors $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ above produce the linear combination

$$\frac{1}{2}\mathbf{s}^{(1)} - 3\mathbf{s}^{(2)} = \frac{1}{2}(2, -4, -6) - 3(-1, 0, -1) = (4, -2, 0)$$

A collection of vectors is said to be **linearly independent** if all are nonzero, and none can be expressed as a linear combination of the others. Otherwise, the collection is **linearly dependent**. For example, the nonzero vectors $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ above are linearly independent because no multiple of one can produce the other. Still, the expanded collection $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}\}$ is linearly dependent because

$$-2\mathbf{s}^{(1)} + 1\mathbf{s}^{(2)} = -2(2, -4, -6) + 1(-1, 0, -1) = (-5, 8, 11) = \mathbf{s}^{(3)}$$

A **basis** is a largest or **maximal** collection of linearly independent vectors in the sense that members can be combined to produce any other vector. Such linear combinations are unique. Thus $\mathbf{e}^{(1)} = (1, 0)$ and $\mathbf{e}^{(2)} = (0, 1)$ form a basis of the 2 vectors because every (q_1, q_2) is expressed uniquely:

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} q_1 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} q_2 = \mathbf{e}^{(1)} q_1 + \mathbf{e}^{(2)} q_2$$

Any m linearly independent m -vectors form a basis, and vice versa. This property leads to the connection between systems of equations and bases, because solving a system for a right-hand side is the same as representing the right-hand side as a linear combination. To be precise, an m by m system of simultaneous linear equations has a unique solution if and only if the coefficient columns for the various variables form a basis (i.e., if and only if the coefficient columns are linearly independent). For example, the x system above has a unique solution for every right-hand side (b_1, b_2, b_3) , exactly because its coefficient columns form a basis, so that there always exist multipliers x_1, x_2 , and x_3 satisfying

$$\begin{pmatrix} 3 \\ 1 \\ -2 \end{pmatrix} x_1 + \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} x_2 + \begin{pmatrix} -7 \\ 0 \\ 11 \end{pmatrix} x_3 = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

On the other hand, corresponding y and z systems do not produce unique solutions because columns $\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{s}^{(3)}\}$ are linearly dependent.

Basic Feasible Solutions and Extreme Points

Table 5.1 enumerates all possible choices of 2 nonbasic and 4 basic variables in the 4 equations of the Top Brass standard form. In two cases there is no basic solution because (principle [5.16]) the equality constraint columns corresponding to basic variables are linearly dependent.

All other combinations produce basic solutions. Notice, however, that nothing in the construction of basic solutions (definition [5.15]) guarantees feasibility. Some of the solutions in Table 5.1 violate nonnegativity constraints. These correspond geometrically to points where active inequalities determine a solution falling outside the feasible region. For example, choosing x_3 and x_4 nonbasic yields point $\mathbf{x}^{(9)}$ in Figure 5.4, which has negative components for standard-form slack variables of the two constraints it violates.

Our interest is in basic feasible solutions.

Principle 5.17 | A **basic feasible solution** to linear program in standard form is a basic solution that satisfies all nonnegativity constraints.

Comparison of Table 5.1 and Figure 5.4 will show why. The six basic feasible solutions are exactly the six extreme points $\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(5)}$ of the feasible region.

This is no accident. We have seen how extreme points are feasible solutions determined by collections of active constraints (principle [5.12]). For linear programs in standard form, which have only nonnegativity-form inequalities, this means that extreme points are determined by basic solutions. It follows that the extreme-point solutions are the basic ones that are feasible.

Principle 5.18 | The basic feasible solutions of a linear program in standard form are exactly the extreme-point solutions of its feasible region.

TABLE 5.1 Basic Solutions of the Top Brass Example

Active Constraints	Basic Variables	Basic Solution	Solution Status	Point in Fig. 5.4
$x_1 \geq 0, x_2 \geq 0$	x_3, x_4, x_5, x_6	$\mathbf{x} = (0, 0, 1000, 1500, 1750, 4800)$	Feasible	$\mathbf{x}^{(0)}$
$x_1 \geq 0, x_3 \geq 0$	x_2, x_4, x_5, x_6	None	Dependent	—
$x_1 \geq 0, x_4 \geq 0$	x_2, x_3, x_5, x_6	$\mathbf{x} = (0, 1500, 1000, 0, 250, 1800)$	Feasible	$\mathbf{x}^{(5)}$
$x_1 \geq 0, x_5 \geq 0$	x_2, x_3, x_4, x_6	$\mathbf{x} = (0, 1750, 1000, -250, 0, 1300)$	Infeasible	$\mathbf{x}^{(6)}$
$x_1 \geq 0, x_6 \geq 0$	x_2, x_3, x_4, x_5	$\mathbf{x} = (0, 2400, 1000, -900, -650, 0)$	Infeasible	$\mathbf{x}^{(7)}$
$x_2 \geq 0, x_3 \geq 0$	x_1, x_4, x_5, x_6	$\mathbf{x} = (1000, 0, 0, 1500, 750, 800)$	Feasible	$\mathbf{x}^{(1)}$
$x_2 \geq 0, x_4 \geq 0$	x_1, x_3, x_5, x_6	None	Dependent	—
$x_2 \geq 0, x_5 \geq 0$	x_1, x_3, x_4, x_6	$\mathbf{x} = (1750, 0, -750, -1500, 0, -2200)$	Infeasible	$\mathbf{x}^{(12)}$
$x_2 \geq 0, x_6 \geq 0$	x_1, x_3, x_4, x_5	$\mathbf{x} = (1200, 0, -200, 1500, 550, 0)$	Infeasible	$\mathbf{x}^{(11)}$
$x_3 \geq 0, x_4 \geq 0$	x_1, x_2, x_5, x_6	$\mathbf{x} = (1000, 1500, 0, 0, -750, -2200)$	Infeasible	$\mathbf{x}^{(9)}$
$x_3 \geq 0, x_5 \geq 0$	x_1, x_2, x_4, x_6	$\mathbf{x} = (1000, 750, 0, 750, 0, -700)$	Infeasible	$\mathbf{x}^{(10)}$
$x_3 \geq 0, x_6 \geq 0$	x_1, x_2, x_4, x_5	$\mathbf{x} = (1000, 400, 0, 1100, 350, 0)$	Feasible	$\mathbf{x}^{(2)}$
$x_4 \geq 0, x_5 \geq 0$	x_1, x_2, x_3, x_6	$\mathbf{x} = (250, 1500, 750, 0, 0, 800)$	Feasible	$\mathbf{x}^{(4)}$
$x_4 \geq 0, x_6 \geq 0$	x_1, x_2, x_3, x_5	$\mathbf{x} = (450, 1500, 550, 0, -200, 0)$	Infeasible	$\mathbf{x}^{(8)}$
$x_5 \geq 0, x_6 \geq 0$	x_1, x_2, x_3, x_4	$\mathbf{x} = (650, 1100, 350, 400, 0, 0)$	Feasible	$\mathbf{x}^{(3)}$

EXAMPLE 5.10: IDENTIFYING BASIC FEASIBLE SOLUTIONS

The constraint set of a standard-form linear program is defined by the following constraints:

$$\begin{aligned} -x_1 + x_2 - x_3 &= 0 \\ +x_1 &+ x_4 &= 2 \\ &+ x_2 &+ x_5 &= 3 \\ x_1, \dots, x_5 &\geq 0 \end{aligned}$$

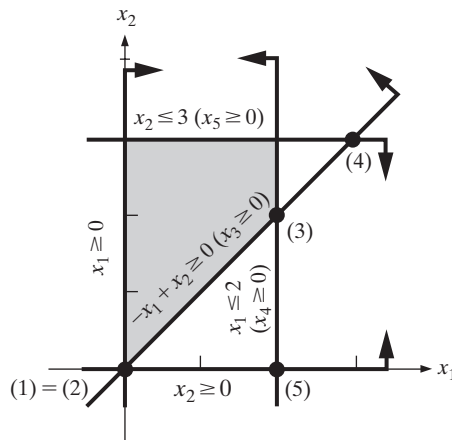
- (a) Assuming that variables $x_3, x_4,$ and x_5 are slack variables added to produce standard form, graph the feasible region in the original variables x_1 and x_2 .
- (b) Compute the basic solutions corresponding to each of the following sets of basic variables, and determine which are basic feasible solutions: $B_1 = \{x_3, x_4, x_5\}, B_2 = \{x_2, x_4, x_5\}, B_3 = \{x_1, x_2, x_5\}, B_4 = \{x_1, x_2, x_4\}, B_5 = \{x_1, x_3, x_5\}$.
- (c) Verify that each basic feasible solution of part (b) corresponds to an extreme point in the graph and that each infeasible basic solution corresponds to a point outside the feasible region determined by the intersection of constraints.

Solution:

(a) Original constraints would have been

$$\begin{aligned} -x_1 + x_2 &\geq 0 \\ x_1 &\leq 2 \\ x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Thus the feasible region is as follows:



(b) Fixing nonbasics $x_1 = x_2 = 0$ for basic set B_1 , the resulting equation system is

$$\begin{array}{rccccr} - (0) & + (0) & - x_3 & & & = 0 \\ + (0) & & & + x_4 & & = 2 \\ & + (0) & & & + x_5 & = 3 \end{array}$$

Solving produces basic solution $\mathbf{x}^{(1)} = (0, 0, 0, 2, 3)$. Results for other basic sets are derived in the same way:

$$\begin{array}{ll} B_2 & \text{implies } \mathbf{x}^{(2)} = (0, 0, 0, 2, 3) \\ B_3 & \text{implies } \mathbf{x}^{(3)} = (2, 2, 0, 0, 1) \\ B_4 & \text{implies } \mathbf{x}^{(4)} = (3, 3, 0, -1, 0) \\ B_5 & \text{implies } \mathbf{x}^{(5)} = (2, 0, -2, 0, 3) \end{array}$$

Notice that different basic sets (here B_1 and B_2) can produce the same basic solution.

Under principle [5.17], only $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$ are basic feasible. The others violate nonnegativity constraints.

(c) Points corresponding to each basic solution of part (b) are indicated on the plot of part (a). Confirming principle [5.18], basic feasible $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$ correspond to extreme points of the feasible region, while basic infeasible $\mathbf{x}^{(4)}$ and $\mathbf{x}^{(5)}$ correspond to infeasible points where active constraints intersect.

5.3 THE SIMPLEX ALGORITHM

The **simplex algorithm** is a variant of improving search elegantly adapted to exploit the special properties of linear programs in standard form. Like all improving searches, it moves from solution to solution, retaining feasibility and improving the objective function until a local optimum (global by [5.1]) is encountered.

What is unusual is that every step of the simplex leaves us at an extreme-point solution. Under principle [5.18] this means that we only need to think about moves among basic feasible solutions.

Standard Display

Definition [5.11] showed how linear programs in standard form can be compactly described by a cost vector \mathbf{c} , a constraint coefficient matrix \mathbf{A} , and a right-hand-side vector \mathbf{b} . The evolution of a simplex search will produce many more vectors for current basic solutions, move directions, and so on.

In this book we display all these vectors simply by adding them as rows of a growing table. To see the idea, return to our Top Brass Trophy model (5.2). Standard-form data produce an initial table, with columns for each of the variables and the right-hand side:

	x_1	x_2	x_3	x_4	x_5	x_6	
max \mathbf{c}	12	9	0	0	0	0	\mathbf{b}
\mathbf{A}	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800

Initial Basic Solution

Any improving search begins by choosing a starting feasible solution, and simplex requires an extreme point.

Principle 5.19 Simplex search begins at an extreme point of the feasible region (i.e., at a basic feasible solution to the model in standard form).

To illustrate for the Top Brass example, we choose (arbitrarily) to begin at extreme point $\mathbf{x}^{(0)} = (0, 0)$ in Figure 5.4. Table 5.1 will show that this solution is obtained as the basic feasible one for basic variables $B = \{x_3, x_4, x_5, x_6\}$ and nonbasic columns $N = \{x_1, x_2\}$. Adding this basis and the corresponding solution vector to our table gives

	x_1	x_2	x_3	x_4	x_5	x_6	
max c	12	9	0	0	0	0	b
	1	0	1	0	0	0	1000
A	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
	N	N	B	B	B	B	
$\mathbf{x}^{(0)}$	0	0	1000	1500	1750	4800	

If we did not already know the solution from Table 5.1, it would have been computed by setting nonbasics $x_1 = x_2 = 0$ and solving as in definition [5.15].

Simplex Directions

The next requirement is move directions. We want simplex to follow edge directions joining current extreme points to adjacent ones. Each such edge direction follows a line determined by all but one of the active constraints that fix our current extreme point because definition [5.12] requires that adjacent extreme points share all but one of their determining active constraints. But the active constraints at a basis feasible solution are nothing more than the nonnegativity constraints on nonbasic variables.

We obtain the simplex directions by making the nonbasic nonnegativities inactive one at a time.

Principle 5.20 Simplex directions are constructed by increasing a single nonbasic variable, leaving other nonbasics unchanged, and computing the (unique) corresponding changes in basic variables necessary to preserve equality constraints.

That is, $\Delta x_j = 1$ on the increasing nonbasic, $\Delta x_j = 0$ on other nonbasics, and basic components are obtained by solving conditions for a feasible direction in the equality constraints.

There is one simplex direction for each nonbasic variable. For example, at current solution $\mathbf{x}^{(0)}$ of the Top Brass example, we have one simplex direction increasing nonbasic x_1 , and another increasing x_2 .

Simplex directions always have +1 on the increasing nonbasic, and other nonbasic components = 0. To complete the directions, we must determine components for the basic variables.

Reaching all the way back to principle [3.25](#) in Section 3.3, we can see what is required. A direction $\Delta \mathbf{x}$ follows the equality constraint

$$\sum_j a_j x_j = b$$

if and only if it satisfies the **net-change-zero condition**

$$\sum_j a_j \Delta x_j = 0$$

Thus for our entire system of equality constraints

$$\mathbf{A} \mathbf{x} = \mathbf{b}$$

every feasible direction must satisfy

$$\mathbf{A} \Delta \mathbf{x} = \mathbf{0} \tag{5.9}$$

At current Top Brass solution $\mathbf{x}^{(0)}$, condition (5.9) produces the following equation system for the simplex direction increasing x_1 when Δx_1 is fixed = 1:

$$\begin{array}{rcccccc} +1(1) & +0(0) & +1 \Delta x_3 & +0 \Delta x_4 & +0 \Delta x_5 & +0 \Delta x_6 = 0 \\ +0(1) & +1(0) & +0 \Delta x_3 & +1 \Delta x_4 & +0 \Delta x_5 & +0 \Delta x_6 = 0 \\ +1(1) & +1(0) & +0 \Delta x_3 & +0 \Delta x_4 & +1 \Delta x_5 & +0 \Delta x_6 = 0 \\ +4(1) & +2(0) & +0 \Delta x_3 & +0 \Delta x_4 & +0 \Delta x_5 & +1 \Delta x_6 = 0 \end{array}$$

Corresponding equations for the simplex direction increasing x_2 are

$$\begin{array}{rcccccc} +1(0) & +0(1) & +1 \Delta x_3 & +0 \Delta x_4 & +0 \Delta x_5 & +0 \Delta x_6 = 0 \\ +0(0) & +1(1) & +0 \Delta x_3 & +1 \Delta x_4 & +0 \Delta x_5 & +0 \Delta x_6 = 0 \\ +1(0) & +1(1) & +0 \Delta x_3 & +0 \Delta x_4 & +1 \Delta x_5 & +0 \Delta x_6 = 0 \\ +4(0) & +2(1) & +0 \Delta x_3 & +0 \Delta x_4 & +0 \Delta x_5 & +1 \Delta x_6 = 0 \end{array}$$

Will these systems have a solution? Absolutely. A basis is a collection of column vectors that can represent every other vector of the same dimension, and we have been careful to leave undetermined only the components on basic variables. Each of the systems (5.9) amounts to finding multipliers Δx_j that weight columns of basic variables to produce the negative of the column for nonbasic k . Such multipliers have to exist. In fact, we know that they are unique (Primer 5).

Completing the two simplex directions for our current Top Brass solution yields the following updated table:

	x_1	x_2	x_3	x_4	x_5	x_6	
max c	12	9	0	0	0	0	b
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
	N	N	B	B	B	B	
$\mathbf{x}^{(0)}$	0	0	1000	1500	1750	4800	
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	-1	-4	
$\Delta \mathbf{x}$ for x_2	0	1	0	-1	-1	-2	

EXAMPLE 5.11: CONSTRUCTING SIMPLEX DIRECTIONS

A minimizing, standard-form linear program has the following coefficient data:

	x_1	x_2	x_3	x_4	
min c	2	0	-3	18	b
A	1	-1	2	1	4
	1	1	0	3	2

Assume that x_1 and x_3 are basic, solve for the current basic feasible solution, and compute all corresponding simplex directions.

Solution: Following definition [5.15], the current basic solution is obtained by setting nonbasics $x_2 = x_4 = 0$ and solving equality constraints for basic variable values. Here,

$$\begin{aligned} +1x_1 - 1(0) + 2x_3 + 1(0) &= 4 \\ +1x_1 + 1(0) + 0x_3 + 3(0) &= 2 \end{aligned}$$

gives $\mathbf{x} = (2, 0, 1, 0)$.

There will be two simplex directions, one increasing nonbasic x_2 and another increasing nonbasic x_4 . Following principle [5.20], the direction increasing x_2 has $\Delta x_2 = 1$ and its components on all other nonbasics (here only x_4) = 0. We must solve for basic components Δx_1 and Δx_3 satisfying net-change-zero condition (5.10). The corresponding linear system

$$\begin{aligned} +1 \Delta x_1 - 1(1) + 2 \Delta x_3 + 1(0) &= 0 \\ +1 \Delta x_1 + 1(1) + 0 \Delta x_3 + 3(0) &= 0 \end{aligned}$$

has unique solution $\Delta x_1 = -1, \Delta x_3 = 1$. Thus the simplex direction increasing x_2 is $\Delta \mathbf{x} = (-1, 1, 1, 0)$.

For $\Delta \mathbf{x}^{(4)}$ the corresponding linear system is

$$\begin{aligned} +1 \Delta x_1 - 1(0) + 2 \Delta x_3 + 1(1) &= 0 \\ +1 \Delta x_1 + 1(0) + 0 \Delta x_3 + 3(1) &= 0 \end{aligned}$$

Solving for the two unknown components yields simplex direction $\Delta \mathbf{x} = (-3, 0, 1, 1)$.

Improving Simplex Directions and Reduced Costs

Having constructed the collection of simplex directions that can be pursued from our current basic solution without losing feasibility, our next task is to see if any of them improve the objective function

$$f(\mathbf{x}) \triangleq \mathbf{c} \cdot \mathbf{x} \triangleq \sum_{j=1}^n c_j x_j$$

We know from Section 3.3 that improvement can be checked by referring to the gradient $\nabla f(\mathbf{x})$ at our current solution.

For linear objective function $f(\mathbf{x})$, the gradient is just the vector of objective function coefficients. That is,

$$\nabla f(\mathbf{x}) = \mathbf{c} \triangleq (c_1, c_2, \dots, c_n) \quad \text{for all } \mathbf{x}$$

Thus gradient conditions [3.21](#) and [3.22](#) require checking quantities that we term reduced costs.

Principle 5.21 | The **reduced cost** \bar{c}_j associated with nonbasic variable x_j is

$$\bar{c}_j = \mathbf{c} \cdot \Delta \mathbf{x}$$

where $\Delta \mathbf{x}$ is the simplex direction increasing x_j .

Principle 5.22 | The simplex direction increasing nonbasic x_j is improving for a maximize linear program if the corresponding reduced cost $\bar{c}_j > 0$, and for a minimize linear program if $\bar{c}_j < 0$.

These simple tests tell us immediately that both simplex directions at solution $\mathbf{x}^{(0)}$ of the Top Brass example improve the maximize objective. For example, the direction increasing x_1 gives

$$\bar{c}_1 = (12, 9, 0, 0, 0, 0) \cdot (1, 0, -1, 0, 0, -4) = 12 > 0$$

and for the one increasing x_2 ,

$$\bar{c}_2 = (12, 9, 0, 0, 0, 0) \cdot (0, 1, 0, -1, -1, -2) = 9 > 0$$

More typically (see Example 5.12), some simplex directions at a current basic solution will improve, and others will not.

EXAMPLE 5.12: CHECKING IMPROVEMENT OF SIMPLEX DIRECTIONS

Determine which of the simplex directions computed in Example 5.11 are improving for the specified minimizing objective function.

Solution: We apply computations [5.21] and [5.22]. For direction $\Delta \mathbf{x} = (-1, 1, 1, 0)$ increasing x_2 ,

$$\bar{c}_2 = (2, 0, -3, 18) \cdot (-1, 1, 1, 0) = -5 < 0$$

Thus the direction does improve. On the other hand, for $\Delta \mathbf{x} = (-3, 0, 1, 1)$, increasing x_4 does not improve because

$$\bar{c}_4 = (2, 0, -3, 18) \cdot (-3, 0, 1, 1) = 9 \not< 0$$

Step Size and the Minimum Ratio Rule

Simplex can adopt for the next move of improving search any simplex direction that improves the objective function. The next issue is “How far?” that is, what step size λ should be applied to chosen direction $\Delta \mathbf{x}$?

Following principle [3.15] in Section 3.2, we would like to take the biggest step that preserves feasibility and improves the objective function value. With the constant gradient for linear programs, an improving simplex direction remains improving forever. Also, we constructed simplex directions to maintain all equality constraints $\mathbf{Ax} = \mathbf{b}$.

If there is any limit on step size λ , it must come from eventually violating a nonnegativity constraint. That is, some component of the revised solution must become negative. With all components feasible, and thus nonnegative at our current solution, infeasibility can occur only if some component of the chosen simplex direction is negative. The first solution component forced to zero fixes λ .

Principle 5.23 If any component is negative in improving simplex direction $\Delta \mathbf{x}$ at current basic solution $\mathbf{x}^{(t)}$, simplex search uses the maximum feasible step of **minimum ratio** computation to determine step size λ

$$\lambda = \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j} : \Delta x_j < 0 \right\}$$

Principle 5.24 If no component is negative in improving simplex direction $\Delta \mathbf{x}$ at current basic solution $\mathbf{x}^{(t)}$, the solution can be improved forever in direction $\Delta \mathbf{x}$. That is, the linear program is unbounded.

To illustrate, we arbitrarily set $\Delta \mathbf{x} = (1, 0, -1, 0, -1, -4)$, increasing x_1 at $\mathbf{x}^{(0)}$ of our Top Brass example. The chosen direction $\Delta \mathbf{x}^{(1)}$ does have negative components, so there is no indication of unboundedness.

To decide the maximum step, we add a row to our table that computes the step size at which each component would drop to $= 0$.

	x_1	x_2	x_3	x_4	x_5	x_6
	N	N	B	B	B	B
$\mathbf{x}^{(0)}$	0	0	1000	1500	1750	4800
$\Delta \mathbf{x}$	1	0	-1	0	-1	-4
	-	-	$\frac{1000}{-(-1)}$	-	$\frac{1750}{-(-1)}$	$\frac{4800}{-(-4)}$

The least of these ratios establishes λ in rule [5.24](#):

$$\lambda = \min \left\{ \frac{1000}{1}, \frac{1750}{1}, \frac{4800}{4} \right\} = 1000$$

Thus our new solution is

$$\begin{aligned} \mathbf{x}^{(1)} &\leftarrow \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x} \\ &= (0, 0, 1000, 1500, 1750, 4800) + 1000(1, 0, -1, 0, -1, -4) \\ &= (1000, 0, 0, 1500, 750, 800) \end{aligned}$$

EXAMPLE 5.13: DETERMINING THE MAXIMUM SIMPLEX STEP

Let the current simplex solution to a linear program in standard form be $\mathbf{x}^{(17)} = (13, 0, 10, 2, 0, 0)$, with x_2 and x_5 nonbasic. Determine the maximum step and new solution (if any), assuming that each of the following is the improving simplex direction associated with increasing x_2 .

- (a) $\Delta \mathbf{x} = (12, 1, -5, -1, 0, 8)$
- (b) $\Delta \mathbf{x} = (0, 1, 6, 3, 0, \frac{7}{2})$
- (c) $\Delta \mathbf{x} = (-1, 1, -8, 0, 0, -5)$

Solution:

(a) This improving simplex direction has negative components, so we apply rule [5.24](#).

$$\lambda = \min \left\{ \frac{10}{-(-5)}, \frac{2}{-(-1)} \right\} = 2$$

Notice that values for x_1 were not included in this computation; with a positive directional component $\Delta x_1 = 12$, x_1 is increasing. The new simplex solution will be

$$\begin{aligned} \mathbf{x}^{(18)} &\leftarrow \mathbf{x}^{(17)} + \lambda \Delta \mathbf{x} \\ &= (13, 0, 10, 2, 0, 0) + 2(12, 1, -5, -1, 0, 8) \\ &= (37, 2, 0, 0, 0, 16) \end{aligned}$$

(b) This improving simplex direction has no negative components, so progress is unlimited. Under principle [5.24], the model is unbounded.

(c) This improving simplex direction does have negative components. Applying rule [5.23] yields

$$\lambda = \min \left\{ \frac{13}{-(-1)}, \frac{10}{-(-8)}, \frac{0}{-(-5)} \right\} = 0$$

and new solution

$$\mathbf{x}^{(18)} \leftarrow \mathbf{x}^{(17)} + 0 \Delta \mathbf{x} = \mathbf{x}^{(17)}$$

The zero step λ results from basic variable x_6 happening to take on the zero value more typical of nonbasics—a common occurrence with large-scale linear programs. In Section 5.6 we discuss this **degenerate** case further.

Updating the Basis

Step size rule [5.24] tells us to continue from our present extreme-point solution, along the edge direction formed by increasing a single nonbasic, until an adjacent extreme point is formed by a newly active nonnegativity constraint (definitions [5.14] and [5.15]). To continue the algorithm we need to find a new basis corresponding to this new extreme-point solution. Active nonnegativity constraints tell us how.

Principle 5.25 After each move of simplex search, the nonbasic variable generating the chosen simplex direction enters the basis, and any one of the (possibly several) basic variables fixing step size λ leaves the basis.

That is, we move to the new basis implied by the nonnegativity constraint on the increasing nonbasic variable becoming inactive, while a nonnegativity constraint on a blocking basic becomes active.

For our Top Brass example, the move from $\mathbf{x}^{(0)}$ increased nonbasic x_1 , and x_3 is the first component to drop to $= 0$. Thus x_1 enters and x_3 leaves, resulting in new basic set $\{x_1, x_4, x_5, x_6\}$.

Figure 5.5 interprets graphically. New solution $\mathbf{x}^{(1)}$ is indeed the adjacent extreme point obtained when we move along the edge direction for x_1 until the nonnegativity constraint for x_3 becomes active. The new basis makes x_1 basic and x_3 nonbasic.

EXAMPLE 5.14: UPDATING THE BASIS

Return to Example 5.13 and determine for bounded cases (a) and (c) what variables should enter and leave the basis.

Solution: We apply rule [5.25]. Increasing nonbasic x_2 enters in both cases. For case (a) there is a choice of leaving variables because both x_3 and x_4 establish λ ; either one could be selected to leave the basis. Variable x_6 alone fixes the value of λ in case (c); it must be chosen as the leaving basic.

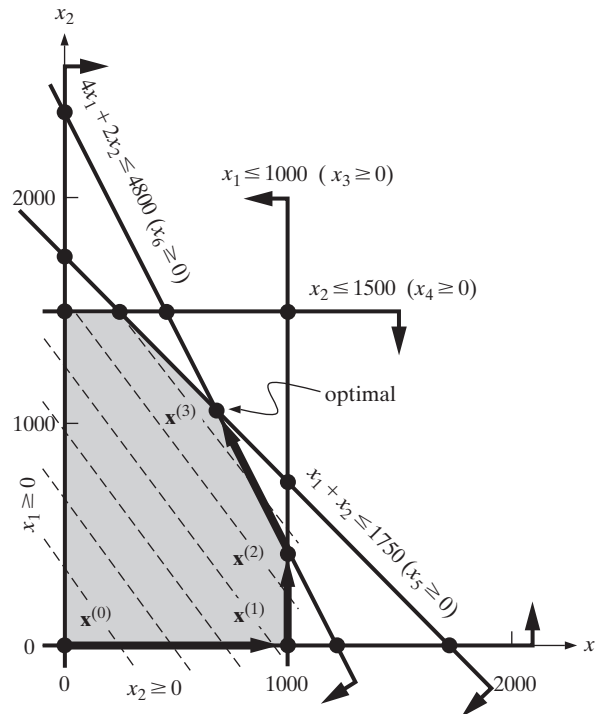


FIGURE 5.5 Simplex Search of the Top Brass Example

Rudimentary Simplex Algorithm

We have now developed all the main ideas of simplex search. Simplex implements extreme-point search by moving from basic feasible solution to basic feasible solution until either the problem is shown to be unbounded or the current solution proves optimal. At every iteration a move is contemplated in each simplex direction. If any of the simplex directions is improving, one is selected and pursued as far as feasibility permits. If no simplex direction improves, we stop and report an optimum. Algorithm 5A gives a formal statement.

ALGORITHM 5A: RUDIMENTARY SIMPLEX SEARCH FOR LINEAR PROGRAMS

Step 0: Initialization. Choose any starting feasible basis, construct the corresponding basic solution $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Simplex Directions. Construct the simplex direction $\Delta \mathbf{x}$ associated with increasing each nonbasic variable x_j , and compute the corresponding reduced cost $\bar{c}_j = \mathbf{c} \cdot \Delta \mathbf{x}$.

Step 2: Optimality. If no simplex direction is improving (no $\bar{c}_j > 0$ for a maximize problem, or no $\bar{c}_j < 0$ for a minimize), then stop; current solution

$\mathbf{x}^{(t)}$ is globally optimal. Otherwise, choose any improving simplex direction as $\Delta\mathbf{x}^{(t+1)}$ and denote the associated entering basic variable x_p .

Step 3: Step Size. If there is no limit on feasible moves in simplex direction $\Delta\mathbf{x}^{(t+1)}$ (all components are nonnegative), then stop; the given model is unbounded. Otherwise, choose leaving variable x_r so that

$$\frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}} = \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\} \quad \text{and set} \quad \lambda \leftarrow \frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}}$$

Step 4: New Point and Basis. Compute the new solution

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta\mathbf{x}^{(t+1)}$$

and replace x_r in the basis by x_p . Then advance $t \leftarrow t + 1$, and return to Step 1.

Rudimentary Simplex Solution of Top Brass Example

Table 5.2 details the full Algorithm 5A search of our Top Brass Trophy example in LP standard form. Figure 5.5 tracks progress graphically. As in our earlier discussion, the search begins at basic feasible solution $\mathbf{x}^{(0)} = (0, 0, 1000, 1500, 1750, 4800)$. We have already detailed the first iteration $t = 0$. Simplex directions for nonbasics x_1 and x_2 both improve. Choosing $p = 1$, the maximum feasible step is determined by the component with subscript $r = 3$ at $\lambda = 1000$.

After the move, we have new solution $\mathbf{x}^{(1)} = (1000, 0, 0, 1500, 750, 800)$ with objective function value \$12,000. Increasing variable x_1 has replaced blockings x_3 in the basis. The process now repeats with $t = 1$. Simplex directions are available for x_2 and x_3 , but only the first improves because $\bar{c}_2 = 9$ and $\bar{c}_3 = -12$. Thus $p = 2$, and we take a maximum step in the direction for x_2 . Step size $\lambda = 400$ is fixed as basics x_6 decreases.

Taking this new step brings us to $\mathbf{x}^{(2)} = (1000, 400, 0, 1100, 350, 0)$ with objective value \$15,600. Figure 5.5 confirms that it too is an extreme-point solution.

The basis is now $\{x_1, x_2, x_4, x_5\}$. As usual, there are two simplex directions available, one for each nonbasic. However, only the one for x_3 improves. At $\lambda = 350$, the nonnegativity constraint on x_5 becomes active, producing new solution $\mathbf{x}^{(3)} = (650, 1100, 350, 400, 0, 0)$.

With $t = 3$ we repeat the sequence again. This time, however, neither simplex direction is improving. The search terminates with (global) optimal solution $\mathbf{x}^* = (650, 1100, 350, 400, 0, 0)$ having profit \$17,700.

Stopping and Global Optimality

If Algorithm 5A stops with a simplex direction along which we can improve forever without losing feasibility, the given model is clearly unbounded. But what if it stops with an optimum?

TABLE 5.2 Rudimentary Simplex Search of Top Brass Trophy Example

	X_1	X_2	X_3	X_4	X_5	X_6	
max c	12	9	0	0	0	0	b
A	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
$t = 0$	N	N	B	B	B	B	
$\mathbf{x}^{(0)}$	0	0	1000	1500	1750	4800	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 0$
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	-1	-4	$\bar{c}_1 = \boxed{12}$
$\Delta \mathbf{x}$ for x_2	0	1	0	-1	-1	-2	$\bar{c}_2 = 9$
	—	—	$\frac{1000}{-(-1)}$	—	$\frac{1750}{-(-1)}$	$\frac{4800}{-(-4)}$	$\lambda = 1000$
$t = 1$	B	N	N	B	B	B	
$\mathbf{x}^{(1)}$	1000	0	0	1500	750	800	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 12,000$
$\Delta \mathbf{x}$ for x_2	0	1	0	-1	-1	-2	$\bar{c}_2 = \boxed{9}$
$\Delta \mathbf{x}$ for x_3	-1	0	1	0	0	4	$\bar{c}_3 = -12$
	—	—	—	$\frac{1500}{-(-1)}$	$\frac{750}{-(-1)}$	$\frac{800}{-(-2)}$	$\lambda = 400$
$t = 2$	B	B	N	B	B	N	
$\mathbf{x}^{(2)}$	1000	400	0	1100	350	0	$\mathbf{c} \cdot \mathbf{x}^{(2)} = 15,600$
$\Delta \mathbf{x}$ for x_3	-1	2	1	-2	-1	0	$\bar{c}_3 = \boxed{6}$
$\Delta \mathbf{x}$ for x_6	0	-0.5	0	0.5	0.5	1	$\bar{c}_6 = -4.5$
	$\frac{1000}{-(-1)}$	—	—	$\frac{1100}{-(-2)}$	$\frac{350}{-(-1)}$	—	$\lambda = 350$
$t = 3$	B	B	B	B	N	N	
$\mathbf{x}^{(3)}$	650	1100	350	400	0	0	$\mathbf{c} \cdot \mathbf{x}^{(3)} = 17,700$
$\Delta \mathbf{x}$ for x_5	1	-2	-1	2	1	0	$\bar{c}_5 = -6$
$\Delta \mathbf{x}$ for x_6	-0.5	0.5	0.5	-0.5	0	1	$\bar{c}_6 = -1.5$
							“optimal”

Simplex search considers only the simplex or edge directions as it looks for an improving feasible move. As long as one of the simplex directions improves, this approach is completely consistent with our previous development of improving search. Still, we stop when no simplex direction is improving and feasible.

Could some non-simplex feasible direction still improve? No.

Principle 5.26 When simplex Algorithm 5A is applied to a linear program in standard form, it either stops with a correct indication of unboundedness or produces a globally optimal solution.

We know Algorithm 5A stops correctly with either a valid conclusion of unboundedness or our reaching a basic feasible solution that admits no improving feasible simplex direction. Could there be an improving feasible direction that is not

simplex? To investigate, pick a feasible direction \mathbf{d} at a current basic solution. Notice that it must increase two or more nonbasics because the basic part of the solution is fixed once nonbasic changes are chosen, and simplex directions already consider the case of single nonbasic increases.

Let $K \triangleq \{\text{nonbasic } k \text{ with } d_k > 0\}$ and $B \triangleq$ the set of currently basic variables. Now, there is a corresponding simplex direction $\Delta \mathbf{x}^{(k)}$ for every k , and their feasibility requires

$$\mathbf{a}^{(k)} = -\sum_{j \in B} \mathbf{a}^{(j)} \Delta x_j^{(k)} \quad \text{for all } k \in K$$

Then summing,

$$\sum_{k \in K} \mathbf{a}^{(k)} d_k = -\sum_{k \in K} [\sum_{j \in B} \mathbf{a}^{(j)} \Delta x_j^{(k)}] d_k$$

That is, the combined change in nonbasic variable values implied by \mathbf{d} can be feasible only if the combined change on basic variables is exactly the weighted sum of associated changes for corresponding simplex directions $\Delta \mathbf{x}^{(k)}$. In short $\mathbf{d} = \sum_{k \in K} \Delta \mathbf{x}^{(k)} d_k$ and the corresponding objective function improvement criterion is $\mathbf{c} \cdot \mathbf{d} = \sum_{k \in K} \mathbf{c} \cdot \Delta \mathbf{x}^{(k)} d_k$. If it tests improving (> 0 for max, < 0 for min), then one of its components of the sum must do so too. It follows that the composite direction can be improving and feasible only if there is an improving simplex direction.

Extreme-Point or Extreme-Direction

It will sometimes be useful to be clear about the polyhedral character of the two simplex outcomes described in [5.26]. In fact, they align exactly with the two fundamental elements of polyhedral sets – **extreme points** and **extreme directions** (see Primer 3).

Principle 5.27 Simplex Algorithm 5A terminates with either an optimal extreme-point solution of the model's feasible set or an extreme direction of the set along which the objective improves forever.

If the given LP has an optimal solution, we already know it will terminate with an optimal basic-feasible one in standard form. That is, it will terminate with an optimal extreme-point of the model's feasible set [5.12]. What about when we conclude the model is unbounded? That occurs when the search encounters a simplex direction $\Delta \mathbf{x}^{(t+1)}$ along which the objective can improve forever. Because it is an improving simplex direction at the current extreme-point solution, it follows an edge (definition [5.14] of the polyhedral feasible set). Although it does not terminate at an adjacent extreme point [5.13], its nature as an edge assures it is extreme among direction of that polyhedron.

5.4 DICTIONARY AND TABLEAU REPRESENTATIONS OF SIMPLEX

Our development of the simplex algorithm in Section 5.3 follows the improving search paradigm, which constitutes one of the main themes of this book. It also reflects the way professionals and researchers in linear programming think about simplex.

Still, readers who have encountered simplex in other introductory texts may find our form a bit difficult to recognize. In this section we make a brief connection to more traditional formats. Those not confused by our departures from tradition can skip the section without loss.

Simplex Dictionaries

Traditional introductory developments of the simplex algorithm view the process that we have conceived as a search in terms of manipulating the objective function and main standard-form equations

$$\sum_{j=1}^n a_{i,j}x_j = b_i \quad \text{for all } i = 1, \dots, m$$

At each iteration representations called simplex dictionaries express the basic variables and the objective function value in terms of the nonbasics.

Principle 5.28 | **Simplex dictionaries** express objective function value z and basic variables $x_k, k \in B$, in terms of nonbasic variables $x_j, j \in N$, as

$$\begin{aligned} z &= \bar{z} + \sum_{j \in N} \bar{c}_j x_j \\ x_k &= \bar{b}_k - \sum_{j \in N} \bar{a}_{k,j} x_j \quad \text{for all } k \in B \end{aligned}$$

Dictionary form is achieved by **Gaussian elimination**—solving for one basic variable at a time and substituting for it in other constraints and the objective. To see the idea, consider the basis corresponding to $t = 2$ in Table 5.2. There basic and nonbasic index sets are

$$\begin{aligned} B &= \{1, 2, 4, 5\} \\ N &= \{3, 6\} \end{aligned}$$

We begin our derivation of the corresponding dictionary with the original objective and constraints:

$$\begin{aligned} z &= 12x_1 + 9x_2 \\ x_1 + x_3 &= 1000 \\ x_2 + x_4 &= 1500 \\ x_1 + x_2 + x_5 &= 1750 \\ 4x_1 + 2x_2 + x_6 &= 4800 \end{aligned}$$

The first constraint expresses basic variable x_1 as

$$x_1 = 1000 - (1x_3)$$

Substituting for x_1 leaves

$$\begin{aligned}x_1 &= 1000 - (1x_3) \\x_2 + x_4 &= 1500 \\(1000 - x_3) + x_2 + x_5 &= 1750 \\4(1000 - x_3) + 2x_2 + x_6 &= 4800\end{aligned}$$

We now continue with basic x_2 . Solving the second equation and substituting gives

$$\begin{aligned}x_1 &= 1000 - (1x_3) \\x_2 &= 1500 - (1x_4) \\(1000 - x_3) + (1500 - x_4) + x_5 &= 1750 \\+4(1000 - x_3) + 2(1500 - x_4) + x_6 &= 4800\end{aligned}$$

The third equation can now be solved for basic variable x_4 . After substitution and solving the last equation for x_5 , we complete the constraints as

$$\begin{aligned}x_1 &= 1000 - (+1x_3 + 0 x_6) \\x_2 &= 400 - (-2x_3 + 0.5x_6) \\x_4 &= 1100 - (+2x_3 - 0.5x_6) \\x_5 &= 350 - (+1x_3 - 0.5x_6)\end{aligned}\tag{5.10}$$

Finally, we substitute these expressions in the objective function to obtain

$$\begin{aligned}z &= 12(1000 - 1x_3) + 9(400 - 2x_3 - 0.5x_6) \\&= 15,600 + 6x_3 - 4.5x_6\end{aligned}\tag{5.11}$$

The dictionary (5.11)–(5.12) is now complete. In the notation of definition [5.28](#), for example, $\bar{z} = 15,600$, $\bar{c}_3 = 6$, $\bar{b}_2 = 400$, and $\bar{a}_{4,6} = -0.5$.

EXAMPLE 5.15: CONSTRUCTING SIMPLEX DICTIONARIES

The computed simplex directions and reduced costs for the linear program of Examples 5.11 and 5.12

$$\begin{aligned}\min \quad & z = 2x_1 - 3x_3 + 18x_4 \\ \text{s.t.} \quad & x_1 - x_2 + 2x_3 + x_4 = 4 \\ & x_1 + x_2 + 3x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0\end{aligned}$$

had x_1 and x_3 basic. Apply Gaussian elimination to derive the corresponding simplex dictionary.

Solution: We want to solve for basic variables x_1 and x_3 in terms of nonbasics. From the first constraint.

$$x_1 = 4 - (-1x_2 + 2x_3 + 1x_4)$$

Substituting in the second constraint yields

$$(4 + x_2 - 2x_3 - 1x_4) + x_2 + 3x_4 = 2$$

Then, solving for x_3 and substituting produces the dictionary:

$$z = 1 + -5x_2 + 9x_4$$

$$x_1 = 2 - (+1x_2 + 3x_4)$$

$$x_3 = 1 - (-1x_2 - 1x_4)$$

Simplex Tableaux

Simplex tableaux are detached-coefficient displays of exactly the same information as simplex dictionaries.

Definition 5.29 The **simplex tableau** associated with basic set $\{x_k : k \in B\}$ displays the coefficients \bar{z} , \bar{c}_j , \bar{b}_i , and $\bar{a}_{i,j}$ of the corresponding simplex dictionary in detached-coefficient form with all variables translated to the left-hand side.

For example, the dictionary (5.11) corresponds to the simplex tableau

x_1	x_2	x_3	x_4	x_5	x_6	
0	0	-6	0	0	+4.5	15,600
1	0	1	0	0	0	1,000
0	1	-2	0	0	0.5	400
0	0	2	1	0	-0.5	1,100
0	0	1	0	1	-0.5	350

The only change is that nonbasic variables have been translated to the left-hand side, and coefficients have been extracted in a matrix.

EXAMPLE 5.16: CONSTRUCTING SIMPLEX TABLEAUX

Construct the simplex tableau corresponding to the dictionary of Example 5.15.

Solution: With all variables translated to the left-hand side, the coefficient tableau is

x_1	x_2	x_3	x_4	
0	5	0	-9	1
1	1	0	3	2
0	-1	1	-1	1

Simplex Algorithm with Dictionaries or Tableaux

In dictionary/tableau form, simplex still moves from basic feasible solution to basic feasible solution. Each simplex iteration begins by checking the sign of objective function coefficients \bar{c}_j on nonbasic variables. If none is negative for a minimize problem (positive for a maximize), the current basic solution is optimal. Otherwise, the solution can be improved by increasing a nonbasic variable from its basic-solution value = 0. We pick one such variable as the entering nonbasic. Its coefficient column in the dictionary or tableau tells us how basic variables will change as the nonbasic increases. If it can increase forever without losing feasibility, the problem is unbounded. Otherwise, the entering variable's increase eventually drives some basic variable to its lower limit of zero. This establishes a leaving basic variable. We update the dictionary or tableau to the new basis and repeat the process.

Correspondence to the Improving Search Paradigm

This brief synopsis of the simplex with dictionaries or tableaux should sound very familiar. It is, in fact, exactly Algorithm 5A. The only difference comes in how the arithmetic is accomplished.

Think first about the current basic solution. When nonbasic variables are fixed = 0, dictionary format makes the corresponding values of basic variables obvious.

Principle 5.30 Right-hand-side constants \bar{b}_k of simplex dictionary/tableau format show current values for corresponding basic variables x_k . Similarly, \bar{z} is the objective function value of the current basic solution.

For example, the dictionary (5.11) shows clearly that with nonbasics $x_3 = x_6 = 0$, the current basic solution is

$$(\bar{b}_1, \bar{b}_2, 0, \bar{b}_4, \bar{b}_5, 0) = (1000, 400, 0, 1100, 350, 0)$$

This is exactly the solution reported in Table 5.2 for $t = 2$. Its objective function value is $\bar{z} = 15,600$.

Next consider the simplex directions. Definition [5.21](#) specifies components for nonbasic variables and leaves those for basic variables to be computed. Specifically, the changes in basics associated with increasing any nonbasic are unique solutions to equations $\mathbf{A}\Delta\mathbf{x} = \mathbf{0}$.

The simplex in dictionary/tableau form uses columns of the tableau to predict the same basic variable changes. Since the changes are unique, there must be a close connection to simplex directions.

Principle 5.31 Simplex dictionary/tableau coefficients $\bar{a}_{k,j}$ for nonbasic variables x_j are exactly the negatives $-\Delta x_k$ of corresponding components in the simplex directions increasing x_j .

The dictionary (5.11) illustrates again. There

$$\begin{aligned}\bar{a}_{1,3} &= +1 & \bar{a}_{1,6} &= 0 \\ \bar{a}_{2,3} &= -2 & \bar{a}_{2,6} &= +0.5 \\ \bar{a}_{4,3} &= +2 & \bar{a}_{4,6} &= -0.5 \\ \bar{a}_{5,3} &= +1 & \bar{a}_{5,6} &= -0.5\end{aligned}$$

Table 5.2 for $t = 2$ shows that the simplex directions are

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} -1 \\ +2 \\ +1 \\ -2 \\ -1 \\ 0 \end{pmatrix} \text{ for } x_3 \quad \text{and} \quad \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ -0.5 \\ 0 \\ 0.5 \\ 0.5 \\ 1 \end{pmatrix} \text{ for } x_6$$

Notice that tableau coefficients are exactly the negatives of corresponding simplex direction components for basic variables $x_1, x_2, x_4,$ and x_5 .

Finally, focus on the reduced costs which tell us whether a nonbasic variable can increase productively. We have employed the same \bar{c}_j notation in both reduced cost principle [5.22] and format [5.28] because they refer to the same quantities.

Principle 5.32 Simplex dictionary/tableau objective function coefficients \bar{c}_j are exactly the reduced costs of definition [5.22].

In both representations they show how objective function value z will change per unit increase in nonbasic variable x_j if basics are adjusted to preserve feasibility in main equality constraints. For example, the reduced cost of x_3 in both dictionary (5.11) and Table 5.2 (at $t = 2$) is $\bar{c}_3 = 6$.

Comparison of Formats

Observations [5.30] to [5.32] document how solving for current basic variables in dictionary format [5.28] is just an alternative way to compute the basic solution, simplex directions, and reduced costs required in rudimentary simplex Algorithm 5A. Nothing is fundamentally different.

One can argue about which computation is easier for hand calculation. Still, it is important to realize that neither is ever employed in serious computer implementations of the simplex algorithm. Sections 5.7 and 5.8 will detail the much more efficient **revised simplex** version at the heart of all commercial simplex codes.

5.5 TWO PHASE SIMPLEX

As usual, we have developed the simplex version of improving search assuming that we know a starting basic feasible solution. In most real problems we will have to search for one.

In Section 3.6 we introduced the generic two-phase approach for all math programs (Algorithm 3B). A solution is chosen that satisfies at least part of the constraints; artificial variables are added to synthetically satisfy all other constraints; Phase I minimizes the sum of the artificials; Phase II proceeds from the result of Phase I to optimize the real objective function.

This section specializes those ideas to LPs and the simplex method.

APPLICATION 5.2: CLEVER CLYDE

We will illustrate two-phase simplex computation with the contrived but instructive case of Clever Clyde. Clyde is an entrepreneur of dubious integrity who is seeking to open a sports collectibles business. He has only \$5000 of his own, and the business will require at least \$100,000. Still, Clyde also has a real knack for convincing others to invest in his projects. One investor has already agreed to pay 50% of the initial cost of the firm in return for an equal partnership, and Clyde has an appointment tomorrow with another prospect. We want to consider several cases for the fraction of support, call it α , that Clyde may get from selling a second “equal” partnership. His goal is to maximize the size of the business.

To formulate Clyde’s dilemma as a linear program, we will use the decision variables

$x_1 \triangleq$ amount invested by present investor 1 (thousands of dollars)

$x_2 \triangleq$ amount invested by new investor 2 (thousands of dollars)

$x_3 \triangleq$ amount Clyde invests of his own money (thousands of dollars)

Then, clearly the objective function is to

$$\max x_1 + x_2 + x_3 \quad (5.12)$$

Some constraints are also easy:

$$x_1 + x_2 + x_3 \geq 100 \quad (5.13)$$

$$x_1 \geq 0, x_2 \geq 0, 5 \geq x_3 \geq 0$$

Total investment must be at least \$100,000, all amounts are nonnegative, and Clyde’s funds are limited to \$5000.

Investment constraints are a bit more complex. If investor 1 is to provide half the capital, and investor 2 a fraction α , we want

$$\frac{x_1}{x_1 + x_2 + x_3} = 0.5 \quad \text{and} \quad \frac{x_2}{x_1 + x_2 + x_3} = \alpha \quad (5.14)$$

These ratio constraints do not even look linear, but knowing $x_1 + x_2 + x_3$ will be positive in feasible solutions, we can clear denominators and collect terms (principle [4.5](#)) to obtain

$$-0.5x_1 + 0.5x_2 + 0.5x_3 = 0$$

$$\alpha x_1 + (\alpha - 1)x_2 + \alpha x_3 = 0 \quad (5.15)$$

Adding slacks x_4 and x_5 to place the LP in standard form, Clyde's dilemma is represented by the model

$$\begin{array}{rllllll}
 \max & +1x_1 & + 1x_2 & + 1x_3 & & & \\
 \text{s.t.} & +1x_1 & + 1x_2 & + 1x_3 & - 1x_4 & & = 100 \\
 & & & + 1x_3 & & + 1x_5 & = 5 \\
 & -0.5x_1 & + 0.5x_2 & + 0.5x_3 & & & = 0 \\
 & +\alpha x_1 & + (\alpha - 1)x_2 & + \alpha x_3 & & & = 0 \\
 & & & & & & \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0 & & & & &
 \end{array} \quad (5.16)$$

Starting Basis in the Two Phase Simplex

Two-phase computation begins by constructing a starting feasible solution to model (5.16) using artificial variables. There is one new issue in adapting the two-phase approach to simplex. Simplex search employs basic solutions. Our starting solution in Phase I must not just be feasible in the artificial model, but basic feasible.

One set of columns that is sure to qualify for a basis by being linearly independent is a collection where each column has only one nonzero component and no two of the columns are nonzero in the same component. That is, constraint columns with a single nonzero entry provide natural choices for a starting basis. Where none exists after the model has been placed in standard form, the column of an artificial variable will do the job. Either way, the single nonzero must have the same sign as the corresponding right-hand-side coefficient, so that the basis will produce a feasible basic solution.

Principle 5.33 | A starting basis for simplex can be obtained by making basic one variable for each constraint row, with that variable having its only nonzero coefficient in the row and coefficient sign matching that of the corresponding right-hand side. Where no standard-form variable meets these requirements, an artificial is introduced.

We can illustrate with Clever Clyde standard form (5.16). Our task is to find (or manufacture with artificial variables) a starting feasible basis. In the first constraint row we have a variable with its only nonzero coefficient there—slack x_4 . However, the coefficient of x_4 is negative, while the right-hand-side 100 is positive. An artificial variable x_6 will be necessary.

Things go somewhat easier in the second row. There x_5 has its only nonzero coefficient with a sign that matches the right-hand side. Variable x_5 will be part of the initial basis. The last two constraints of standard form have no variable with its only nonzero coefficient in their rows. Artificials x_7 and x_8 will have to be added.

Summarizing, simplex will begin Phase I with the data

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
max \mathbf{c}	1	1	1	0	0	0	0	0	
min \mathbf{d}	0	0	0	0	0	1	1	1	\mathbf{b}
\mathbf{A}	1	1	1	-1	0	1	0	0	100
	0	0	1	0	1	0	0	0	5
	-0.5	0.5	0.5	0	0	0	1	0	0
	α	$\alpha - 1$	α	0	0	0	0	1	0
	N	N	N	N	B	B	B	B	
$\mathbf{x}^{(0)}$	0	0	0	0	5	100	0	0	

Initial solution $\mathbf{x}^{(0)}$ is the basic (artificially) feasible one obtained by assigning the value zero to all nonbasics and solving for the basics.

Notice that we now have two objective function rows. The vector \mathbf{d} denotes the objective function vector for Phase I (sum of artificials). We first minimize $\mathbf{d} \cdot \mathbf{x}$, then return to maximize $\mathbf{c} \cdot \mathbf{x}$ in Phase II.

EXAMPLE 5.17: CONSTRUCTING AN ARTIFICIAL BASIS

Introduce artificial variables as necessary to construct a Phase I artificial model and corresponding starting (artificially) feasible basis for the following standard-form linear program:

$$\begin{aligned}
 \min \quad & 14x_1 \quad \quad \quad - 9x_3 \quad + x_4 \\
 \text{s.t.} \quad & 8x_1 \quad + x_2 \quad - x_3 \quad \quad \quad = 74 \\
 & \quad \quad + 4x_2 \quad - 7x_3 \quad + x_4 \quad = -22 \\
 & \quad \quad + x_2 \quad + x_3 \quad \quad \quad = 11 \\
 & x_1, x_2, x_3, x_4 \geq 0
 \end{aligned}$$

Solution: We apply principle [5.33](#). Variable x_1 appears only in the first main constraint and has the same sign as the right-hand-side 74. Thus it can be the basic variable for that constraint. Variable x_4 occurs only in the second main constraint, but its sign differs from the right-hand side; artificial variable x_5 will be needed. Another artificial x_6 is needed in the last constraint because no variable has its only nonzero coefficient there. Summarizing, the artificial model will be

$$\begin{aligned}
 \min \quad & x_5 + x_6 \\
 \text{s.t.} \quad & 8x_1 \quad + x_2 \quad - x_3 \quad \quad \quad = 74 \\
 & \quad \quad + 4x_2 \quad - 7x_3 \quad + x_4 \quad - x_5 \quad = -22 \\
 & \quad \quad + x_2 \quad + x_3 \quad \quad \quad + x_6 \quad = 11 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
 \end{aligned}$$

The starting (artificially) feasible basis is x_1, x_5 , and x_6 .

Three Possible Outcomes for Linear Programs

Recall that there are three possible outcomes for a linear programming model:

Principle 5.34 A linear program may be infeasible (have no feasible solutions), be unbounded (have arbitrarily good feasible solutions), or have finite optimal solutions.

In the discussion of Section 3.6 we detailed how we detect each case, and Algorithm 5B provides details for the simplex. The problem is infeasible if Phase I reaches optimality without reducing the sum of artificials to $= 0$ (principle [3.39]). Otherwise it has feasible solutions ([3.38]). Phase II then either detects unboundedness ([5.25]) or stops on an optimal solution ([5.27]).

Clever Clyde Infeasible Case

The Clever Clyde application of this section was contrived so that we can illustrate all three possibilities for LP outcomes and know by easy deduction that simplex is drawing the right conclusion. Think for a moment about what Clyde needs from investor 2. His own \$5000 would provide 5% of the \$100,000 minimum for the business, and investor 1 has guaranteed 50%. If $\alpha < 0.45$, there can be no feasible solution.

Take $\alpha = 0.4$. Table 5.3 shows the Phase I simplex computations leading to a conclusion of infeasibility. Beginning with x_5, x_6, x_7 , and x_8 basic, the sum of artificial variables is $\mathbf{d} \cdot \mathbf{x}^{(0)} = 100$. There are 4 available simplex directions, and 3 improve the objective. (Remember that we are minimizing, so that $\bar{d}_j < 0$ implies improvement.) The direction increasing nonbasic x_3 is chosen.

ALGORITHM 5B: TWO-PHASE SIMPLEX SEARCH

- Step 0: Artificial Model.** If there are convenient starting feasible bases for the given standard-form linear program, identify one, and proceed to Step 3. Otherwise, construct an artificial model and artificially feasible basis by adding (or subtracting) artificial variables as in principle [5.33].
- Step 1: Phase I.** Beginning from the artificially feasible basis of Step 0, apply simplex search to minimize the sum of the artificial variables subject to the constraints of the artificial model.
- Step 2: Infeasibility.** If Phase I search terminated with a minimum having artificial sum > 0 , stop; the original model is infeasible. Otherwise, use the final Phase I basis in the artificial model to identify a starting feasible basis for the original model.
- Step 3: Phase II.** Beginning from the identified starting feasible basis, apply simplex search to compute an optimal solution to the original standard-form model or demonstrate that it is unbounded.

TABLE 5.3 Simplex Computation for Clever Clyde Infeasible Case

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
max \mathbf{c}	1	1	1	0	0	0	0	0	
min \mathbf{d}	0	0	0	0	0	1	1	1	\mathbf{b}
\mathbf{A}	1	1	-1	-1	0	1	0	0	100
	0	0	1	0	1	0	0	0	5
	-0.5	0.5	0.5	0	0	0	1	0	0
	0.4	-0.6	0.4	0	0	0	0	1	0
$t = 0$	N	N	N	N	B	B	B	B	Phase I
$\mathbf{x}^{(0)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(0)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	0	0	0	-1	0.5	-0.4	$\bar{d}_1 = -0.9$
$\Delta \mathbf{x}$ for x_2	0	1	0	0	0	-1	-0.5	0.6	$\bar{d}_2 = -0.9$
$\Delta \mathbf{x}$ for x_3	0	0	1	0	-1	-1	-0.5	-0.4	$\bar{d}_3 = \boxed{-1.9}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
—	—	—	—	—	$\frac{5}{1}$	$\frac{100}{1}$	$\frac{0}{0.5}$	$\frac{0}{0.4}$	$\lambda = 0.0$
$t = 1$	N	N	B	N	B	B	B	N	Phase I
$\mathbf{x}^{(1)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(1)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	1	0	1	0	$\bar{d}_1 = 1.0$
$\Delta \mathbf{x}$ for x_2	0	1	1.5	0	-1.5	-2.5	-1.25	0	$\bar{d}_2 = \boxed{-3.75}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_8	0	0	-2.5	0	2.5	2.5	1.25	1	$\bar{d}_8 = 4.75$
—	—	—	—	—	$\frac{5}{1.5}$	$\frac{100}{2.5}$	$\frac{0}{1.25}$	—	$\lambda = 0.0$
$t = 2$	N	B	B	N	B	B	N	N	Phase I
$\mathbf{x}^{(2)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(2)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0.8	0.2	0	-0.2	-2	0	0	$\bar{d}_1 = \boxed{-2.0}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_7	0	-0.8	-1.2	0	1.2	2	1	0	$\bar{d}_7 = 3.0$
$\Delta \mathbf{x}$ for x_8	0	1	-1	0	1	0	0	1	$\bar{d}_8 = 1.0$
—	—	—	—	—	$\frac{5}{0.2}$	$\frac{100}{2}$	—	—	$\lambda = 25$
$t = 3$	B	B	B	N	N	B	N	N	Phase I
$\mathbf{x}^{(3)}$	25	20	5	0	0	50	0	0	$\mathbf{d} \cdot \mathbf{x}^{(3)} = 50$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_5	-5	-4	-1	0	1	10	0	0	$\bar{d}_5 = 10.0$
$\Delta \mathbf{x}$ for x_7	6	4	0	0	0	-10	1	0	$\bar{d}_7 = \boxed{-9.0}$
$\Delta \mathbf{x}$ for x_8	5	5	0	0	0	-10	0	1	$\bar{d}_8 = -9.0$
—	—	—	—	—	—	$\frac{50}{10}$	—	—	$\lambda = 5$
$t = 4$	B	B	B	N	N	N	B	N	Phase I
$\mathbf{x}^{(4)}$	55	40	5	0	0	0	5	0	$\mathbf{d} \cdot \mathbf{x}^{(4)} = 5$
$\Delta \mathbf{x}$ for x_4	0.6	0.4	0	1	0	0	0.1	0	$\bar{d}_4 = 0.1$
$\Delta \mathbf{x}$ for x_5	1	0	-1	0	1	0	1	0	$\bar{d}_5 = 1.0$
$\Delta \mathbf{x}$ for x_6	-0.6	-0.4	0	0	0	1	-0.1	0	$\bar{d}_6 = 0.9$
$\Delta \mathbf{x}$ for x_8	-1	1	0	0	0	0	-1	1	$\bar{d}_8 = 0.0$
“infeasible”									

The next step is to determine how far we can pursue the $\Delta \mathbf{x}$ for x_3 without losing feasibility. Here, something new occurs. Basic variables x_7 and x_8 , which were already $= 0$ in solution $\mathbf{x}^{(0)}$, are both decreased by $\Delta \mathbf{x}$. As a consequence, the largest possible step size is $\lambda = 0$.

We will have more to say about this **degenerate** case in the next section. For the moment we merely act as if λ were small but positive. Variable x_3 enters the basis, and x_7 (one of the two variables establishing λ) leaves.

The new basic solution $\mathbf{x}^{(1)}$ is identical to $\mathbf{x}^{(0)}$, but the basis that determines it has changed. As a consequence, we have new simplex directions to consider. This time only the $\Delta \mathbf{x}$ for x_2 improves. It, too, leads to a degenerate step of $\lambda = 0$, but x_2 enters the basis and x_8 leaves.

In iteration $t = 2$ we finally see real progress. Nonbasic x_1 enters, and x_5 leaves after a step of $\lambda = 25$. The objective function value (sum of artificials) is halved to $\mathbf{d} \cdot \mathbf{x}^{(3)} = 50$. Iteration $t = 3$ is similar, reducing infeasibility to $\mathbf{d} \cdot \mathbf{x}^{(4)} = 5$.

Here the progress stops. As we consider the four available simplex directions at iteration $t = 4$, we see all fail the test for improvement. That is, $\mathbf{x}^{(4)}$ is optimal in the Phase I problem. But the sum of artificials is still 5. No solution using only the original variables satisfies all constraints [i.e., the original model (with $\alpha = 0.4$) is infeasible].

EXAMPLE 5.18: DETECTING INFEASIBILITY IN SIMPLEX PHASE I

Use two-phase simplex Algorithm 5B to establish that the linear program

$$\begin{array}{ll} \max & 8x_1 + 11x_2 \\ \text{s.t.} & x_1 + x_2 \leq 2 \\ & x_1 + x_2 \geq 3 \\ & x_1, x_2 \geq 0 \end{array}$$

has no solution.

Solution: We begin by including slack variables to obtain standard-form model

$$\begin{array}{ll} \max & 8x_1 + 11x_2 \\ \text{s.t.} & x_1 + x_2 + x_3 = 2 \\ & x_1 + x_2 - x_4 = 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Variable x_3 provides a starting basic variable in the first constraint, but artificial x_5 is needed in the second constraint. We are now ready to begin Phase I simplex with initial basis $\{x_3, x_5\}$.

	x_1	x_2	x_3	x_4	x_5	
max c	8	11	0	0	0	
min d	0	0	0	0	1	b
A	1	1	1	0	0	2
	1	1	0	-1	1	3
$t = 0$	N	N	B	N	B	Phase I
$\mathbf{x}^{(0)}$	0	0	2	0	3	$\mathbf{d} \cdot \mathbf{x} = 3$
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	-1	$\bar{d}_1 = \boxed{-1}$
$\Delta \mathbf{x}$ for x_2	0	1	-1	0	-1	$\bar{d}_2 = -1$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	1	$\bar{d}_4 = +1$
	-	-	$\boxed{\frac{2}{1}}$	-	$\frac{3}{1}$	$\lambda = 2$
$t = 1$	B	N	N	N	B	Phase I
$\mathbf{x}^{(1)}$	2	0	0	0	1	$\mathbf{d} \cdot \mathbf{x} = 1$
$\Delta \mathbf{x}$ for x_2	-1	1	0	0	0	$\bar{d}_2 = 0$
$\Delta \mathbf{x}$ for x_3	-1	0	1	0	1	$\bar{d}_3 = +1$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	1	$\bar{d}_4 = +1$
						"infeasible"

The original model is infeasible because the Phase I optimum has artificial total 1.

Clever Clyde Optimal Case

Suppose now that investor 2 is willing to provide 49% of the startup capital Clever Clyde requires. There should be a finite optimal solution. Table 5.4 traces Phase I simplex computation on this $\alpha = 0.49$ case. The first two iterations closely parallel the infeasible variation of Table 5.3; two degenerate basis changes produce no actual progress.

Iteration $t = 2$ shows something new. The direction for entering nonbasic x_1 improves the objective, and the maximum feasible step of $\lambda = 50$ erases all infeasibility. The resulting solution $\mathbf{x}^{(3)}$ is 0 in all artificial components and thus corresponds to a feasible solution in the original model. Iteration $t = 3$ confirms that it is optimal in the Phase I problem, with objective function value = 0.

With a known basic feasible solution, we are ready to pass to Phase II. Table 5.5 details the computations. Now that only the standard-form variables are present, there is only one nonbasic. Its simplex direction $\Delta \mathbf{x}$ tests as improving for our maximize objective function, so it will enter the basis.

A check of ratios shows that feasible progress will stop at $\lambda = 400$ as x_5 leaves the basis. The new solution is $\mathbf{x}^{(1)} = (250, 245, 5, 400, 0)$. Now the only simplex direction is that of x_5 and it does not improve the objection function. We conclude that $\mathbf{x}^{(1)}$ is optimal. If investor 2 will provide 49% of the capital, a \$500,000 business is achievable, taking \$250,000 from investor 1, \$245,000 from investor 2, and the remaining \$5000 from Clyde.

TABLE 5.4 Phase I Simplex Computation for Clever Clyde Optimal Case

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
$\min \mathbf{d}$	0	0	0	0	0	1	1	1	\mathbf{b}
\mathbf{A}	1	1	1	-1	0	1	0	0	100
	0	1	0	0	1	0	0	0	5
	-0.50	0.50	0.50	0	0	0	1	0	0
	0.49	-0.51	0.49	0	0	0	0	1	0
$t = 0$	N	N	N	N	B	B	B	B	Phase I
$\mathbf{x}^{(0)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(0)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	0	0	0	-1	0.50	-0.49	$\bar{d}_1 = -0.99$
$\Delta \mathbf{x}$ for x_2	0	1	0	0	0	-1	-0.50	0.51	$\bar{d}_2 = -0.99$
$\Delta \mathbf{x}$ for x_3	0	0	1	0	-1	-1	-0.50	-0.49	$\bar{d}_3 = \boxed{-1.99}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
	-	-	-	-	$\frac{5}{1}$	$\frac{100}{1}$	$\frac{0}{0.50}$	$\frac{0}{0.49}$	$\lambda = 0.0$
$t = 1$	N	N	B	N	B	B	B	N	Phase I
$\mathbf{x}^{(1)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(1)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	1	0	1	0	$\bar{d}_1 = 1.0$
$\Delta \mathbf{x}$ for x_2	0	1	1.04	0	-1.04	-2.04	-1.02	0	$\bar{d}_2 = \boxed{-3.06}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_8	0	0	-2.04	0	2.04	2.04	1.02	1	$\bar{d}_8 = 4.06$
	-	-	-	-	$\frac{5}{1.04}$	$\frac{100}{2.04}$	$\frac{0}{1.02}$	-	$\lambda = 0.0$
$t = 2$	N	B	B	N	B	B	N	N	Phase I
$\mathbf{x}^{(2)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(2)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0.98	0.02	0	-0.02	-2	0	0	$\bar{d}_1 = \boxed{-2.0}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_7	0	-0.98	-1.02	0	1.02	2	1	0	$\bar{d}_7 = 3.0$
$\Delta \mathbf{x}$ for x_8	0	1	-1	0	1	0	0	1	$\bar{d}_8 = 1.0$
	-	-	-	-	$\frac{5}{0.02}$	$\frac{100}{2}$	-	-	$\lambda = 50$
$t = 3$	B	B	B	N	B	N	N	N	Phase I
$\mathbf{x}^{(3)}$	50	49	1	0	4	0	0	0	$\mathbf{d} \cdot \mathbf{x}^{(3)} = 0$
$\Delta \mathbf{x}$ for x_4	0.50	0.49	0.01	1	-0.01	0	0	0	$\bar{d}_4 = 0.0$
$\Delta \mathbf{x}$ for x_6	-0.50	-0.49	-0.01	0	0.01	1	0	0	$\bar{d}_6 = 1.0$
$\Delta \mathbf{x}$ for x_7	1	0	-1	0	1	0	1	0	$\bar{d}_7 = 1.0$
$\Delta \mathbf{x}$ for x_8	0	1	-1	0	1	0	0	1	$\bar{d}_8 = 1.0$
									"feasible"

EXAMPLE 5.19: MOVING TO SIMPLEX PHASE II

A standard-form linear program has variables x_1, \dots, x_5 . To begin two-phase simplex Algorithm 5B, we introduce artificial variables x_6, \dots, x_8 . Explain for each of the following Phase I outcomes how the algorithm should proceed.

feasible basis provided by Phase I consists of $x_1, x_2, x_3,$ and $x_5,$ with basic solution $\mathbf{x}^{(0)} = (50, 50, 0, 0, 5)$. The only nonbasic, $x_4,$ yields an improving simplex direction.

Notice that this improving simplex direction has no negative components (compare with Table 5.5). That is, the direction is not moving toward any nonnegativity constraint. It follows that there is no limit on how far we may follow the direction while retaining feasibility (principle [5.25]). Since every step in the direction improves the objective, the model must be unbounded.

5.6 DEGENERACY AND ZERO-LENGTH SIMPLEX STEPS

Our development of the simplex algorithm as a form of improving search has implicitly assumed that each iteration makes positive progress toward an optimal solution by advancing from a current extreme point solution to a superior one. If a better extreme point is encountered at each move, it is not hard to see that simplex must eventually produce an optimal solution or show that none exists (see Section 5.7).

Unfortunately, this is not always the case. It is much more typical for a simplex search to follow the pattern of Figure 5.6. Progress at some iterations is interspersed with periods of no advance. In this section we explore the degeneracy phenomenon in linear programming, which brings about simplex steps with no gain.

Degenerate Solutions

Degeneracy happens in linear programming whenever more constraints are active than the minimum number needed to define a point.

Principle 5.35 A basic feasible solution to a standard-form linear program is **degenerate** if nonnegativity constraints for some basic variables are active (i.e., if some basic variables have value = 0).

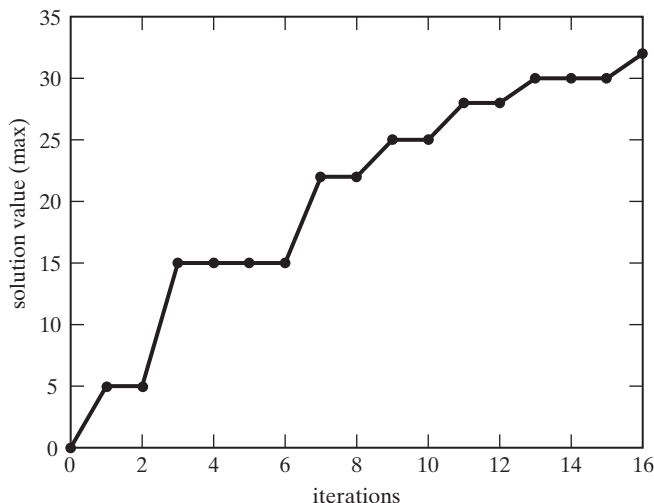


FIGURE 5.6 Typical Objective Function Progress of Simplex Search

Basic solutions require nonnegativity constraints to be active for the nonbasics (definition 5.16), but when nonnegativities also happen to be active for some basics, there are alternative choices for the basic set.

Principle 5.36 In the presence of degeneracy, several bases can compute the same basic solution.

Figure 5.7 illustrates graphically. Any three of the five inequalities B , C , H , I , and G define extreme point $\mathbf{x}^{(1)}$. In standard form, each of those constraints will correspond to a nonnegativity constraint on a slack variable. This means that $(5 - 3) = 2$ basic variables will have value $= 0$. The solution is degenerate.

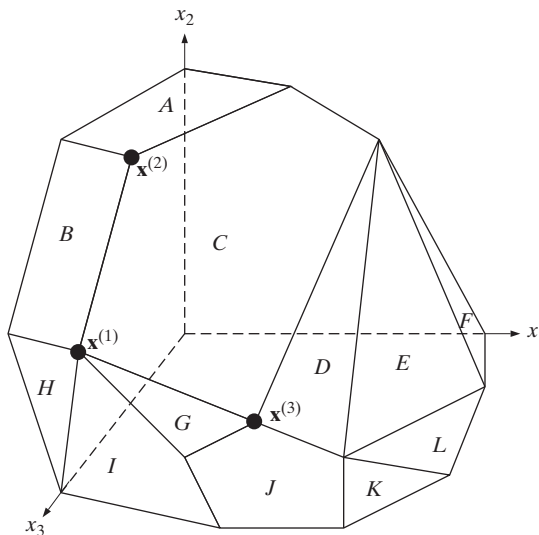


FIGURE 5.7 A Degenerate Extreme Point $\mathbf{x}^{(1)}$

EXAMPLE 5.20: RECOGNIZING DEGENERATE SOLUTIONS

Determine for each of the following basic variable sets and basic solutions to a linear program in standard form whether the solution indicated is degenerate.

(a) $B = \{x_1, x_3, x_4\}$ for $\mathbf{x} = (3, 0, 7, 2, 0, 0)$

(b) $B = \{x_1, x_2, x_5\}$ for $\mathbf{x} = (0, 8, 0, 0, 1, 0)$

Solution: We apply definition 5.35.

(a) This basic solution is nondegenerate because none of the nonnegativity constraints on basic variables are active.

(b) This basic solution is degenerate because basic variable $x_1 = 0$.

Zero-Length Simplex Steps

We can see how degeneracy inhibits simplex algorithm progress by returning to the Clever Clyde application of Table 5.3. Table 5.7 recapitulates the first three iterations. Notice that the first three basic solutions encountered were the same:

$$\mathbf{x}^{(0)} = \mathbf{x}^{(1)} = \mathbf{x}^{(2)} = (0, 0, 0, 0, 5, 100, 0, 0)$$

But this basic solution was computed by three different basic sets:

$$\begin{aligned} \{x_5, x_6, x_7, x_8\} & \text{ at } t = 0 \\ \{x_3, x_5, x_6, x_7\} & \text{ at } t = 1 \\ \{x_2, x_3, x_5, x_6\} & \text{ at } t = 2 \end{aligned}$$

The two positive components were basic in every case, but there are several choices for the two remaining basics at value = 0.

TABLE 5.7 Degenerate Iterations in Clever Clyde Application

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	
$t = 0$	N	N	N	N	B	B	B	B	Phase I (min)
$\mathbf{x}^{(0)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(0)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	0	0	0	-1	0.5	-0.4	$\bar{d}_1 = -0.9$
$\Delta \mathbf{x}$ for x_2	0	1	0	0	0	-1	-0.5	0.6	$\bar{d}_2 = -0.9$
$\Delta \mathbf{x}$ for x_3	0	0	1	0	-1	-1	-0.5	-0.4	$\bar{d}_3 = \boxed{-1.9}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
	-	-	-	-	$\frac{5}{1}$	$\frac{100}{1}$	$\frac{0}{0.5}$	$\frac{0}{0.4}$	$\lambda = 0.0$
$t = 1$	N	N	B	N	B	B	B	N	Phase I (min)
$\mathbf{x}^{(1)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(1)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	1	0	1	0	$\bar{d}_1 = 1.0$
$\Delta \mathbf{x}$ for x_2	0	1	1.5	0	-1.5	-2.5	-1.25	0	$\bar{d}_2 = \boxed{-3.75}$
$\Delta \mathbf{x}$ for x_4	0	0	1	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_8	0	0	-2.5	0	2.5	2.5	1.25	1	$\bar{d}_8 = 4.75$
	-	-	-	-	$\frac{5}{1.5}$	$\frac{100}{2.5}$	$\frac{0}{1.25}$	-	$\lambda = 0.0$
$t = 2$	N	B	B	N	B	B	N	N	Phase I (min)
$\mathbf{x}^{(2)}$	0	0	0	0	5	100	0	0	$\mathbf{d} \cdot \mathbf{x}^{(2)} = 100$
$\Delta \mathbf{x}$ for x_1	1	0.8	0.2	0	-0.2	-2	0	0	$\bar{d}_1 = \boxed{-2.0}$
$\Delta \mathbf{x}$ for x_4	0	0	0	1	0	1	0	0	$\bar{d}_4 = 1.0$
$\Delta \mathbf{x}$ for x_7	0	-0.8	-1.2	0	1.2	2	1	0	$\bar{d}_7 = 3.0$
$\Delta \mathbf{x}$ for x_8	0	1	-1	0	1	0	0	1	$\bar{d}_8 = 1.0$
	-	-	-	-	$\frac{5}{0.2}$	$\frac{100}{2}$	-	-	$\lambda = 25$
$t = 3$	B	B	B	N	N	B	N	N	Phase I (min)
$\mathbf{x}^{(3)}$	25	20	5	0	0	50	0	0	$\mathbf{d} \cdot \mathbf{x}^{(3)} = 50$

Simplex directions are structured to keep the equality constraints satisfied, and also nonnegativity constraints on nonbasic variables. But there is no guarantee that a basic component of the solution will not be decreased. If that component already happens to $= 0$, as it does in degenerate cases, the result may be a step of $\lambda = 0$.

Principle 5.37 Simplex directions that decrease basic variables already $= 0$ in a degenerate solution may produce moves with steps $\lambda = 0$.

This is precisely what happens on the initial two moves of Table 5.7. In the first case it was decreasing $\Delta x_8 = -0.4$, which produced a minimum ratio at $\lambda = 0$. In the second, it was $\Delta x_7 = -1.25$.

Progress through Changing of Bases

Computations continued in Table 5.7 as if there were nothing troublesome about $\lambda = 0$. This is the normal way to proceed.

Principle 5.38 When degenerate solutions cause the simplex algorithm to compute a step of $\lambda = 0$, the basis should be changed according to rule [5.26](#) and computations continued as if a positive step had been taken.

At $t = 2$ in Table 5.7, such perseverance was rewarded. Solution $\mathbf{x}^{(2)}$ is just as degenerate as earlier $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$, but real progress was made with a step of $\lambda = 25$.

In virtually all practical linear programs, eventual objective function progress will occur in this way (see Section 5.7 for exceptions). The reason is a much more subtle form of progress going on as we change bases at each iteration.

Principle 5.39 Simplex computations will normally escape a sequence of degenerate moves because changing basic representations of the current solution, which also changes simplex directions, eventually produce a direction along which positive progress can be achieved.

The sequence of simplex directions for x_1 is illustrated in Table 5.7. At $t = 1$ that simplex direction was not even improving. But after a change of basis at $t = 2$, the direction for x_1 is completely different. As a result it now improves without decreasing a zero-valued basis variable, and a positive step is possible.

EXAMPLE 5.21: DEALING WITH ZERO STEPS

Consider the standard-form linear program

$$\begin{aligned} \max \quad & -x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 - x_2 - x_3 = 0 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

(a) Begin with x_3 basic and apply rudimentary simplex Algorithm 5A to demonstrate that the first iteration produces a move with step $\lambda = 0$.

- (b) Continue for another iteration to demonstrate that a positive (in fact, infinite) step λ is achieved at the second step.
- (c) Explain how the change of basis the first iteration contributed to this progress.

Solution: Simplified Algorithm 5A proceeds as follows:

	x_1	x_2	x_3	
$\max \mathbf{c}$	-1	2	0	\mathbf{b}
\mathbf{A}	1	-1	-1	0
$t = 0$	N	N	B	
$\mathbf{x}^{(0)}$	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 0$
$\Delta \mathbf{x}$ for x_1	1	0	1	$\bar{c}_1 = -1$
$\Delta \mathbf{x}$ for x_2	0	1	-1	$\bar{c}_2 = 2$
	-	-	$\frac{0}{-(-1)}$	$\lambda = 0$
$t = 1$	N	B	N	
$\mathbf{x}^{(1)}$	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 0$
$\Delta \mathbf{x}$ for x_1	1	1	0	$\bar{c}_1 = \boxed{1}$
$\Delta \mathbf{x}$ for x_3	0	-1	1	$\bar{c}_3 = -2$
	-	-	-	"unbounded"

- (a) The only improving simplex direction at $t = 0$ decreases basic x_3 , which is already at degenerate value $= 0$. A step $\lambda = 0$ results.
- (b) After the change of basis at $t = 1$, the direction for x_1 now improves. It decreases no component of the solution, so $\lambda = +\infty$ is possible. The model is unbounded.
- (c) Progress became possible because the change of basis produced a different simplex direction for x_1 .

5.7 CONVERGENCE AND CYCLING WITH SIMPLEX

A numerical search algorithm is said to **converge** if iterations make steady progress toward a solution. It converges **finitely** if it is guaranteed to stop after a finite number of iterations.

Finite Convergence with Positive Steps

Does simplex converge? Yes, if there is progress at each iteration.

Principle 5.40 If each iteration of simplex search yields a positive step λ , the algorithm will stop after finitely many iterations with either an optimal solution or a conclusion of unboundedness.

With two phases we can also detect infeasibility.

To see why this is true, recall that every iteration of the simplex algorithm produces a basic feasible solution (i.e., one generated by a selected set of basic

columns). If the model in standard form has constraint matrix \mathbf{A} with m rows and n columns, there are only finitely many possible bases. Each chooses m columns from the list of n as we did in Table 5.1. This limit gives

$$\text{maximum number of bases} = \frac{n!}{m!(n - m)!}$$

where $k! \triangleq k(k - 1), \dots, (1)$. Certainly, this is a large number, but it is still finite.

When each step size λ is positive, no basis can ever repeat. Each time, we pursue a simplex direction that strictly improves the objective. Thus there is no way we could return to a previous one with poorer objective value.

EXAMPLE 5.22: BOUNDING THE NUMBER OF SIMPLEX ITERATIONS

A standard-form linear program has 10 variables and 7 main constraints. Assuming step size $\lambda > 0$ at every step, compute a finite bound on the number of iterations that Algorithm 5A might require before terminating.

Solution: If there is positive progress at each iteration, we need only bound the number of possible bases. Each would have 7 basic variables selected from among the 10 available. Thus there can be at most

$$\frac{n!}{m!(n - m)!} = \frac{10!}{7!3!} = \frac{3628,800}{5040(6)} = 120$$

iterations before simplex computation must stop.

Degeneracy and Cycling

Degenerate linear programs (definition [5.35](#)) pose another threat to convergence of the simplex method. We saw in Section 5.6 (principle [5.37](#)) that degeneracy can sometimes lead to simplex steps with $\lambda = 0$.

The concern raised by such degenerate moves of simplex search is that since no progress is being achieved in the objective function, a sequence of iterations might return to a basis we have already visited. Thereafter, the cycle would repeat, and the simplex would never stop.

Cycling can occur. Table 5.8 shows a carefully crafted example. For brevity, the table shows simplex directions only if they were actually used, but each is clearly an improving direction. Simplex passed through the degenerate basis sequence

- x_1, x_2, x_3
- x_2, x_3, x_4
- x_3, x_4, x_5
- x_3, x_5, x_6
- x_3, x_6, x_7
- x_1, x_3, x_7
- x_1, x_2, x_3

starting and ending on the same basic set.

TABLE 5.8 Example of Cycling with Simplex

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
$\min \mathbf{c}$	0	0	0	-0.75	20	-0.50	6	\mathbf{b}
	1	0	0	0.25	-8	-1	9	0
\mathbf{A}	0	1	0	0.50	-12	-0.50	3	0
	0	0	1	0	0	1	0	1
$t = 0$	B	B	B	N	N	N	N	
$\mathbf{x}^{(0)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 0$
$\Delta \mathbf{x}$ for x_4	-0.25	-0.50	0	1	0	0	0	$\bar{c}_4 = \boxed{-0.75}$
	$\boxed{\frac{0}{0.25}}$	$\frac{0}{0.50}$	-	-	-	-	-	$\lambda = 0$
$t = 1$	N	B	B	B	N	N	N	
$\mathbf{x}^{(1)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 0$
$\Delta \mathbf{x}$ for x_5	0	-4	0	32	1	0	0	$\bar{c}_5 = \boxed{-4.0}$
	-	$\boxed{\frac{0}{4}}$	-	-	-	-	-	$\lambda = 0$
$t = 2$	N	N	B	B	B	N	N	
$\mathbf{x}^{(2)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(2)} = 0$
$\Delta \mathbf{x}$ for x_6	0	0	-1	-8	-0.38	1	0	$\bar{c}_6 = \boxed{-2.0}$
	-	-	$\frac{1}{1}$	$\boxed{\frac{0}{8}}$	$\frac{0}{0.38}$	-	-	$\lambda = 0$
$t = 3$	N	N	B	N	B	B	N	
$\mathbf{x}^{(3)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(3)} = 0$
$\Delta \mathbf{x}$ for x_7	0	0	-10.5	0	-0.19	10.5	1	$\bar{c}_7 = \boxed{-2.0}$
	-	-	$\frac{1}{10.5}$	-	$\boxed{\frac{0}{0.19}}$	-	-	$\lambda = 0$
$t = 4$	N	N	B	N	N	B	B	
$\mathbf{x}^{(4)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(4)} = 0$
$\Delta \mathbf{x}$ for x_1	1	0	2	0	0	-2	-0.33	$\bar{c}_1 = \boxed{-1.0}$
	-	-	-	-	-	$\boxed{\frac{0}{2}}$	$\frac{0}{0.33}$	$\lambda = 0$
$t = 5$	B	N	B	N	N	N	B	
$\mathbf{x}^{(5)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(5)} = 0$
$\Delta \mathbf{x}$ for x_2	3	1	0	0	0	0	-0.33	$\bar{c}_2 = \boxed{-2.0}$
	-	-	-	-	-	-	$\boxed{\frac{0}{0.33}}$	$\lambda = 0$
$t = 6$	B	B	B	N	N	N	N	
$\mathbf{x}^{(6)}$	0	0	1	0	0	0	0	$\mathbf{c} \cdot \mathbf{x}^{(6)} = 0$

Despite examples like the one in Table 5.8, cycling is quite rare. In fact, small examples of LPs that cycle are so rare that the one in Table 5.8 is rather famous.

For the overwhelming majority of LP models it is safe to proceed just as set out in principle [5.38]. If a step λ happens to $= 0$, act as if it were small but positive, and continue the search.

Principle 5.41 It is usually safe to assume that cycling will not occur in applied linear programming models and thus that simplex search will converge finitely.

In those rare cases where cycling is a threat, careful choice of entering and leaving basic variables can restore simplex convergence. However, details of such **anticycling** rules are beyond the scope of this book.

5.8 DOING IT EFFICIENTLY: REVISED SIMPLEX

The approach we have taken to simplex search so far emphasizes the underlying logic of the algorithm. However, many of the steps can be executed much more efficiently with the aid of a bit of matrix algebra. In this section we outline the **revised simplex algorithm** which is at the heart of large-scale codes.

Computations with Basis Inverses

Rudimentary Algorithm 5A of Section 4.3 solves a great many systems of linear equations. We must solve one such system to find the first basic solution. Then at each iteration a linear system is solved for every simplex direction.

The first insight leading to computational efficiency is to realize that the linear systems for any iteration have the same left-hand side. All involve solving for weights on basic columns to express another vector.

Consider, for example, iteration $t = 1$ of the Top Brass computation detailed in Table 5.2. With all nonbasic variables fixed = 0 (definition [\[5.16\]](#)), the current basic solution can be completed by solving the linear system

$$\begin{array}{cccccc} +1x_1 & + 0x_4 & + 0x_5 & + 0x_6 & = & 1000 \\ +0x_1 & + 1x_4 & + 0x_5 & + 0x_6 & = & 1500 \\ +1x_1 & + 0x_4 & + 1x_5 & + 0x_6 & = & 1750 \\ +4x_1 & + 0x_4 & + 0x_5 & + 1x_6 & = & 4800 \end{array}$$

Simplex directions are similar. Definition [\[5.21\]](#) implicitly requires solving for a representation of the negative of the constraint column for x_j as we derive the corresponding simplex direction. For example, basic parts of such simplex directions $\Delta \mathbf{x}$ for Top Brass iteration $t = 1$ all come from linear equation systems

$$\begin{array}{cccccc} +1\Delta x_1 & + 0\Delta x_4 & + 0\Delta x_5 & + 0\Delta x_6 & = & -a_{1,j} \\ +0\Delta x_1 & + 1\Delta x_4 & + 0\Delta x_5 & + 0\Delta x_6 & = & -a_{2,j} \\ +1\Delta x_1 & + 0\Delta x_4 & + 1\Delta x_5 & + 0\Delta x_6 & = & -a_{3,j} \\ +4\Delta x_1 & + 0\Delta x_4 & + 0\Delta x_5 & + 1\Delta x_6 & = & -a_{4,j} \end{array}$$

Think of the collection of columns for basic variables as a **basis matrix**:

$$\mathbf{B} \triangleq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 \end{pmatrix}$$

All needed linear systems for this \mathbf{B} can be solved easily if we compute just once the **inverse** of matrix \mathbf{B} :

$$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix}$$

Primer 6 reviews the key facts about matrix inverses. For our purposes the important observation is that multiplication by a basis inverse can solve a corresponding system of linear constraints.

PRIMER 6: IDENTITY AND INVERSE MATRICES

Primer 3 reviewed general matrix arithmetic. One very special form of square matrix is an **identity** matrix, denoted \mathbf{I} , that leaves any matrix or vector unchanged after multiplication. That is,

$$\mathbf{IA} = \mathbf{AI} = \mathbf{A} \quad \text{and} \quad \mathbf{Ix} = \mathbf{xI} = \mathbf{x}$$

for any matrix \mathbf{A} and any vector \mathbf{x} . The unique matrices that have this property are those of the form

$$\mathbf{I} \triangleq \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

with 1 down the diagonal and 0 off-diagonal. The dimension of an \mathbf{I} matrix is understood to be whatever is required for multiplication to be defined.

For every square nonsingular matrix \mathbf{M} , there is a unique square nonsingular matrix \mathbf{M}^{-1} such that

$$\mathbf{MM}^{-1} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{1}$$

Matrix \mathbf{M}^{-1} is called the **inverse** of \mathbf{M} and denoted by a -1 exponent. Matrices that are not square or are singular (see Primer 5) have no inverses.

As an example, the inverse of the matrix

$$\mathbf{M} \triangleq \begin{pmatrix} 5 & -1 & 3 \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ 7 & 4 & 0 \end{pmatrix} \quad \text{is} \quad \mathbf{M}^{-1} = \begin{pmatrix} \frac{2}{3} & 4 & -\frac{1}{3} \\ -\frac{7}{6} & -7 & \frac{5}{6} \\ -\frac{7}{6} & -9 & \frac{5}{6} \end{pmatrix}$$

because

$$\mathbf{MM}^{-1} = \begin{pmatrix} 5 & -1 & 3 \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ 7 & 4 & 0 \end{pmatrix} \begin{pmatrix} \frac{2}{3} & 4 & -\frac{1}{3} \\ -\frac{7}{6} & -7 & \frac{5}{6} \\ -\frac{7}{6} & -9 & \frac{5}{6} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(Continued)

For 2 by 2 nonsingular matrices, inverses have the simple form

$$\begin{pmatrix} p & q \\ r & s \end{pmatrix}^{-1} = \frac{1}{ps-qr} \begin{pmatrix} s & -q \\ -r & p \end{pmatrix}, \quad \text{so} \quad \begin{pmatrix} 2 & 3 \\ -4 & 5 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{5}{22} & -\frac{3}{22} \\ \frac{4}{22} & \frac{2}{22} \end{pmatrix}$$

Many computational procedures are available to find matrix inverses in higher dimensions, but a calculator or computer is usually required.

When a matrix has an inverse, the inverse of its transpose is the transpose of its inverse. Thus for \mathbf{M} above,

$$(\mathbf{M}^T)^{-1} = \begin{pmatrix} 5 & 0 & 7 \\ -1 & \frac{1}{2} & 4 \\ 3 & -\frac{1}{2} & 0 \end{pmatrix}^{-1} = (\mathbf{M}^{-1})^T = \begin{pmatrix} \frac{2}{3} & -\frac{7}{6} & -\frac{7}{6} \\ 4 & -7 & -9 \\ -\frac{1}{3} & \frac{5}{6} & \frac{5}{6} \end{pmatrix}$$

Our principal use of matrix inverses is in computations involving systems of linear equations. In particular, if \mathbf{Q} is a square nonsingular matrix, the unique solution to the equation system is

$$\mathbf{Q}\mathbf{x} = \mathbf{r} \quad \text{is} \quad \mathbf{x} = \mathbf{Q}^{-1}\mathbf{r}$$

This follows because multiplication of the original system on the left by \mathbf{Q}^{-1} gives $\mathbf{Q}^{-1}\mathbf{Q}\mathbf{x} = \mathbf{Q}^{-1}\mathbf{r}$, and the left-hand side is by definition $\mathbf{I}\mathbf{x} = \mathbf{x}$. Similarly, multiplication by \mathbf{Q}^{-1} on the right shows that the unique solution to

$$\mathbf{v}\mathbf{Q} = \mathbf{h} \quad \text{is} \quad \mathbf{v} = \mathbf{h}\mathbf{Q}^{-1}$$

Illustrating with matrix \mathbf{M} above and $\mathbf{r} \triangleq (2, 1, 2)$, the unique solution \mathbf{x} to the system $\mathbf{M}\mathbf{x} = \mathbf{r}$, or

$$\begin{array}{rclcl} -5x_1 & -1x_2 & +3x_3 & = & 2 \\ -0x_1 & +\frac{1}{2}x_2 & -\frac{1}{2}x_3 & = & 1 \\ +7x_1 & -4x_2 & +0x_3 & = & 2 \end{array} \quad \text{is} \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{r} = \begin{pmatrix} \frac{14}{3} \\ -\frac{23}{3} \\ -\frac{29}{3} \end{pmatrix}$$

With $\mathbf{h} = (-1, 1, 1)$ the unique solution \mathbf{v} to the system $\mathbf{v}\mathbf{M} = \mathbf{h}$, or

$$\begin{array}{rclcl} +5v_1 & +0v_2 & +7v_3 & = & -1 \\ -1v_1 & +\frac{1}{2}v_2 & +4v_3 & = & 1 \\ +3v_1 & -\frac{1}{2}v_2 & +0v_3 & = & 1 \end{array} \quad \text{is} \quad \mathbf{v} = \mathbf{h}\mathbf{M}^{-1} = \begin{pmatrix} -\frac{5}{3} \\ -20 \\ 2 \end{pmatrix}$$

Principle 5.42 Using a representation of the current basis matrix inverse \mathbf{B}^{-1} , basic components of the corresponding basic solution can be computed by matrix multiplication as $\mathbf{B}^{-1}\mathbf{b}$, and basic components of simplex directions for nonbasic variables x_j are $-\mathbf{B}^{-1}\mathbf{a}^{(j)}$, where \mathbf{b} is the vector of right-hand-side coefficients, and $\mathbf{a}^{(j)}$ is the constraint column for x_j .

We can illustrate with the basis inverse above. Basic components of the corresponding basic solution can be computed (principle [5.42]):

$$\begin{pmatrix} x_1 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \mathbf{B}^{-1}\mathbf{b} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1000 \\ 1500 \\ 1750 \\ 4800 \end{pmatrix} = \begin{pmatrix} 1000 \\ 1500 \\ 750 \\ 800 \end{pmatrix}$$

Similarly, basic components of the simplex direction for x_2 are

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_4 \\ \Delta x_5 \\ \Delta x_6 \end{pmatrix} = -\mathbf{B}^{-1}\mathbf{a}^{(2)} = -\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ -2 \end{pmatrix}$$

EXAMPLE 5.23: COMPUTING WITH BASIS INVERSES

Consider the standard-form linear program

$$\begin{aligned} \min \quad & 9x_1 + 3x_2 + 1x_4 \\ \text{s.t.} \quad & 2x_1 + 1x_2 - 1x_3 = 12 \\ & 1x_1 + 9x_3 + 2x_4 = 5 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

and assume that x_1 and x_2 are basic.

- Identify the current basis matrix.
- Compute the current basis inverse.
- Use your basis inverse to compute the current basic solution.
- Use the basis matrix inverse to compute all simplex directions at the current solution.

Solution:

- Columns for basic variables x_1 and x_2 make the basis matrix

$$\mathbf{B} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$$

- Applying the formula in Primer 6 yields

$$\mathbf{B}^{-1} = \frac{1}{(2)(0) - (1)(1)} \begin{pmatrix} 0 & -1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}$$

- Using computation [5.42], the components of the basic solution for basic variables are

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathbf{B}^{-1}\mathbf{b} = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 12 \\ 5 \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

Thus the full basic solution is $\mathbf{x} = (5, 2, 0, 0)$.

Here $\Delta x_{j\text{th}}$ denotes the simplex direction component for the j th basic variable, and Δx_{leave} indicates the component for the leaving basic. The matrix is almost an identity. Only the column position of the leaving basic variable is replaced. The j th row of that special column is $-\Delta x_{j\text{th}}/\Delta x_{\text{leave}}$, except on the diagonal, where it is $-1/\Delta x_{\text{leave}}$.

Again using iteration $t = 1$ of Table 5.2, the required pivot replaces x_6 in the fourth position of the old basis $\{x_1, x_4, x_5, x_6\}$ by x_2 . The corresponding \mathbf{E} matrix is

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & -\frac{0}{-2} \\ 0 & 1 & 0 & -\frac{-1}{-2} \\ 0 & 0 & 1 & -\frac{-1}{-2} \\ 0 & 0 & 0 & -\frac{1}{-2} \end{pmatrix}$$

The basis inverse after the iteration can then be computed from the old \mathbf{B}^{-1} above as (pivot formula [5.43](#))

$$\begin{aligned} \text{new } \mathbf{B}^{-1} &\triangleq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 4 & 0 & 0 & 2 \end{pmatrix}^{-1} \\ &= \mathbf{E}(\text{old } \mathbf{B}^{-1}) \\ &= \begin{pmatrix} 1 & 0 & 0 & -\frac{0}{-2} \\ 0 & 1 & 0 & -\frac{-1}{-2} \\ 0 & 0 & 1 & -\frac{-1}{-2} \\ 0 & 0 & 0 & -\frac{1}{-2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & -0.5 \\ 1 & 0 & 1 & -0.5 \\ -2 & 0 & 0 & 0.5 \end{pmatrix} \end{aligned}$$

Although it is beyond the scope of this book to provide details, update formula [5.43](#) also hints at how production simplex computer codes actually represent basis inverses. Since each inverse is just \mathbf{E} times the last, we could represent the current inverse simply by remembering its initial form along with the \mathbf{E} s from iterations so

far. Then the process of multiplying a vector by \mathbf{B}^{-1} , required, say, to compute a simplex direction, can be accomplished just as well by multiplying in turn by the initial form and all the \mathbf{E} s.

EXAMPLE 5.24: UPDATING BASIS INVERSES

Assume in the linear program of Example 5.23 that nonbasic x_3 enters the basis and x_1 leaves. Use the simplex direction results of part (d) to compute update matrix \mathbf{E} , and apply the result to compute the new basis inverse.

Solution: From Example 5.23(d), the basic components of the simplex direction for x_3 are $\Delta x_1 = -9$ and $\Delta x_2 = 19$. With x_1 leaving the basic set $\{x_1, x_2\}$, schema (5.18) gives

$$\mathbf{E} = \begin{pmatrix} -\frac{1}{\Delta x_1} & 0 \\ \frac{\Delta x_2}{\Delta x_1} & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{9} & 0 \\ \frac{19}{9} & 1 \end{pmatrix}$$

Using this \mathbf{E} and the old basis inverse of Example 5.23(b), updating [5.43] gives

$$\text{new } \mathbf{B}^{-1} = \mathbf{E}(\text{old } \mathbf{B}^{-1}) = \begin{pmatrix} \frac{1}{9} & 0 \\ \frac{19}{9} & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{9} \\ 1 & \frac{1}{9} \end{pmatrix}$$

Basic Variable Sequence in Revised Simplex

One slightly confusing aspect of equation (5.17) is its reference to basic variables by sequential position (1st, 2nd, etc.) rather than the original subscript. For example, the pivot above for Top Brass iteration $t = 1$ entered x_2 to replace x_6 as the 4th basic variable. The new basis thus has

$$x_{1\text{st}} \triangleq x_1, \quad x_{2\text{nd}} \triangleq x_4, \quad x_{3\text{rd}} \triangleq x_5, \quad x_{4\text{th}} \triangleq x_2$$

and columns are arranged in that order in the foregoing expressions.

Principle 5.44 Variables in revised simplex enter the basis in the same sequential positions as the variables they replace.

This sequence-based numbering of basis variables recurs in several elements of revised simplex computation. Readers doing calculations by hand will find it somewhat tedious. Keep in mind, however, that revised simplex ideas are designed for efficient computer calculation, not ease of human understanding.

EXAMPLE 5.25: TRACKING BASIC VARIABLE SEQUENCE

Assume that revised simplex computation on a linear program in standard form begins with basic variable sequence $\{x_1, x_2, x_3\}$. On subsequent pivots x_6 replaces x_1 , x_5 replaces x_3 , and x_4 replaces x_6 . Show the basic variable sequences for those three iterations.

Solution: The initial basis has

$$x_{1\text{st}} \triangleq x_1, \quad x_{2\text{nd}} \triangleq x_2, \quad x_{3\text{rd}} \triangleq x_3$$

As variables enter and leave, it changes to

$$x_{1\text{st}} \triangleq x_6, \quad x_{2\text{nd}} \triangleq x_2, \quad x_{3\text{rd}} \triangleq x_3$$

$$x_{1\text{st}} \triangleq x_6, \quad x_{2\text{nd}} \triangleq x_2, \quad x_{3\text{rd}} \triangleq x_5$$

$$x_{1\text{st}} \triangleq x_4, \quad x_{2\text{nd}} \triangleq x_2, \quad x_{3\text{rd}} \triangleq x_5$$

Computing Reduced Costs by Pricing

Simplex search chooses a move direction (or concludes optimality) by computing the reduced cost of definition [5.22] for nonbasic variables x_j , that is,

$$\bar{c}_j \triangleq \mathbf{c} \cdot \Delta \mathbf{x} \triangleq \sum_{k=1}^n c_k \Delta x_k$$

where $\Delta \mathbf{x}$ is the simplex direction for nonbasic x_j .

The only reason for computing most of the simplex directions at any iteration is to determine reduced costs. We actually use just one simplex direction in the move. Obviously, a great deal of computation could be saved if we could obtain the \bar{c}_j without generating full simplex directions.

To see exactly how to do that, recall that the simplex direction $\Delta \mathbf{x}$ for nonbasic x_j is $= 1$ in the j th component and $= 0$ on components for all other nonbasics (definition [5.21]). Thus we could write

$$\bar{c}_j = c_j + \sum_{k \in B} c_k \Delta x_k \quad (5.18)$$

where B denotes the set of basic variable subscripts. For example, in iteration $t = 1$ of the Top Brass application,

$$\bar{c}_3 = 0 + [12(-1) + 0(0) + 0(1) + 0(4)] = -12$$

We now know how to determine the basic directional components Δx_k in expression (5.19) using a representation of the basis inverse (principle [5.42]).

$$\begin{pmatrix} \Delta x_{1\text{st}} \\ \Delta x_{2\text{nd}} \\ \vdots \\ \Delta x_{m\text{th}} \end{pmatrix} = \mathbf{B}^{-1} \begin{pmatrix} -a_{1,k} \\ -a_{2,k} \\ \vdots \\ -a_{m,k} \end{pmatrix}$$

where again $\Delta x_{j\text{th}}$ is the component of simplex direction $\Delta \mathbf{x}$ on the j th basic variable. Denoting corresponding basic objective function coefficients by $c_{j\text{th}}$ and substituting produces the pricing vector for the current iteration.

Definition 5.45 | The **pricing vector** \mathbf{v} corresponding to the current basis $\{x_{1\text{st}}, x_{2\text{nd}}, \dots, x_{m\text{th}}\}$ is

$$\mathbf{v} \triangleq (c_{1\text{st}}, c_{2\text{nd}}, \dots, c_{m\text{th}}) \mathbf{B}^{-1}$$

Then, using formula [5.42](#), equation (5.19) becomes

$$\begin{aligned} \bar{c}_j &= c_j + (c_{1\text{st}}, c_{2\text{nd}}, \dots, c_{m\text{th}}) \cdot \begin{pmatrix} \Delta \mathbf{x}_{1\text{st}} \\ \Delta \mathbf{x}_{2\text{nd}} \\ \vdots \\ \Delta \mathbf{x}_{m\text{th}} \end{pmatrix} \\ &= c_j - (c_{1\text{st}}, c_{2\text{nd}}, \dots, c_{m\text{th}}) \mathbf{B}^{-1} \begin{pmatrix} a_{1,j} \\ a_{2,j} \\ \vdots \\ a_{m,j} \end{pmatrix} \\ &= c_j - \mathbf{v} \cdot \begin{pmatrix} a_{1,j} \\ a_{2,j} \\ \vdots \\ a_{m,j} \end{pmatrix} \end{aligned}$$

We see that reduced costs \bar{c}_j can be computed without explicitly generating simplex directions.

Principle 5.46 Reduced costs can be computed directly from original data as $\bar{c}_j = c_j - \mathbf{v} \cdot \mathbf{a}^{(j)}$, where \mathbf{v} is the pricing vector of the current iteration and $\mathbf{a}^{(j)}$ is the original constraint matrix column for variable x_k .

Again we illustrate with iteration $t = 1$ of Table 5.2. The pricing vector for that iteration is

$$\mathbf{v} = (c_{1\text{st}}, c_{2\text{nd}}, \dots, c_{m\text{th}}) \mathbf{B}^{-1} = (12, 0, 0, 0) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix} = (12, 0, 0, 0)$$

Then reduced costs follow as

$$\begin{aligned} \bar{c}_2 &= c_2 - \mathbf{v} \cdot \mathbf{a}^{(2)} = 9 - (12, 0, 0, 0) \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \end{pmatrix} = 9 \\ \bar{c}_3 &= c_3 - \mathbf{v} \cdot \mathbf{a}^{(3)} = 0 - (12, 0, 0, 0) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = -12 \end{aligned}$$

Formula [5.46](#) changes absolutely nothing about the value of the reduced costs, but it decreases enormously the effort to compute them. We now need to compute only one simplex direction explicitly—the one that the nonbasic variable was chosen to enter.

EXAMPLE 5.26: COMPUTING REDUCED COSTS BY PRICING

Consider the linear program

$$\begin{array}{llllll} \min & 3x_1 & + & 100x_2 & + & 12x_3 & - & 8x_4 \\ \text{s.t.} & 3x_1 & + & 1x_2 & & - & 1x_3 & & = & 90 \\ & -1x_1 & - & 1x_2 & & & & + & 1x_4 & = & 22 \\ & & & & & & & & & & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Assume that the current basic variable sequence is $\{x_3, x_1\}$.

- Compute the corresponding basis inverse.
- Compute the associated pricing vector.
- Compute the reduced costs on nonbasic variables without generating any simplex directions.

Solution:

(a) In the sequence given, the basis inverse matrix is

$$\mathbf{B}^{-1} = \begin{pmatrix} -1 & 3 \\ 0 & -1 \end{pmatrix}^{-1} = \begin{pmatrix} -1 & -3 \\ 0 & -1 \end{pmatrix}$$

(b) Applying definition [5.46](#), the associated pricing vector is

$$\mathbf{v} = (c_{1\text{st}}, c_{2\text{nd}})\mathbf{B}^{-1} = (12, 3) \begin{pmatrix} -1 & -3 \\ 0 & -1 \end{pmatrix} = (-12, -39)$$

(c) We can now compute all reduced costs via principle [5.46](#).

$$\begin{aligned} \bar{c}_2 &= c_2 - \mathbf{v} \cdot \mathbf{a}^{(2)} = 100 - (-12, -39) \cdot (1, -1) = 73 \\ \bar{c}_4 &= c_4 - \mathbf{v} \cdot \mathbf{a}^{(4)} = -8 - (-12, -39) \cdot (0, 1) = 31 \end{aligned}$$

Revised Simplex Search of Top Brass Application

We now have all the pieces of the revised simplex procedure detailed in Algorithm 5C. Table 5.9 shows the complete revised simplex search of the Top Brass Trophy application done earlier in Table 5.2. It is important to note that the sequence of basic feasible solutions encountered is exactly the same in the two tables. Only the method of computation at intermediate steps has changed.

To avoid confusion about sequential numbering of basis variables, bases in Table 5.9 are marked with their positions. For example, the basis to start iteration $t = 3$ has x_1 as the first basic, x_2 as the fourth basic, x_3 as the third basic, and x_4 as the second basic. The variables x_5 and x_6 are nonbasic.

In Table 5.9 we maintain an updated representation of the basis matrix inverse at each iteration. That of iteration $t = 0$ is an identity matrix because the corresponding \mathbf{B} is an identity, and it is easy to check that the inverse of an identity matrix is an identity matrix.

TABLE 5.9 Revised Simplex Search of Top Brass Trophy Application

	x_1	x_2	x_3	x_4	x_5	x_6	
max c	12	9	0	0	0	0	b
	1	0	1	0	0	0	1000
	0	1	0	1	0	0	1500
A	1	1	0	0	1	0	1750
	4	2	0	0	0	1	4800
$t = 0$	N	N	1st	2nd	3rd	4th	
$\mathbf{x}^{(0)}$	0	0	1000	1500	1750	4800	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 0$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$							
\bar{c}_j	12	9	0	0	0	0	
$\Delta \mathbf{x}$ for x_1	1	0	-1	0	-1	-4	
	-	-	$\frac{1000}{-(-1)}$	-	$\frac{1750}{-(-1)}$	$\frac{4800}{-(-1)}$	$\lambda = 1000$
$t = 1$	1st	N	N	2nd	3rd	4th	
$\mathbf{x}^{(1)}$	1000	0	0	1500	750	800	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 12,000$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 0 \end{pmatrix}$							
\bar{c}_j	0	9	-12	0	0	0	
$\Delta \mathbf{x}$ for x_2	0	1	0	-1	-1	-2	
	-	-	-	$\frac{1500}{-(-1)}$	$\frac{750}{-(-1)}$	$\frac{800}{-(-2)}$	$\lambda = 400$
$t = 2$	1st	4th	N	2nd	3rd	N	
$\mathbf{x}^{(2)}$	1000	400	0	1100	350	0	$\mathbf{c} \cdot \mathbf{x}^{(2)} = 15,600$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & -0.5 \\ 1 & 0 & 1 & -0.5 \\ -2 & 0 & 0 & 0.5 \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} -6 \\ 0 \\ 0 \\ 4.5 \end{pmatrix}$							
\bar{c}_j	0	0	6	0	0	-4.5	
$\Delta \mathbf{x}$ for x_3	-1	2	1	-2	-1	0	
	$\frac{1000}{-(-1)}$	-	-	$\frac{1100}{-(-2)}$	$\frac{350}{-(-1)}$	-	$\lambda = 350$
$t = 3$	1st	4th	3rd	2nd	N	N	
$\mathbf{x}^{(3)}$	650	1100	350	400	0	0	$\mathbf{c} \cdot \mathbf{x}^{(3)} = 17,700$
$\mathbf{B}^{-1} = \begin{pmatrix} 0 & 0 & -1 & 0.5 \\ 0 & 1 & -2 & 0.5 \\ 1 & 0 & 1 & -0.5 \\ 0 & 0 & 2 & -0.5 \end{pmatrix} \cdot \mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ 6 \\ 1.5 \end{pmatrix}$							
\bar{c}_j	0	0	0	0	-6	-1.5	“optimal”

ALGORITHM 5C: REVISED SIMPLEX SEARCH FOR LINEAR PROGRAMS

Step 0: Initialization. Choose any starting feasible basis, and construct a representation of the corresponding basic column matrix inverse \mathbf{B}^{-1} . Then use that representation to solve $\mathbf{B}\mathbf{x}^B = \mathbf{b}$ for basic components of the initial solution $\mathbf{x}^{(0)}$, set all nonbasic $x_j^{(0)} \leftarrow 0$, and initialize solution index $t \leftarrow 0$.

Step 1: Pricing. Use the representation of the current basic column inverse \mathbf{B}^{-1} to solve $\mathbf{v}\mathbf{B} = \mathbf{c}^B$ for pricing vector \mathbf{v} where \mathbf{c}^B is the vector of basic objective function coefficients. Then evaluate reduced costs $\bar{c}_j \leftarrow c_j - \mathbf{v} \cdot \mathbf{a}^{(j)}$ for each nonbasic x_j .

Step 2: Optimality. If no $\bar{c}_j > 0$ for a maximize problem (or no $\bar{c}_j < 0$ for a minimize), then stop; current solution $\mathbf{x}^{(t)}$ is globally optimal. Otherwise, choose an entering nonbasic variable x_p with improving \bar{c}_p .

Step 3: Simplex Direction. Construct the simplex direction $\Delta\mathbf{x}^{(t+1)}$ for nonbasic variable x_p using the representation of current basic column inverse \mathbf{B}^{-1} to solve $\mathbf{B}\Delta\mathbf{x} = -\mathbf{a}^{(p)}$ for basic components.

Step 4: Step Size. If there is no limit on feasible moves in simplex direction $\Delta\mathbf{x}^{(t+1)}$ (all components are nonnegative), stop; the given model is unbounded. Otherwise, choose leaving variable x_r so that

$$\frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}} = \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\} \quad \text{and set} \quad \lambda \leftarrow \frac{x_r^{(t)}}{-\Delta x_r^{(t+1)}}$$

Step 5: New Point and Basis. Compute the new solution

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta\mathbf{x}^{(t+1)}$$

and replace x_r in the basis by x_p . Also construct the associated pivot matrix \mathbf{E} and update the basis inverse representation as $\mathbf{E}\mathbf{B}^{-1}$. Then advance $t \leftarrow t + 1$, and return to Step 1.

The next three iterations update \mathbf{B}^{-1} via formula [5.43](#). Required \mathbf{E} matrices are, respectively,

$$\begin{pmatrix} -\frac{1}{-1} & 0 & 0 & 0 \\ -\frac{0}{-1} & 1 & 0 & 0 \\ -\frac{-1}{-1} & 0 & 1 & 0 \\ -\frac{-4}{-1} & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & -\frac{0}{-2} \\ 0 & 1 & 0 & -\frac{-1}{-2} \\ 0 & 0 & 1 & -\frac{-1}{-2} \\ 0 & 0 & 0 & -\frac{1}{-2} \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 1 & 0 & -\frac{-1}{-1} & 0 \\ 0 & 1 & -\frac{-2}{-1} & 0 \\ 0 & 0 & -\frac{1}{-1} & 0 \\ 0 & 0 & -\frac{2}{-1} & 1 \end{pmatrix}$$

At each iteration of revised simplex, reduced costs \bar{c}_j are computed directly using pricing vector \mathbf{v} . Table 5.9 shows all $\mathbf{v}\mathbf{S}$, and the resulting \bar{c}_j , boxing the one

selected as improving at each iteration. The simplex direction associated with that improving reduced cost, which follows immediately in the table, is the only one explicitly generated at each iteration.

Once a move direction has been selected, the revised simplex makes the step size λ decision exactly as in the earlier simplex algorithm. A new basic solution $\mathbf{x}^{(t+1)}$ is computed, a leaving variable is selected, the basis inverse representation is updated, and processing continues.

5.9 SIMPLEX WITH SIMPLE UPPER AND LOWER BOUNDS

In Section 5.1 we showed how LP standard form converts all inequalities of a problem to nonnegativity constraints $x_j \geq 0$. In later sections we demonstrated how easy this form makes construction of basic solutions, simplex directions, and so on.

Almost all those simplifications would follow just as well if we allowed inequalities to take the slightly more general **simple lower-bound** form

$$x_j \geq \ell_j$$

or **simple upper-bound** form

$$x_j \leq u_j$$

Here the ℓ_j and u_j are given model constants. Nonnegativities are just the special case of simple lower bounds with $\ell_j = 0$. In this section we explore briefly how the revised simplex method of Section 5.8 can be modified to encompass simple lower and upper bounds.

Lower and Upper-Bounded Standard Form

The standard form for linear programs with simple lower and upper bounds is

$$\begin{aligned} \min \text{ (or max) } & \sum_{j=1}^n c_j x_j \\ \text{s.t.} & \sum_{j=1}^n a_{i,j} x_j = b_i \quad \text{for all } i = 1, 2, \dots, m \\ & u_j \geq x_j \geq \ell_j \quad \text{for all } j = 1, 2, \dots, n \end{aligned}$$

Collecting lower and upper bounds in vectors ℓ and \mathbf{u} produces the corresponding matrix form.

Definition 5.47 In matrix notation, the **lower and upper-bounded standard form** for linear programs is

$$\begin{aligned} \min \text{ (or max) } & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{u} \geq \mathbf{x} \geq \ell \end{aligned}$$

Feasibility requires that $u_j \geq \ell_j$, and we allow the possibility that $u_j = \infty$, or $\ell_j = -\infty$, or both.

Often, allowing simple bounds in this way considerably reduces the number of rows m in main constraints $\mathbf{Ax} = \mathbf{b}$. The result is a savings in most of the complex steps of the simplex algorithm, such as generating simplex directions and updating \mathbf{B}^{-1} . For example, the Top Brass model of Figure 5.5 required that $m = 4$ rows in Tables 5.2 and 5.9; it needs only that $m = 2$ in lower or upper-bounded form.

	x_1	x_2	x_5	x_6	
max \mathbf{c}	12	9	0	0	\mathbf{b}
	1	1	1	0	1750
	4	2	0	1	4800
ℓ	0	0	0	0	
\mathbf{u}	1000	1500	∞	∞	

Notice that only two slack variables are now required. We continue to call them x_5 and x_6 , for consistency with Figure 5.5.

EXAMPLE 5.27: CONSTRUCTING LOWER-AND UPPER-BOUNDED STANDARD FORM

Place the following linear program in lower- and upper-bounded standard form.

$$\begin{aligned}
 \min \quad & 3x_1 - x_2 + x_3 + 11x_4 \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 50 \\
 & x_1 \leq 30 \\
 & 3x_1 + x_4 \leq 90 \\
 & 9x_3 + x_4 \geq 5 \\
 & x_2 \leq 10 \\
 & x_3 \geq -2 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Solution: Nonnegative slack variables x_5 and x_6 are required in the third and fourth constraints, but the rest of the model, including unrestricted variable x_4 , can be accommodated by lower- and upper-bounded standard form [5.47]. A full coefficient array is

	x_1	x_2	x_3	x_4	x_5	x_6	
min \mathbf{c}	3	-1	1	11	0	0	\mathbf{b}
	1	1	1	1	0	0	50
	3	0	0	1	1	0	90
	0	0	9	1	0	-1	5
ℓ	0	0	-2	$-\infty$	0	0	
\mathbf{u}	30	10	∞	∞	∞	∞	

Basic Solutions with Lower and Upper Bounds

The main idea of basic solutions is to make active the inequalities for all nonbasic variables, then solve for the basics. With lower and upper bounds there may be choices for each nonbasic x_j , depending on which of ℓ_j and u_j are finite.

Principle 5.48 Basic solutions in lower- and upper-bounded simplex set non-basics to either (finite) lower bound ℓ_j or (finite) upper bound u_j and solve $\mathbf{Ax} = \mathbf{b}$ for the basics.

The solution is **basic feasible** if computed basic values satisfy their lower- and upper-bound constraints

$$u_j \geq x_j \geq \ell_j \text{ for all } j \text{ basic}$$

The linear system to be solved in computing basic variable values has the form

$$\sum_{j \in B} a_{i,j}x_j^{(t)} = b_i - \sum_{j \in L} a_{i,j}\ell_j - \sum_{j \in U} a_{i,j}u_j \quad \text{for all } i = 1, \dots, m \quad (5.19)$$

where L indexes the set of lower-bounded nonbasics and U the set of upper-bounded nonbasics. Except for the more complicated right-hand side, it is no more difficult to solve than that of the ordinary simplex.

EXAMPLE 5.28: COMPUTING LOWER- AND UPPER-BOUNDED BASIC SOLUTIONS

Consider the linear program

$$\begin{aligned} \min \quad & -2x_1 + 5x_2 + 3x_3 \\ \text{s.t.} \quad & 1x_1 - 1x_2 + 3x_4 = 13 \\ & 2x_1 + 1x_2 + 2x_3 - 2x_4 = 6 \\ & 6 \geq x_1 \geq 4, \quad 10 \geq x_2 \geq -10, \quad 5 \geq x_3 \geq 1, \quad 3 \geq x_4 \geq 2 \end{aligned}$$

Compute the basic solution having x_2 and x_3 basic, x_1 nonbasic lower bounded, and x_4 nonbasic upper bounded.

Solution: Following definition [5.48], nonbasic variables will be fixed:

$$x_1 = \ell_1 = 4 \quad \text{and} \quad x_4 = u_4 = 3$$

We solve for basic components in equations (5.20):

$$\begin{aligned} -1x_2 + 0x_3 &= 13 - (1)(4) - (-3)(3) = 0 \\ +1x_2 + 2x_3 &= 6 - (2)(4) - (-2)(3) = 4 \end{aligned}$$

The unique solution is $x_2 = 0, x_3 = 2$, implying a full basic solution of $\mathbf{x} = (4, 0, 2, 3)$.

Unrestricted Variables with No Bounds

Sometimes models contain unrestricted variables that can take on any value. Examples include inventory levels where backordering is allowed and temperatures.

In such cases $\ell_j = -\infty$ and $u_j = \infty$. With neither finite, the variables cannot be nonbasic under definition [5.48].

Principle 5.49 | Unrestricted variables must be basic in every lower- and upper-bounded basic solution.

Increasing and Decreasing Nonbasic Variable Values

Simplex directions $\Delta \mathbf{x}$ are constructed to be feasible when nonbasic x_j are increased. The j th directional component is $\Delta x_j = +1$.

This logic works fine for lower-bounded nonbasic variables because the only feasible move is an increase. Tests for improvement are as in principle [5.23].

Principle 5.50 | The simplex direction $\Delta \mathbf{x}$ for lower-bounded nonbasic x_j improves in a maximize problem if $\bar{c}_j > 0$, and in a minimize problem if $\bar{c}_j < 0$.

The new element in lower- and upper-bounded simplex comprises upper-bounded nonbasic variables x_j . If they are to change values and stay feasible, they must decrease (i.e., the j th directional component of a move must be negative).

A little review of the derivations in Section 4.3 will show that the direction decreasing nonbasic x_j changing no other nonbasic, and maintaining $\mathbf{Ax} = \mathbf{b}$ is precisely the negative of simplex direction increasing x_j . Improvement tests are similarly reversed.

Principle 5.51 | Negative simplex direction $-\Delta \mathbf{x}$ for upper-bounded nonbasic x_j improves in a maximize problem if $\bar{c}_j < 0$, and in a minimize problem if $\bar{c}_j > 0$.

EXAMPLE 5.29: TESTING SIMPLEX DIRECTIONS IN LOWER AND UPPER FORMAT

Return to the linear program of Example 5.28. Compute reduced costs for all nonbasic variables, and determine which could enter to improve the objective value.

Solution: Basis matrix

$$\mathbf{B} = \begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix} \text{ has inverse } \mathbf{B}^{-1} = \begin{pmatrix} -1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Thus, pricing vector

$$\mathbf{v} = \mathbf{c}^B \mathbf{B}^{-1} = (5.3) \begin{pmatrix} -1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} = \left(-\frac{7}{2}, \frac{3}{2} \right)$$

Then, applying computation [5.46] yields

$$\bar{c}_1 = c_1 - \mathbf{v} \cdot \mathbf{a}^{(1)} = -2 - \left(-\frac{7}{2}, \frac{3}{2} \right) \cdot (1, 2) = -\frac{1}{2}$$

$$\bar{c}_4 = c_4 - \mathbf{v} \cdot \mathbf{a}^{(4)} = 0 - \left(-\frac{7}{2}, \frac{3}{2} \right) \cdot (3, -2) = \frac{27}{2}$$

These reduced costs both qualify as improving under conditions [5.50] and [5.51].

Step Size with Increasing and Decreasing Values

Step size rule [5.24] assumes that nonbasic variables always increase and that the lower bound of every basic variable is zero. With a lower- and upper-bounded simplex, many more possibilities exist. If a basic variable is decreasing from value x_k , the maximum feasible decrease is now $(x_k - \ell_k)$. If a basic variable is increasing from value x_k , the maximum feasible increase is $(u_k - x_k)$. Also, the changing nonbasic variable may itself fix λ because it can feasibly increase or decrease only $(u_k - \ell_k)$. These cases lead to a more complex step size rule.

Principle 5.52 With upper and lower bounds, the maximum feasible step size λ in direction $\delta \Delta \mathbf{x}$ ($\delta = \pm 1$) is $\lambda = \min \{ \lambda^-, \lambda^+ \}$, where

$$\lambda^- = \min \left\{ \frac{x_j^{(t)} - \ell_j}{-\delta \Delta x_j} : \delta \Delta x_j < 0 \right\} \quad (+\infty \text{ if none})$$

$$\lambda^+ = \min \left\{ \frac{u_j - x_j^{(t)}}{\delta \Delta x_j} : \delta \Delta x_j > 0 \right\} \quad (+\infty \text{ if none})$$

If $\lambda = \infty$ the problem is unbounded.

EXAMPLE 5.30: DETERMINING STEP SIZE IN LOWER- AND UPPER-BOUNDED SIMPLEX

Return again to the example of Example 5.28. Example 5.29 established that both nonbasic variables qualify to enter. Determine the corresponding stepsize, leaving variable for each.

Solution: Using the results of Example 5.29, basic components of the simplex directions for x_1 are

$$\begin{pmatrix} \Delta x_2 \\ \Delta x_3 \end{pmatrix} = -\mathbf{B}^{-1} \mathbf{a}^{(1)} = -\begin{pmatrix} -1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ -\frac{3}{2} \end{pmatrix}$$

The full direction is $\Delta \mathbf{x} = (1, 1, -\frac{3}{2}, 0)$. With increasing orientation $\delta = +1$, rule [5.52] gives

$$\lambda^- \leftarrow \min \left\{ \frac{2 - 1}{\frac{3}{2}} \right\} = \frac{2}{3}$$

$$\lambda^+ \leftarrow \min \left\{ \frac{6 - 4}{1}, \frac{10 - 0}{1} \right\} = 2$$

$$\lambda \leftarrow \min \left\{ \frac{2}{3}, 2 \right\} = \frac{2}{3}$$

The leaving variable, which set the value of λ , is the basic x_3 . Basic components of the simplex directions for x_4 are

$$\begin{pmatrix} \Delta x_2 \\ \Delta x_3 \end{pmatrix} = -\mathbf{B}^{-1} \mathbf{a}^{(4)} = -\begin{pmatrix} -1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 3 \\ -2 \end{pmatrix} = \begin{pmatrix} 3 \\ -\frac{1}{2} \end{pmatrix}$$

making the full direction $\Delta \mathbf{x} = (0, 3, -\frac{1}{2}, 1)$. With decreasing orientation $\delta = -1$, rule 5.52 gives

$$\begin{aligned}\lambda^- &\leftarrow \min \left\{ \frac{0 - (-10)}{3}, \frac{3 - 2}{1} \right\} = 1 \\ \lambda^+ &\leftarrow \min \left\{ \frac{5 - 2}{\frac{1}{2}} \right\} = 6 \\ \lambda &\leftarrow \min \{1, 6\} = 1\end{aligned}$$

This time the leaving variable is the nonbasic x_4 itself.

Case with No Basis Change

The third possibility above, where λ is established by the changing nonbasic itself, presents another new issue. We have no leaving basic variable.

Principle 5.53 If a changing nonbasic establishes step size λ , it merely switches from lower-bounded to upper-bounded status, or vice versa. We need not change the basis or the associated representation of \mathbf{B}^{-1} .

Lower- and Upper-Bounded Simplex Algorithm

We are now ready to state the full lower- and upper-bounded revised simplex Algorithm 5D employed in most commercial computer codes. As in earlier sections, $\mathbf{a}^{(j)}$ denotes the (lower- and upper-bounded) standard-form constraint column for x_j , and $\mathbf{c}^B \triangleq (c_{1st}, c_{2nd}, \dots, c_{mth})$ represents the vector of basic variable objective function coefficients.

Lower- and Upper-Bounded Simplex on Top Brass Application

Application Table 5.10 returns one last time to the Top Brass Trophy application of Figure 5.5. It details the lower- and upper-bounded simplex computations paralleling those in Tables 5.2 and 5.9.

With lower and upper bounds, only 2 basic variables must be chosen. We select the two slacks and make both x_1 and x_2 nonbasic lower bounded. This produces starting basic feasible solution $\mathbf{x}^{(0)} = (0, 0, 1750, 4800)$ with objective function value = 0. The corresponding basis inverse \mathbf{B}^{-1} and pricing vector \mathbf{v} are shown in Table 5.10.

At iteration $t = 0$, both nonbasic lower-bounded variables have positive reduced costs for this \mathbf{v} . We choose the one for x_1 , making $p = 1$ and orientation $\delta = +1$ to indicate increase.

Next we compute step size λ by rules 5.52. Each unit step in that direction decreases x_5 by 1 and x_6 by 4, so that

$$\lambda^- \leftarrow \min \left\{ \frac{1750 - 0}{1}, \frac{4800 - 0}{4} \right\} = 1200$$

The move also increases x_1 by 1 per unit step, making

$$\lambda^+ \leftarrow \min \left\{ \frac{1000 - 0}{1} \right\} = 1000$$

The maximum step feasible for both sorts of change is

$$\lambda = \min [\lambda^-, \lambda^+] = \min \{1200, 1000\} = 1000$$

with $r = 1$.

A step of size λ in the chosen simplex direction moves us to new solution $\mathbf{x}^{(1)} = (1000, 0, 750, 800)$ at objective value 12,000. Since it was the changing nonbasic x_1 that fixed λ (i.e., $p = r$), no modification of the basis is required (principle [5.41]). The variable x_1 merely switches to nonbasic upper bounded.

Iteration $t = 1$ proceeds in much the same way except that this time we consider both decreasing nonbasic x_1 and increasing x_2 . Only the second satisfies tests for improvement, so the $\Delta \mathbf{x}$ for x_2 is pursued until the basic x_6 drops to $\ell_6 = 0$ at step $\lambda = 400$. The new solution is $\mathbf{x}^{(2)} = (1000, 400, 350, 0)$, with objective value 15,600.

ALGORITHM 5D: LOWER- AND UPPER-BOUNDED REVISED SIMPLEX

Step 0: Initialization. Choose any starting feasible basis with initial nonbasic $j \in L$ at ℓ_j and nonbasic $j \in U$ at u_j , and construct a representation of the corresponding basis inverse \mathbf{B}^{-1} . Then use the representation of \mathbf{B}^{-1} to solve

$$\sum_{j \in B} a_{ij}x_j^{(0)} = b_i - \sum_{j \in L} a_{ij}\ell_j - \sum_{j \in U} a_{ij}u_j \quad \text{for all } i = 1, \dots, m$$

for basic components $j \in B$ of initial solution $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Pricing. Use the representation of the current basic column matrix inverse \mathbf{B}^{-1} to solve $\mathbf{v}\mathbf{B} = \mathbf{c}^B$ for pricing vector \mathbf{v} , where \mathbf{c}^B is the vector of basic objective function coefficients. Then evaluate reduced costs $\bar{c}_j \leftarrow c_j - \mathbf{v} \cdot \mathbf{a}^{(j)}$ for each nonbasic x_j .

Step 2: Optimality. If no \bar{c}_j indicates improvement (rules [5.50] and [5.51]), stop; current solution $\mathbf{x}^{(t)}$ is globally optimal. Otherwise, choose an entering nonbasic variable x_p with improving \bar{c}_p and set orientation $\delta = +1$ if the nonbasic is lower bounded and $\delta = -1$ if the nonbasic is upper bounded.

Step 3: Simplex Direction. Construct the simplex direction $\Delta \mathbf{x}^{(t+1)}$ for nonbasic variable x_p using the representation of current basic column inverse \mathbf{B}^{-1} to solve $\mathbf{B}\Delta \mathbf{x} = -\delta \mathbf{a}^{(p)}$ for basic components.

Step 4: Step Size. Apply rule [5.52] to determine the maximum feasible step λ in direction $\Delta \mathbf{x}^{(t+1)}$. If there is no limit ($\lambda = \infty$), stop; the given model is unbounded. Otherwise, choose as leaving x_r any blocking variable that established the value of λ in [5.52].

Step 5: New Point and Basis. Compute the new solution

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \delta \Delta \mathbf{x}^{(t+1)}$$

If $p \neq r$, also replace x_r in the basis by x_p , construct the associated pivot matrix \mathbf{E} , and update the basis inverse representation as $\mathbf{E}\mathbf{B}^{-1}$. Then advance $t \leftarrow t + 1$, and return to Step 1.

TABLE 5.10 Upper- and Lower-Bounded Simplex Search of Top Brass Trophy Application

	x_1	x_2	x_5	x_6	
$\max \mathbf{c}$	12	9	0	0	\mathbf{b}
	1	1	1	0	1750
	4	2	0	1	4800
ℓ	0	0	0	0	
\mathbf{u}	1000	1500	—	—	
$t = 0$	L	L	1st	2nd	
$\mathbf{x}^{(0)}$	0	0	1750	4800	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 0$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$					
\bar{c}_j	12	9	0	0	
$\Delta \mathbf{x}$ for x_1	1	0	-1	-4	
	1000 1	—	$\frac{1750}{1}$	$\frac{4800}{4}$	$\lambda = 1000$
$t = 1$	U	L	1st	2nd	
$\mathbf{x}^{(1)}$	1000	0	750	800	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 12,000$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$					
\bar{c}_j	12	9	0	0	
$\Delta \mathbf{x}$ for x_2	0	1	-1	-4	
	—	$\frac{1500}{1}$	$\frac{750}{1}$	800 2	$\lambda = 400$
$t = 2$	U	2nd	1st	L	
$\mathbf{x}^{(2)}$	1000	400	350	0	$\mathbf{c} \cdot \mathbf{x}^{(2)} = 15,600$
$\mathbf{B}^{-1} = \begin{pmatrix} 1 & -0.5 \\ 0 & 0.5 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} 0 \\ 4.5 \end{pmatrix}$					
\bar{c}_j	-6	0	0	-4.5	
$\Delta \mathbf{x}$ for x_1	-1	2	-1	0	
	$\frac{1000}{1}$	$\frac{1100}{2}$	350 1	—	$\lambda = 350$
$t = 3$	1st	2nd	L	L	
$\mathbf{x}^{(3)}$	650	1100	0	0	$\mathbf{c} \cdot \mathbf{x}^{(3)} = 17,700$
$\mathbf{B}^{-1} = \begin{pmatrix} -1 & 0.5 \\ 2 & -0.5 \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} 6 \\ 1.5 \end{pmatrix}$					
\bar{c}_j	0	0	-6	-1.5	“optimal”

At iteration $t = 2$ we encounter a nonbasic decreasing from its upper bound. With $\bar{c}_1 = -6 < 0$, the negative simplex direction for x_2 helps our maximize objective (5.51). Orientation indicator $\delta = -1$. We step distance $\lambda = 350$ in direction $\delta \Delta \mathbf{x}$ to reach new point $\mathbf{x}^{(3)} = (650, 1100, 0, 0)$ at value of 17,700. Iteration $t = 3$ completes the computation, verifying that no available simplex move can improve the objective function.

EXERCISES**5-1** Consider the linear constraints

$$-w_1 + w_2 \leq 1$$

$$w_2 \leq 3$$

$$w_1, w_2 \geq 0$$

- (a) Sketch the feasible space in a 2-dimensional plot.
- ✓ (b) Determine geometrically whether each of the following solutions are infeasible, boundary, extreme, and/or interior: $\mathbf{w}^{(1)} = (2, 3)$, $\mathbf{w}^{(2)} = (0, 3)$, $\mathbf{w}^{(3)} = (2, 1)$, $\mathbf{w}^{(4)} = (3, 3)$, and $\mathbf{w}^{(5)} = (2, 4)$.
- ✓ (c) For those points of part (b) that are feasible, demonstrate algebraically whether they are boundary or interior.
- ✓ (d) Determine for each of the points in part (b) whether a suitable (nonconstant) objective function could make the point optimal or uniquely optimal. Explain.
- (e) Determine which of the points correspond to basic solutions, and when they do, identify the defining active constraints.

5-2 Do Exercise 5-1 for the LP

$$3w_1 + 5w_2 \leq 15$$

$$5w_1 + 3w_2 \leq 15$$

$$w_1, w_2 \geq 0$$

and points $\mathbf{w}^{(1)} = (0, 0)$, $\mathbf{w}^{(2)} = (1, 1)$, $\mathbf{w}^{(3)} = (2, 0)$, $\mathbf{w}^{(4)} = (3, 3)$ and $\mathbf{w}^{(5)} = (5, 0)$.

5-3 Place each of the following LPs in standard form and identify the corresponding **A**, **b**, and **c** of definition [5.4](#).

✓ (a) $\min \quad 4x_1 + 2x_2 - 33x_3$
s.t. $x_1 - 4x_3 + x_3 \leq 12$

$$9x_1 + 6x_3 = 15$$

$$-5x_1 + 9x_2 \geq 3$$

$$x_1, x_2, x_3 \geq 0$$

(b) $\max \quad 45x_1 + 15x_3$
s.t. $4x_1 - 2x_2 + 9x_3 \geq 22$

$$-2x_1 + 5x_2 - x_3 = 1$$

$$x_1 - x_2 \leq 5$$

$$x_1, x_2, x_3 \geq 0$$

✓ (c) $\max \quad 15(x_1 + 2x_2) + 11(x_2 - x_3)$
s.t. $3x_1 \geq x_1 + x_2 + x_3$
 $0 \leq x_j \leq 3 \quad j = 1, \dots, 3$

(d) $\max \quad -53x_1 + 33(x_1 + 3x_3)$
s.t. $x_j + 1 \leq x_{j+1} \quad j = 1, 2$

$$\sum_{j=1}^3 x_j = 12$$

$$x_j \geq 0 \quad j = 1, \dots, 3$$

✓ (e) $\min \quad 2x_1 + x_2 - 4x_3$
s.t. $x_1 - x_2 - 5x_3 \leq 10$
 $3x_2 + 9x_3 = -6$
 $x_1 \geq 0, x_3 \leq 0$

(f) $\min \quad 4x_1 - x_2$
s.t. $-4x_1 - x_2 + 7x_3 = 9$
 $-x_1 - x_2 + 3x_3 \leq 14$
 $x_2 \leq 0, x_3 \geq 0$

5-4 Consider the linear constraints

$$-y_1 + y_2 \leq 2$$

$$5y_1 \leq 10$$

$$y_1, y_2 \geq 0$$

- (a) Sketch the feasible set in a 2-dimensional plot.
- ✓ (b) Add slacks y_3 and y_4 to place constraints in LP standard form.
- ✓ (c) Determine whether columns of standard form corresponding to each of the following sets of variables form a basis: $\{y_1, y_2\}$, $\{y_2, y_3\}$, $\{y_3, y_4\}$, $\{y_1, y_4\}$, $\{y_3\}$, $\{y_1, y_2, y_4\}$.
- ✓ (d) For each set that does form a basis in part (c), determine the corresponding basic solution and classify it as feasible or infeasible.
- ✓ (e) Identify each solution of part (d) on your plot of part (a), and comment on the connection between basic feasible solutions and extreme points.

5-5 Do Exercise 5-4 for the LP

$$y_1 + 2y_2 \leq 6$$

$$y_2 \leq 2$$

$$y_1, y_2 \geq 0$$

and possible basic sets $\{y_1, y_2\}, \{y_2, y_3\}, \{y_1\}, \{y_2, y_4\}, \{y_2, y_3, y_4\}, \{y_1, y_3\}$.

5-6 Write all conditions that a feasible direction $\Delta \mathbf{w}$ must satisfy, at the solution \mathbf{w} indicated, to each of the following standard-form systems of LP constraints.

- ✓ (a) $5w_1 + 1w_2 - 1w_3 = 9$ at $\mathbf{w} = (2, 0, 1)$
 $3w_1 - 4w_2 + 8w_3 = 14$
 $w_1, w_2, w_3 \geq 0$
- (b) $4w_1 - 2w_2 + 5w_3 = 34$ at $\mathbf{w} = (1, 0, 6)$
 $4w_1 + 2w_2 - 3w_3 = -14$
 $w_1, w_2, w_3 \geq 0$

5-7 Following is a maximizing, standard-form linear program and a classification of variables as basic and nonbasic.

	x_1	x_2	x_3	x_4	
max \mathbf{c}	10	1	0	0	\mathbf{b}
	-1	1	4	21	13
	2	6	0	-2	2
	B	N	B	B	

- ✓ (a) Compute the current basic solution.
- ✓ (b) Compute all simplex directions available at the current basis.
- ✓ (c) Verify that all your simplex directions are feasible at the current basic solution.
- ✓ (d) Determine whether each of the simplex directions is improving.
- ✓ (e) Regardless of whether it improves, determine the maximum step that preserves feasibility in each simplex direction and the new basis that would result after such a step.

5-8 Do Exercise 5-7 for minimizing the standard form LP

	x_1	x_2	x_3	x_4	
min \mathbf{c}	8	-5	0	1	\mathbf{b}
	13	2	3	1	7
	-4	1	0	-1	-1
	N	N	B	B	

5-9 Consider the linear program

$$\begin{aligned} \max \quad & 3z_1 + z_2 \\ \text{s.t.} \quad & -2z_1 + z_2 \leq 2 \\ & z_1 + z_2 \leq 6 \\ & z_1 \leq 4 \\ & z_1, z_2 \geq 0 \end{aligned}$$

- ✓ (a) Solve the problem graphically.
- ✓ (b) Add slacks $z_3, z_4,$ and z_5 to place the model in standard form.
- ✓ (c) Apply rudimentary simplex Algorithm 5A to compute an optimal solution to your standard form starting with all slacks basic.
- ✓ (d) Plot your progress in part (c) on the graph of part (a).

5-10 Do Exercise 5-9 for the LP

$$\begin{aligned} \max \quad & 2z_1 + 5z_2 \\ \text{s.t.} \quad & 3z_1 + 2z_2 \leq 18 \\ & z_1 \leq 5 \\ & z_2 \leq 3 \\ & z_1, z_2 \geq 0 \end{aligned}$$

5-11 Consider the linear program

$$\begin{aligned} \max \quad & 10y_1 + y_2 \\ \text{s.t.} \quad & 3y_1 + 2y_2 \geq 6 \\ & 2y_1 + 4y_2 \leq 8 \\ & y_1, y_2 \geq 0 \end{aligned}$$

- (a) Solve the problem graphically. Be sure to identify all constraints, show contours of the objective, outline the feasible space, and justify that an optimal solution is $y_1^* = 4, y_2^* = 0$.
- (b) Place the above LP in standard form using nonnegative slack variables y_3 and y_4 .
- (c) Explain why a main constraint in the original model will be active at a given solution exactly when the corresponding solution in standard form has its slack's nonnegative constraint active at value = 0.

5-12 Consider the standard-form linear program

$$\begin{aligned} \min \quad & x_2 + x_4 + x_5 \\ \text{s.t.} \quad & -2x_1 + x_2 + 2x_4 = 7 \\ & x_4 + x_5 = 5 \\ & x_2 + x_3 - x_4 = 3 \\ & x_1, \dots, x_5 \geq 0 \end{aligned}$$

- (a) Compute the basic solution corresponding to x_1, x_3, x_4 basic, and explain why it provides an appropriate place for the rudimentary simplex Algorithm 5A to begin its search.
- (b) Starting from the basis of part (a), apply Algorithm 5A to compute an optimal solution to the given LP.

5-13 Do Exercise 5-12 on standard-form linear program

$$\begin{aligned} \max \quad & 5x_1 - 10x_2 \\ \text{s.t.} \quad & 1x_1 - 1x_2 + 2x_3 + 4x_5 = 2 \\ & 1x_1 + 1x_2 + 2x_4 + x_5 = 8 \\ & x_1, \dots, x_5 \geq 0 \end{aligned}$$

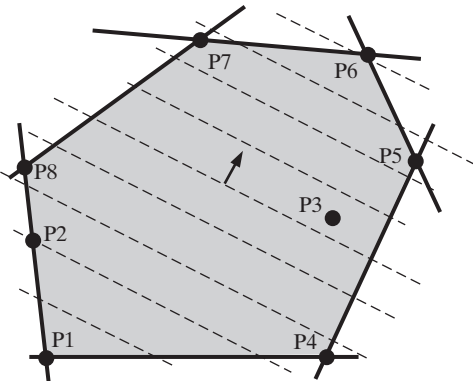
starting with x_3 and x_4 basic.

5-14 Do Exercise 5-12 on standard-form linear program

$$\begin{aligned} \min \quad & 2x_1 + 4x_2 + 6x_3 + 10x_4 + 7x_5 \\ \text{s.t.} \quad & x_1 + x_4 = 6 \\ & x_2 + x_3 - x_4 + 2x_5 = 9 \\ & x_1, \dots, x_5 \geq 0 \end{aligned}$$

starting with x_1 and x_2 basic.

5-15 The following plot shows several feasible points in a linear program and contours of its objective function.



Determine whether each of the following sequences of solutions could have been one followed by the simplex algorithm applied to the corresponding LP standard form.

- ✓ (a) P1,P8,P7,P6
- ✓ (b) P2,P8,P7,P6
- ✓ (c) P1,P3,P6
- (d) P1,P4,P5,P6,P7

- ✓ (e) P1,P7,P6
- (f) P4,P1,P8,P7,P6

5-16 Construct the simplex dictionary form 5.28 corresponding to each of the following.

- ✓ (a) The model and basis shown in Exercise 5-7
- (b) The model and basis shown in Exercise 5-8

5-17 Rudimentary simplex Algorithm 5A is being applied to optimize a linear program with objective function

$$\min \quad 3w_1 + 11w_2 - 8w_3$$

Determine whether each of the following simplex directions for w_4 leads to a conclusion that the given LP is unbounded.

- ✓ (a) $\Delta \mathbf{w} = (1, 0, -4, 1)$
- (b) $\Delta \mathbf{w} = (1, 3, 10, 1)$
- ✓ (c) $\Delta \mathbf{w} = (1, 3, 0, 1)$
- (d) $\Delta \mathbf{w} = (0, 1, -2, 1)$

5-18 Consider the linear program

$$\begin{aligned} \max \quad & 4y_1 + 5y_2 \\ \text{s.t.} \quad & -y_1 + y_2 \leq 4 \\ & y_1 - y_2 \leq 10 \\ & y_1, y_2 \geq 0 \end{aligned}$$

- (a) Show graphically that the model is unbounded.
- ✓ (b) Add slacks y_3 and y_4 to place the model in standard form.
- ✓ (c) Starting with all slacks basic, apply rudimentary simplex Algorithm 5A to establish that the original model is unbounded.

5-19 Do Exercise 5-18 for the LP

$$\begin{aligned} \min \quad & -10y_1 + y_2 \\ \text{s.t.} \quad & -5y_1 + 3y_2 \leq 15 \\ & 3y_1 - 5y_2 \leq 8 \\ & y_1, y_2 \geq 0 \end{aligned}$$

5-20 Setup each of the following to begin Phase I of two-phase simplex Algorithm 5B. Also indicate the basic variables of the initial Phase I solution.

- ✓ (a) $\max \quad 2w_1 + w_2 + 9w_3$
 $\text{s.t.} \quad w_1 + w_2 \leq 18$
 $-2w_1 + w_3 = -2$
 $3w_2 + 5w_3 \geq 15$
 $w_1, w_2, w_3 \geq 0$

$$\begin{aligned}
 \text{(b) max } & 5w_1 + 18w_2 \\
 \text{s.t. } & 2w_1 + 4w_2 = 128 \\
 & 7w_1 + w_2 \geq 11 \\
 & 6w_1 + 16w_2 \geq 39 \\
 & w_1 + 3w_2 \leq 222 \\
 & w_1, w_2 \geq 0
 \end{aligned}$$

5-21 Setup each of the models in Exercise 5-20 to begin a big- M solution using rudimentary simplex Algorithm 5A. Also indicate the basic variables of the initial solution.

5-22 Consider the linear program

$$\begin{aligned}
 \text{max } & 9y_1 + y_2 \\
 \text{s.t. } & -2y_1 + y_2 \geq 2 \\
 & y_2 \leq 1 \\
 & y_1, y_2 \geq 0
 \end{aligned}$$

(a) Show graphically that the model is infeasible.

- ✓ (b) Add slacks and artificials y_3, \dots, y_5 to setup the model for Phase I of Algorithm 5B.
- ✓ (c) Apply rudimentary simplex Algorithm 5A to this Phase I problem to establish that the original model is infeasible.

5-23 Do Exercise 5-22 for the linear program

$$\begin{aligned}
 \text{min } & 2y_1 + 8y_2 \\
 \text{s.t. } & y_1 + y_2 \leq 5 \\
 & y_2 \geq 6 \\
 & y_1, y_2 \geq 0
 \end{aligned}$$

5-24 Assuming that step size $\lambda > 0$ at every step, compute a finite bound on the number of iterations of Algorithm 5A for each of the following standard-form linear programs.

- ✓ (a) The model in Exercise 5-7
- (b) The model in Exercise 5-8
- ✓ (c) A model with 1150 main constraints and 2340 variables
- (d) A model with 211 main constraints and 7200 variables

5-25 Rudimentary simplex Algorithm 5A is being applied to a standard-form linear program with variables x_1, \dots, x_5 . Determine whether each of the following basic solutions is degenerate for the given basic variable set.

- ✓ (a) $B = \{x_1, x_2, x_3\}$, $\mathbf{x} = (1, 0, 5, 0, 0)$
- (b) $B = \{x_3, x_4, x_5\}$, $\mathbf{x} = (0, 0, 1, 0, 9)$
- ✓ (c) $B = \{x_1, x_3, x_5\}$, $\mathbf{x} = (1, 0, 5, 0, 8)$
- (d) $B = \{x_1, x_2, x_4\}$, $\mathbf{x} = (0, 0, 2, 0, 1)$

5-26 Return to Exercise 5-4 and consider adding additional constraint $y_2 \leq 4$ to the original LP.

- (a) Repeat parts (a)-(c) of Exercise 5-4 with the extra constraint, and additional slack y_5 included in all potential basis sets of part (c).
- (b) Demonstrate that in your revised standard form LP has 3 feasible bases corresponding to extreme-point $(y_1, y_2) = (2, 4)$.
- (c) Explain how the condition in (b) leads to degeneracy, and show a feasible simplex direction relative to one of the basic solutions that produces a $\lambda = 0$ step to another one.

5-27 Do Exercise 5-26 on the LP of Exercise 5-5 with additional constraint $y_1 \leq 6$ and focusing degeneracy parts on extreme point $(y_1, y_2) = (6, 0)$.

5-28 Consider the linear program

$$\begin{aligned}
 \text{max } & x_1 + x_2 \\
 \text{s.t. } & x_1 + x_2 \leq 9 \\
 & -2x_1 + x_2 \leq 0 \\
 & x_1 - 2x_2 \leq 0 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

- ✓ (a) Solve the problem graphically.
- ✓ (b) Add slacks x_3, \dots, x_5 to place the model in standard form.
- ✓ (c) Apply rudimentary simplex Algorithm 5A to compute an optimal solution to your standard form starting with all slacks basic.
- ✓ (d) Plot your progress in part (c) on the graph of part (a).
- (e) How can the algorithm be making progress when $\lambda = 0$ if some moves of part (c) left the solution unchanged? Explain.

5-29 Do Exercise 5-28 for the LP

$$\begin{aligned}
 \text{max } & x_1 \\
 \text{s.t. } & 6x_1 + 3x_2 \leq 18 \\
 & 12x_1 - 3x_2 \leq 0 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

5-30 Return to the LP of Exercise 5-7.

- ✓ (a) Compute the basis matrix inverse corresponding to the basic variables indicated.
- ✓ (b) Compute the corresponding pricing vector of [5.45].
- ✓ (c) Without generating the implied simplex directions, use your pricing vector to determine whether each of them will be improving.
- ✓ (d) For each improving simplex direction in part (c), generate the corresponding matrix **E** of [5.43], and compute the next basis matrix inverse.

5-31 Do Exercise 5-30 for the LP of Exercise 5-8.

5-32 Consider applying revised simplex Algorithm 5C to the tabulated standard-form LP

	x_1	x_2	x_3	x_4	x_5	
min $c =$	5	4	3	2	16	b
A =	2	0	1	0	6	8
	0	1	1	2	3	12

starting with x_1 and x_2 basic.

- (a) Determine the corresponding starting basis matrix and its inverse, along with the associated primal basic solution and pricing vector.
 - (b) Use appropriate column pricing to decide which nonbasics are eligible to enter the basis, pick the one with most negative \bar{c}_j , and generate its simplex direction.
 - (c) Complete the rest of one iteration by (i) choosing a step size and leaving basic variable, (ii) updating the current basis matrix, (iii) updating the corresponding basis inverse with an appropriate E-matrix, and (iv) updating the primal solution and the pricing vector.
- 5-33** Solve each of the following standard form linear programs by revised simplex Algorithm 5C, showing the basic inverse, the pricing vector, and update matrix **E** used at each iteration. Start from the basis specified in each original exercise.
- ✓ (a) The LP of Exercise 5-12.
 - (b) The LP of Exercise 5-13.
 - (c) The LP of Exercise 5-14.

5-34 Suppose lower- and upper-bounded simplex Algorithm 5D is being applied to a problem with objective function

$$\max 3x_1 - 4x_2 + x_3 - 4x_4 + 10x_5$$

3 main constraints, and bounds

$$0 \leq x_j \leq 5 \quad j = 1, \dots, 5$$

For each of the following current basic solutions **x** and corresponding simplex directions $\Delta \mathbf{x}$, determine whether the appropriate move of $\pm \Delta \mathbf{x}$ is improving. Also compute the maximum step λ that could be applied without losing feasibility and the basis status of variables that would result after such a step. Take the current basic variables to be those strictly between lower and upper bounds.

- ✓ (a) $\mathbf{x} = (2, 2, 4, 0, 5)$, $\Delta \mathbf{x} = (1, -1, 0, 0, 1)$ for x_5
- (b) $\mathbf{x} = (5, 0, 2, 3, 2)$,
 $\Delta \mathbf{x} = \left(0, 1, \frac{1}{10}, -\frac{1}{5}, \frac{1}{3}\right)$ for x_2
- ✓ (c) $\mathbf{x} = (0, 1, 0, 4, 2)$, $\Delta \mathbf{x} = \left(0, 0, 1, -\frac{2}{5}, \frac{2}{5}\right)$ for x_3
- (d) $\mathbf{x} = (5, 5, 1, 3, 1)$, $\Delta \mathbf{x} = (1, 0, 0, 4, 1)$ for x_1

5-35 Consider the linear program

$$\begin{aligned} \min \quad & 5z_1 + 6z_2 \\ \text{s.t.} \quad & z_1 + z_2 \geq 3 \\ & 3z_1 + 2z_2 \geq 8 \\ & 0 \leq z_1 \leq 6 \\ & 0 \leq z_2 \leq 5 \end{aligned}$$

- ✓ (a) Solve the problem graphically.
- ✓ (b) Add slacks z_3 and z_4 to place the model in standard form for a lower- and upper-bounded simplex.
- ✓ (c) Apply lower- and upper-bounded simplex Algorithm 5D to compute an optimal solution to your standard form starting with all slacks basic and original variables nonbasic at their upper bounds.
- ✓ (d) Plot your progress in part (c) on the graph of part (a).

5-36 Do Exercise 5-35 on the LP

$$\begin{aligned} \max \quad & 6z_1 + 8z_2 \\ \text{s.t.} \quad & z_1 + 3z_2 \leq 10 \\ & z_1 + z_2 \leq 5 \\ & 0 \leq z_1 \leq 4 \\ & 0 \leq z_2 \leq 3 \end{aligned}$$

Start with original variable z_1 nonbasic lower-bounded and z_2 upper-bounded.

5-37 Solve each of the following standard form linear programs by lower- and upper-bounded simplex Algorithm 5D, showing the basic inverse, the pricing vector, and update matrix \mathbf{E} used at each iteration.

- (a) The LP of Exercise 5-12 with added upper bounds $x_j \leq 3, j = 1, \dots, 5$, starting with x_1, x_3, x_5 basic and x_2, x_4 nonbasic upper-bounded.
- (b) The LP of Exercise 5-13 with added upper bounds $x_j \leq 2, j = 1, \dots, 5$, starting with x_4, x_5 basic and x_1, x_2 nonbasic upper-bounded.
- (c) The LP of Exercise 5-14 with added upper bounds $x_j \leq 4, j = 1, \dots, 6$, starting with x_4, x_5 basic and x_1, x_2, x_3 nonbasic upper-bounded.

REFERENCES

- Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.
- Bertsimas, Dimitris and John N. Tsitklis (1997), *Introduction to Linear Optimization*. Athena Scientific, Nashua, New Hampshire.
- Chvátal, Vašek (1980), *Linear Programming*, W.H. Freeman, San Francisco, California.
- Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009). *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.
- Luenberger, David G. and Yinyu Ye (2008), *Linear and Nonlinear Programming*, Springer, New York, New York.

This page intentionally left blank

Duality, Sensitivity, and Optimality in Linear Programming

With the simplex and interior point methods of Chapters 5, 6, and 7 we can compute mathematically optimal solutions to linear programming models. By comparison to nonlinear, discrete, and other more difficult optimization forms, this is no small achievement. Still, it rarely fulfills all an analyst's needs.

Mathematical optima do not suffice because the constant parameters from which they are derived—such items as costs, profits, yields, supplies, and demands—are almost never known with certainty at the time the model is solved. Often, they are not even within a factor of 10.

How much can we trust mathematically optimal answers to very imperfectly parameterized models? Maybe a cost or demand controls everything. Maybe it could change dramatically with absolutely no impact.

These are exactly the sorts of questions that **sensitivity analysis** tries to address. In Section 1.3 we explained that constants of OR models are really **input parameters**—values we agree to take as fixed at the system boundary in order to produce a tractable model. Optimal solutions simply provide a best choice of decision variables for one fixing of the inputs. Sensitivity analysis then tries to complete the picture by studying how results would vary with changes in parameter values.

Since linear programs have proved the most tractable of mathematical programs to numerical search, it should not surprise us that they also admit powerful sensitivity analysis. In this chapter we will see how post-optimality analyses of LP models can exploit the by-products of our search for an optimum to illuminate how the solution might change with variations in input variables. Remarkably in fact, we will see that there is an entirely separate **dual** linear program, defined on the same input constants as the main **primal** model, that has optimal solutions replete with sensitivity insights.

6.1 GENERIC ACTIVITIES VERSUS RESOURCES PERSPECTIVE

We have encountered linear programming models drawn from a wide range of application settings. How can we speak about the sensitivity of their optimal solutions to changes in input parameters when they model such different things?

We need a generic perspective—a standard intuition about the variables, constraints, and objective functions of LP models. Then, although the details will certainly change with the model, we can still speak in broad, common terms about the meaning of sensitivity results.

Objective Functions as Costs and Benefits

The easiest part of optimization models to interpret generically is their objective functions. Although the exact meaning varies greatly, virtually every model we will encounter can be thought about in terms of **costs** and **benefits**.

Principle 6.1 Optimization model objective functions usually can be interpreted as minimizing some measure of cost or maximizing some measure of benefit.

Choosing a Direction for Inequality Constraints

Now consider inequality constraints. What does a \geq inequality sign typically signify? Is a \leq sign any different?

Every beginning algebra student knows that there is no mathematical distinction. For example, the gasoline demand constraint of the Two Crude refining model (2.3) can be written equivalently as either

$$0.3x_1 + 0.4x_2 \geq 2.0 \quad \text{or} \quad -0.3x_1 - 0.4x_2 \leq -2.0$$

Still, the first certainly depicts more clearly the idea that output must meet or exceed demand.

Most constraints have a “natural” format of this sort. No absolute principle tells us which direction is most intuitive, but a rule of thumb covers the foregoing and most other examples.

Principle 6.2 The most natural expression of a constraint is usually the one making the right-hand-side constant nonnegative.

Inequalities as Resource Supplies and Demands

At the level of intuitive understanding, we see that \leq inequalities do differ from \geq inequalities. To assign a generic meaning, let us review some of the many \leq inequalities that we have encountered:

- $x_1 \leq 9$ restricts the supply of Saudi petroleum in the Two Crude model (2.6) in Section 2.1.
- $4x_1 + 2x_2 \leq 4800$ in model (5.1) (Section 5.1) limits the wood supply in the Top Brass Trophy example.

- $0.25(z_{1/4,A,B} + z_{1/4,A,C} + z_{1/4,B,C}) + 0.40(z_{1/2,A,B} + z_{1/2,A,C} + z_{1/2,B,C}) \leq 4500$ enforces the supply limit on pressing capacity in the CFPL model of Table 4.5 in Section 4.3.
- $y_{11} + y_{12} \leq 20$ fixes the availability of overtime in the ONB example of Table 4.7 in Section 4.4.

Common elements are a resource and a limited supply of that resource.

Principle 6.3 Optimization model constraints of the \leq form usually can be interpreted as restricting the **supply** of some commodity or resource.

The corresponding interpretation of \geq inequalities is as output demands.

- $0.4x_1 + 0.2x_2 \geq 1.5$ in model (2.6) (Section 2.1) requires jet fuel output to meet demand in the Two Crude refining example.
- $0.120x_1 + 0.011x_2 + 1.0x_6 \geq 10$ set the minimum acceptable level of chromium in the blend of the Swedish Steel model (4.4) (Section 4.2).
- $y_{1/8,A,B} + y_{1/8,B,B} + y_{1/8,C,B} - z_{1/2,A,B} - z_{1/2,A,C} - z_{1/2,B,C} \geq 0$ demands production of $\frac{1}{8}$ -inch veneer to exceed consumption in the CFPL example of Table 4.5 (Section 4.3).
- $1y_{12} + 1x_{13} + 0.8z_{20} - w_{21} \geq 8$ enforces the demand that all checks be processed by 22:00 (10 P.M.) in the ONB model of Table 4.7 (Section 4.4).

As with \leq inequalities, the common element is a commodity or resource, but the direction is different.

Principle 6.4 Optimization model constraints of the \geq form usually can be interpreted as requiring satisfaction of a **demand** for some commodity or resource.

Equality Constraints as Both Supplies and Demands

Equality constraints are hybrids. Consider, for example, the Swedish Steel model (4.4) constraint:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000$$

enforcing the requirement that ingredients in the melting furnace charge should have a total weight of exactly 1000 kilograms. This (or any other) equality can just as well be expressed as two opposed inequalities:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 1000$$

and

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \geq 1000$$

Thinking of equalities in this way makes it easy to see the hybrid.

Principle 6.5 Optimization model equality constraints usually can be interpreted as imposing both a supply restriction and a demand requirement on some commodity or resource.

The effect is a mixture of both limits on supply of a resource and demand for its output.

Variable-Type Constraints

Variable-type limitations such as nonnegativities are constraints in a linear program. We can even stretch principle [6.4] to think of nonnegativities as demands for some “positiveness” resource. Still, it usually makes more sense to keep variable-type constraints separate.

Principle 6.6 Nonnegativity and other sign restriction constraints are usually best interpreted as declarations of variable type rather than supply or demand limits on resources.

Variables as Activities

Turning now to the decision variables in an optimization model, let us again recall some examples:

- x_1 in the Two Crude model (2.6) (Section 2.1) chooses the amount of Saudi crude to be refined.
- $x_{p,m}$ in the TP model (4.5) (Section 4.3) decides the amount of product p produced at mill m .
- z_h in the ONB model of Table 4.7 (Section 4.4) sets the number of part-time employees starting at hour h .
- x_j in the Swedish Steel model (4.4) (Section 4.2) establishes the number of kilograms of ingredient j used in the blend.

The common element shared by these and other examples is a sense of **activity**.

Principle 6.7 Decision variables in optimization models can usually be interpreted as choosing the level of some activity.

LHS Coefficients as Activity Inputs and Outputs

Summarizing, our generic linear program chooses activity levels of appropriate sign to minimize cost or maximize benefit, subject to \leq supply limits on input resources, \geq demand requirements for output resources, and $=$ constraints doing both. It follows immediately how we should think about the objective function and constraint coefficients on decision variables.

Principle 6.8 Nonzero objective function and constraint coefficients on LP decision variables display the impacts per unit of the variable’s activity on resources or commodities associated with the objective and constraints.

Sometimes those impacts become clearer in block diagrams such as those of Figure 6.1. Each variable or activity forms a block. Inputs and outputs to the block indicate how a unit of the activity affects the constraints and objective. For example, each unit (thousand barrels) of Saudi petroleum refined in the Two Crude example (2.6) inputs 1 unit (thousand barrels) of Saudi availability and 20 units (thousand

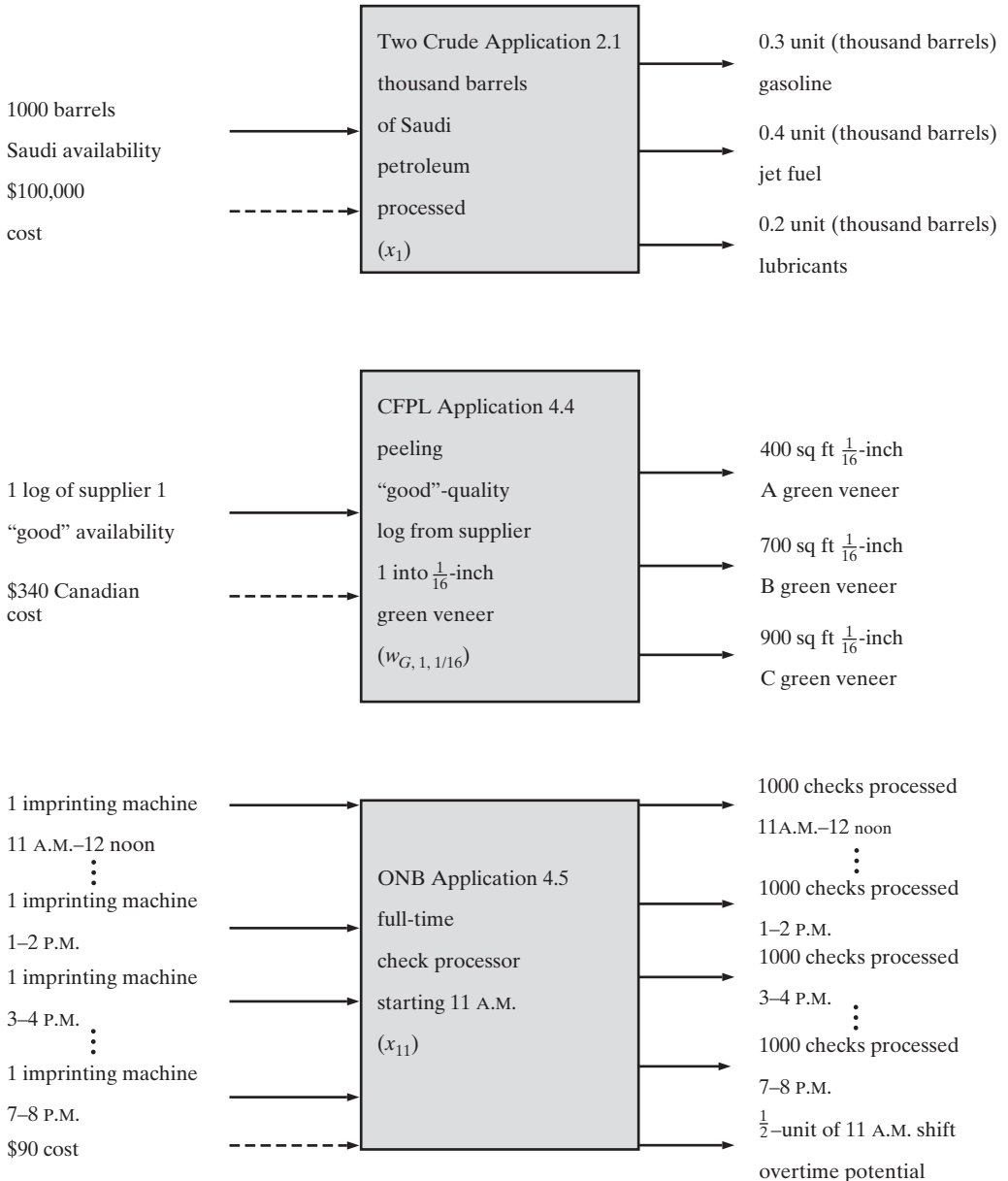


FIGURE 6.1 Inputs and Outputs for Activities in Various Models

dollars) cost to output 0.3 unit (thousand barrels) of gasoline, 0.4 unit (thousand barrels) of jet fuel, and 0.2 unit (thousand barrels) of lubricants.

The second diagram in Figure 6.1 comes from the CFPL model of Table 4.5. Peeling “good”-quality logs from supplier 1 as $\frac{1}{16}$ -inch green veneer consumes log availability and cost to obtain specified amounts of different veneer grades.

Finally, we have an activity from the ONB model of Table 4.7. Using a fulltime employee on the 11 A.M. shift in the ONB example inputs cost and equipment for each on-duty hour to output check processing through the shift and overtime potential. Each input or output produces a nonzero model coefficient.

EXAMPLE 6.1: GENERICALLY INTERPRETING LINEAR PROGRAMS

Give a generic interpretation of the objective function, constraints variables, and coefficients in the following linear program:

$$\begin{array}{rllll} \max & +13x_1 & +24x_2 & +5x_3 & +50x_4 \\ \text{s.t.} & +1x_1 & +3x_2 & & \geq 89 \\ & & & -3x_3 & -5x_4 \leq -60 \\ & +10x_1 & +6x_2 & +8x_3 & +2x_4 \leq 608 \\ & & +1x_2 & & +1x_4 \leq 28 \\ & & & & x_1, \dots, x_4 \geq 0 \end{array}$$

Solution: We begin by reversing the direction of the second inequality to make its right-hand side nonnegative (principle [6.2](#)):

$$\begin{array}{rllll} \max & +13x_1 & +24x_2 & +5x_3 & +50x_4 \\ \text{s.t.} & +1x_1 & +3x_2 & & \geq 89 \\ & & & +3x_3 & +5x_4 \geq 60 \\ & +10x_1 & +6x_2 & +8x_3 & +2x_4 \leq 608 \\ & & +1x_2 & & +1x_4 \leq 28 \\ & & & & x_1, \dots, x_4 \geq 0 \end{array}$$

We may now interpret the model as one of deciding the optimal level of 4 activities corresponding to the 4 decision variables (principle [6.7](#)). The objective function maximizes the benefit derived from these activities (principle [6.1](#)) with coefficients showing the benefit per unit activity (principle [6.8](#)).

The 4 main constraints deal with 4 commodities or resources. The initial 2, being of \geq form, specify a demand of 89 for the first commodity and 60 for the second (principle [6.4](#)). The last 2, being of \leq form, restrict the supply of commodities 3 and 4. Nonnegativity constraints merely enforce the variable type (principle [6.6](#)).

Coefficients on the left-hand side of main constraints show inputs and outputs per unit of each variable’s activity (principle [6.8](#)). Specifically, each unit of activity 1 consumes 10 of commodity 3 to produce 1 of commodity 1; each unit of activity 2 consumes 6 of commodity 3 and 1 of commodity 4 to produce 3 of commodity 1; each unit of activity 3 consumes 8 of commodity 3 to produce 1 of commodity 2; and each unit of activity 4 consumes 2 of commodity 3 and 1 of commodity 4 to produce 5 of commodity 2.

6.2 QUALITATIVE SENSITIVITY TO CHANGES IN MODEL COEFFICIENTS

Armed with a generic way of thinking about LP models, we are ready to consider the sensitivity of optimization model results to changes in their input parameters or constants. Let us begin qualitatively. Much can be learned from looking just at the directions of change rather than their magnitudes.

Relaxing versus Tightening Constraints

Consider first relaxing versus tightening constraints. Figure 6.2 shows the idea graphically. A two-variable model in part (a) has the feasible region indicated by shading. Relaxing a constraint as in part (b) admits new feasible solutions. The corresponding optimal value must stay the same or improve. On the other hand, if the constraint is tightened as in part (c), fewer feasible solutions are available. The optimal value can only stay the same or worsen.

Principle 6.9 | **Relaxing** the constraints of an optimization model either leaves the optimal value unchanged or makes it better (higher for a maximize, lower for a minimize). **Tightening** the constraints either leaves the optimal value unchanged or makes it worse.

Principle [6.9](#) is the most broadly applicable of all sensitivity insights. It holds for any optimization model—LP or not—and any constraint type.

Swedish Steel Application Revisited

For a more concrete illustration, we revisit our Swedish Steel blending example of Section 4.2. The associated LP model is

$$\begin{array}{llll}
 \min & 16x_1 + 10x_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 & & \text{(cost)} \\
 \text{s.t.} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 & = & 1000 \quad \text{(weight)} \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 & \geq & 6.5 \quad \text{(carbon)} \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 & \leq & 7.5 \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 & \geq & 30.0 \quad \text{(nickel)} \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 & \leq & 30.5 & (6.1) \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 & \geq & 10.0 \quad \text{(chromium)} \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 & \leq & 12.0 \\
 & 0.001x_2 + 1.0x_7 & \geq & 11.0 \quad \text{(molybdenum)} \\
 & 0.001x_2 + 1.0x_7 & \leq & 13.0 \\
 & x_1 \leq 75 & & \text{(availability)} \\
 & x_2 \leq 250 & & \\
 & x_1, \dots, x_7 \geq 0 & & \text{(nonnegativity)}
 \end{array}$$

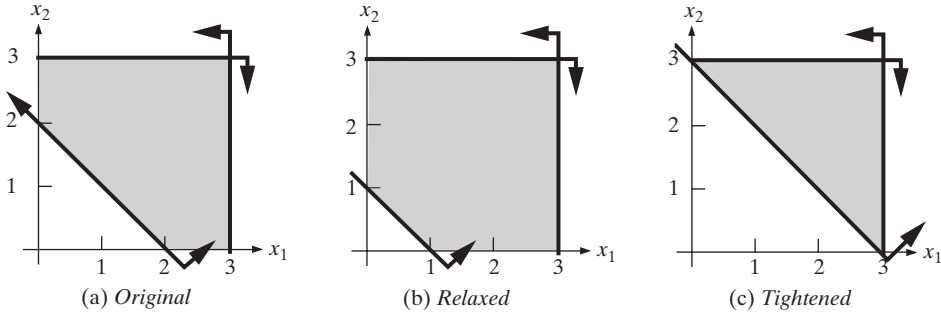


FIGURE 6.2 Effects of Relaxing Versus Tightening Constraints

Effects of Changes in Right-Hand Sides

Figure 6.3 plots the optimal value in the Swedish Steel model (6.1) versus two of its right-hand-side coefficients. Specifically, part (a) follows the impact of the right-hand side (RHS) for the scrap 1 availability constraint

$$x_1 \leq 75 \tag{6.2}$$

assuming that all other parameters are held constant. Similarly, part (b) tracks changes with the right-hand side of the minimum chromium content constraint:

$$0.120x_1 + 0.011x_2 + 1.0x_6 \geq 10.0 \tag{6.3}$$

Infeasible cases of this minimize cost problem are plotted as infinite costs.

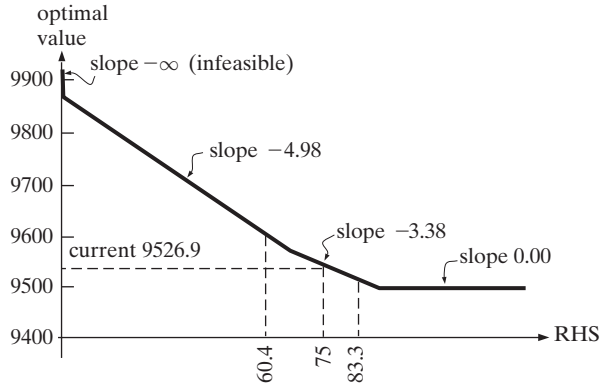
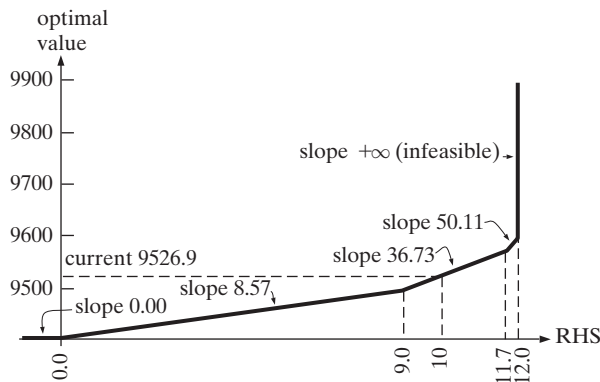
In Section 6.6 we discuss how to generate such plots. For the moment, just notice how the trends differ. Increases in the scrap 1 right-hand side help the minimize cost objective and decreases hurt. Changes in the minimum chromium right-hand side produce exactly the opposite effect.

This apparently contradictory experience follows from one of the constraints being of \leq supply form and the other a \geq demand. An increase in the right-hand side for the two forms produces different effects on the feasible set.

Principle 6.10 Changes in LP model right-hand-side coefficients affect the feasible space as follows:

Constraint Type	RHS Increase	RHS Decrease
Supply (\leq)	Relax	Tighten
Demand (\geq)	Tighten	Relax

The plot in Figure 6.3(a) declines because an increase in the right-hand side relaxes a \leq constraint. Under principle [6.9] the optimal value must stay the same or decrease (improve). Increases in the RHS of part (b) tighten the corresponding \geq constraint. The optimal value must either stay the same or increase (worsen).

(a) Supply (\leq) constraint case (scrap 1 availability)(b) Demand (\geq) constraint case (minimum chromium content)**FIGURE 6.3** Sensitivity of Swedish Steel Optimal Value to Right-Hand Sides**EXAMPLE 6.2: QUALITATIVELY ASSESSING RHS SENSITIVITY**

Determine for each of the following objective function and constraint pairs the qualitative effect of increasing or decreasing the constraint right-hand side. Assume that the given constraint is not the only one.

- | | |
|---------------------------------|---------------------------------|
| (a) max $13w_1 - 11w_2 + w_3$ | (b) max $13w_1 - 11w_2 + w_3$ |
| s.t. $9w_1 + w_2 - w_3 \leq 50$ | s.t. $9w_1 + w_2 - w_3 \geq 50$ |
| (c) min $8z_1 - 4z_2 + 15z_3$ | (d) min $8z_1 - 4z_2 + 15z_3$ |
| s.t. $6z_1 - 3z_2 \leq -19$ | s.t. $6z_1 - 3z_2 \geq -19$ |

Solution: We apply principles [6.9] and [6.10].

(a) A RHS increase relaxes this \leq constraint, meaning that the optimal value will stay the same or increase (improve for a maximize objective). RHS decreases have the opposite effect.

- (b) A RHS increase tightens this \geq constraint, meaning that the optimal value will stay the same or decrease (worsen).
- (c) A RHS increase relaxes this \leq constraint, meaning that the optimal value will stay the same or increase (improve for a minimize objective).
- (d) A RHS increase tightens this \geq constraint, meaning that the optimal value will stay the same or increase (worsen).

Effects of Changes in LHS Constraint Coefficients

With linear programs, principle [6.9](#) holds just as well for coefficient changes on the left-hand side (LHS) of a constraint as for RHS variations such as those in Figure 6.3. Since constraint functions are just weighted sums of the variables, minimum chromium content constraint (6.3) is relaxed if we increase the 0.120 yield coefficient on x_1 to, say, 0.500. It is tightened if, for instance, we reduce it to -0.400 . A larger coefficient on the left-hand side of a \geq constraint makes it easier (for nonnegative variables) to satisfy, and a smaller coefficient makes it harder to satisfy. Other cases are similar.

Principle 6.11 Changes in LP model LHS constraint coefficients on nonnegative decision variables affect the feasible space as follows:

Constraint Type	Coefficient Increase	Coefficient Decrease
Supply (\leq)	Tighten	Relax
Demand (\geq)	Relax	Tighten

EXAMPLE 6.3: QUALITATIVELY ASSESSING LHS SENSITIVITY

Return to the objective functions and constraints of Example 6.2, and determine the qualitative impact of each of the following coefficient changes. Assume that all variables are required to be nonnegative.

- (a) Change the coefficient of w_2 to 6 in Example 6.2(a).
- (b) Change the coefficient on w_3 to 0 in Example 6.2(b).
- (c) Change the coefficient on z_1 to 2 in Example 6.2(c).
- (d) Change the coefficient on z_3 to -1 in Example 6.2(d).

Solution: We apply principles [6.9](#) and [6.11](#).

- (a) The change from 1 to 6 is an increase, meaning that the \leq constraint is tightened and the maximize optimal value will stay the same or decrease.
- (b) The change from -1 to 0 is an increase, meaning that the \geq constraint is relaxed and the maximize optimal value will stay the same or increase.
- (c) The change from 6 to 2 is a decrease, meaning that the \leq constraint is relaxed and the minimize optimal value will stay the same or decrease.
- (d) The change from 0 to -1 is a decrease, meaning that the \geq constraint is tightened and the minimize optimal value will stay the same or increase.

Effects of Adding or Dropping Constraints

Principle 6.9 can even be stretched to cases where we change the model more dramatically by adding or dropping constraints.

Principle 6.12 Adding constraints to an optimization model tightens its feasible set, and dropping constraints relaxes its feasible set.

It follows that adding constraints can only make the optimal value worse. Dropping constraints can only make it better.

Figure 6.3(a) implicitly includes the dropping case. Elimination of the scrap 1 availability constraint is the same thing as making its right-hand side very large. Part (a) of the figure shows how the optimal value declines, an improvement for a minimize cost problem.

EXAMPLE 6.4: QUALITATIVELY ASSESSING ADDED AND DROPPED CONSTRAINTS

Consider the linear program

$$\begin{array}{ll} \max & 6y_1 + 4y_2 \\ \text{s.t.} & y_1 + y_2 \leq 3 \\ & y_1 \leq 2 \\ & y_2 \leq 2 \\ & y_1, y_2 \geq 0 \end{array}$$

Assess the qualitative impact of each of the following constraint changes.

- (a) Dropping the first main constraint
- (b) Adding the new constraint $y_2 \geq y_1$

Solution: We apply principles 6.9 and 6.12.

- (a) Dropping a constraint relaxes the feasible set. Thus the maximize optimal value can only stay the same or increase.
- (b) Adding a new constraint tightens the feasible set. Thus the maximize optimal value can only stay the same or decrease.

Effects of Unmodeled Constraints

Often in OR studies the possibility of adding constraints arises in the context of unmodeled phenomena. For example, managers in the Swedish Steel model (6.1) might prefer to use all or none of the 250 kilograms of scrap 2 instead of trying to measure amounts in between.

Such a requirement is discrete. (Do you see why?) Thus there are good reasons for modelers to neglect it in the interest of tractability. But how would results change if such a difficult-to-express limit on solutions had been modeled?

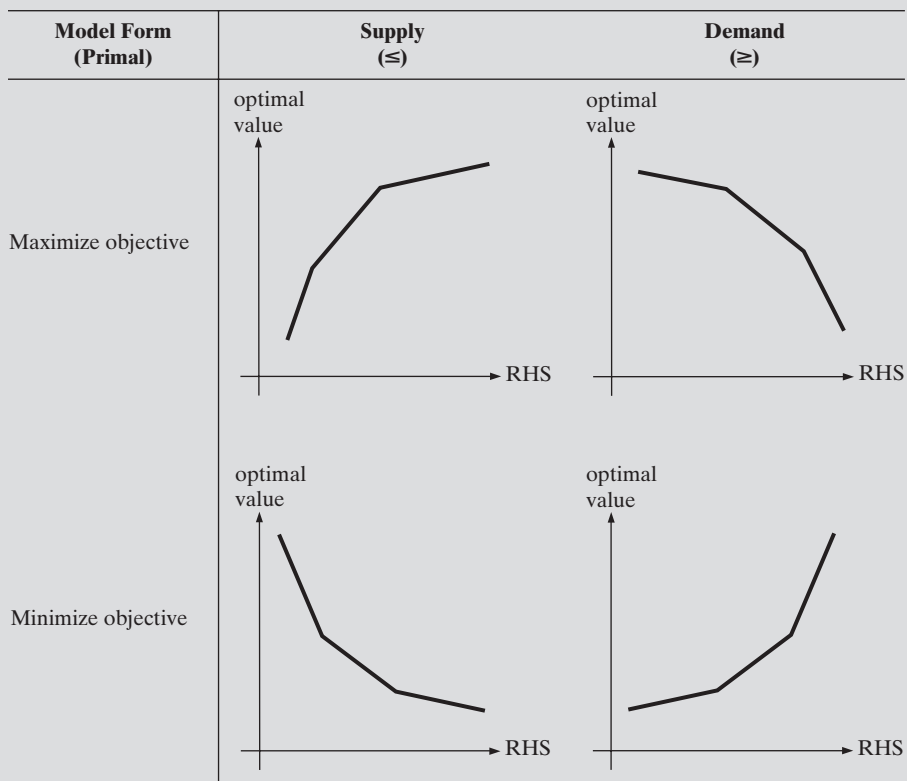
Principle 6.13 Explicitly including previously unmodeled constraints in an optimization model must leave the optimal value either unchanged or worsened.

Changing Rates of Constraint Coefficient Impact

Look again at Figure 6.3. Besides the fact that tightening constraints leaves the optimal value the same or worse, there is another pattern. The more we tighten, the greater the rate of damage. In part (a), for instance, tightening means reducing the RHS coefficient. It has no impact at all for right-hand sides above 83.3. As we continue to squeeze the feasible set, however, the rates of change increase. Ultimately, the model becomes infeasible.

Relaxing constraints produces an opposite effect. For example, in Figure 6.3(b) relaxing means reducing the RHS. The figure shows a steep rate of decline at higher values of the right-hand side that diminishes as we make the coefficient smaller. Every linear program exhibits similar behavior.

Principle 6.14 Coefficient changes that help the optimal value in linear programs by relaxing constraints help less and less as the change becomes large. Changes that hurt the optimal value by tightening constraints hurt more and more. The result is sensitivity plots shaped like the following:



Note that principle [6.14](#) is limited to linear programs. Models with nonconvex feasible sets (see Section 3.4) may behave differently.

To get some intuition about [6.14](#), think of yourself as struggling heroically to force solutions to conform to your favorite supply or demand requirement. The more you tighten the constraint you control, the more it becomes the driving force behind all decisions; its impact becomes greater and greater. On the other hand, if you are driven back and forced to relax your constraint, there are many others to take up the cause; you make less and less difference.

EXAMPLE 6.5: QUALITATIVELY ASSESSING RATES OF CONSTRAINT CHANGE

Determine whether rates of optimal objective function value change would be steeper or less steep with the magnitude of each of the following RHS changes.

- (a) Increase the RHS of $4y_1 - 3y_2 \leq 19$ in a maximize linear program.
- (b) Decrease the RHS of $3y_1 + 50y_2 \geq 40$ in a maximize linear program.
- (c) Decrease the RHS of $14y_1 + 8y_2 \leq 90$ in a minimize linear program.
- (d) Increase the RHS of $3y_1 - 2y_2 \geq 10$ in a minimize linear program.

Solution:

(a) Under principle [6.10](#), the optimal value will improve (become larger) with increases in this RHS because the \leq constraint is being relaxed. Principle [6.14](#) implies that the rate of improvement should stay the same or diminish.

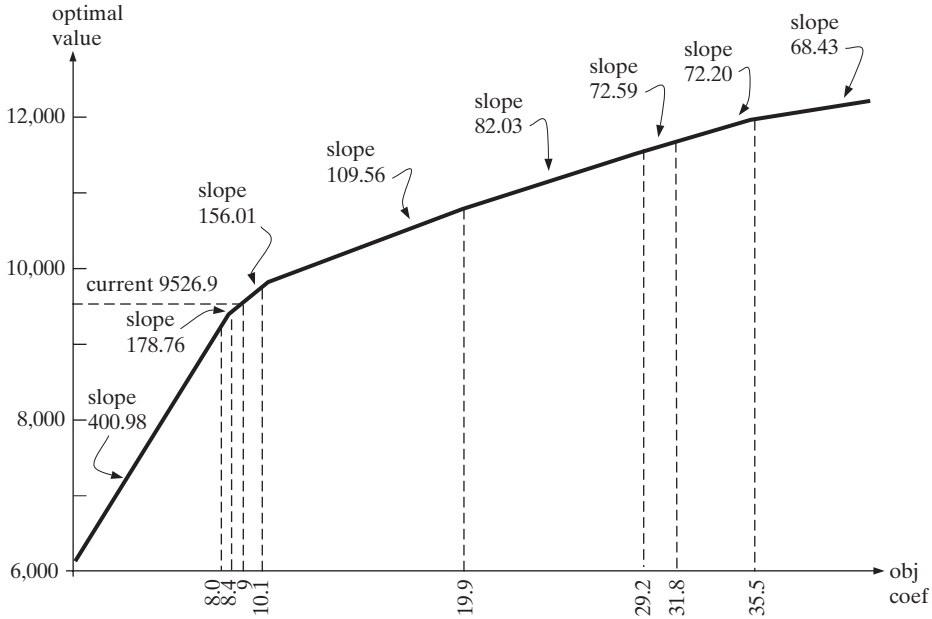
(b) Under principle [6.10](#), the optimal value will improve (become larger) with decreases in this RHS because the \geq constraint is being relaxed. Principle [6.14](#) implies that the rate of decline should stay the same or diminish.

(c) Under principle [6.10](#), the optimal value will worsen (become larger) with increases in this RHS because the \leq constraint is being relaxed. Principle [6.14](#) implies that the rate of improvement should stay the same or steepen.

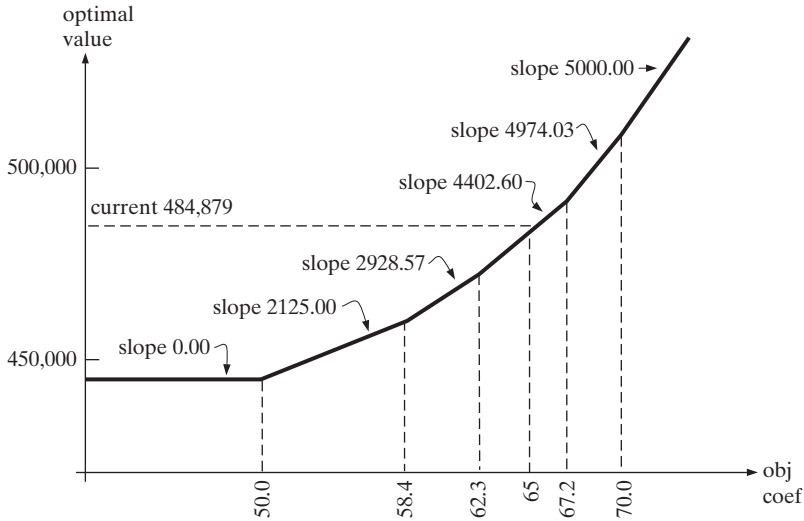
(d) Under principle [6.10](#), the optimal value will worsen (become larger) with increases in this RHS because the \geq constraint is being tightened. Principle [6.14](#) implies that the rate of decline should stay the same or steepen.

Effects of Objective Function Coefficient Changes

Objective function coefficients neither relax nor tighten constraints, but we may still be very interested in their impact on results. Figure 6.4(a) plots the optimal value in the (minimize cost) Swedish Steel model (6.1) as a function of the unit cost for scrap 4 (coefficient of x_4). Part (b) tracks the optimal value in the (maximize margin) CFPL model in Table 4.5 versus the selling price for sheets of $\frac{1}{2}$ -inch AC plywood (coefficient of $z_{1/2,a,c}$).



(a) Min case (Swedish Steel scrap 4 cost)



(b) Max case (CFPL selling price for 1/2-inch AC plywood)

FIGURE 6.4 Sensitivity of Optimal Values to Changes in Objective Function Coefficients

When an objective function coefficient multiplies a nonnegative variable, as most of them do, the direction of change is easy to predict. For example, decreasing the unit cost of an activity certainly makes lower-cost solutions easier to find; the optimal value could only improve.

Principle 6.15 Changing the objective function coefficient of a nonnegative variable in an optimization model affects the optimal value as follows:

Model Form (Primal)	Coefficient Increase	Coefficient Decrease
Maximize objective	Same or better	Same or worse
Minimize objective	Same or worse	Same or better

EXAMPLE 6.6: QUALITATIVELY ASSESSING OBJECTIVE CHANGES

For each of the following objective functions, determine the qualitative impact of the indicated change in objective function coefficient. Assume that all variables are nonnegative.

- (a) Change the coefficient of w_1 to 7 in $\max 12w_1 - w_2$.
- (b) Change the coefficient of w_2 to 9 in $\max 12w_1 - w_2$.
- (c) Change the coefficient of w_1 to 60 in $\min 44w_1 + 3w_2$.
- (d) Change the coefficient of w_2 to -9 in $\min 44w_1 + 3w_2$.

Solution: We apply principle [6.15](#).

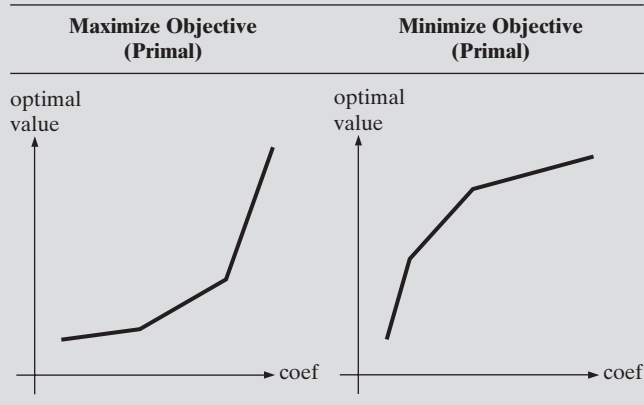
- (a) A change from 12 to 7 is a decrease, implying that the maximize optimal value will stay the same or worsen (decrease).
- (b) A change from -1 to 9 is an increase, implying that the maximize optimal value will stay the same or improve (increase).
- (c) A change from 44 to 60 is an increase, implying that the minimize optimal value will stay the same or worsen (increase).
- (d) A change from 3 to -9 is a decrease, implying that the minimize optimal value will stay the same or improve (decrease).

Changing Rates of Objective Function Coefficient Impact

Readers who absorbed the discussion above about rates of optimal value change caused by variations in constraint coefficients should be surprised by the shapes of curves in Figure 6.4. Consider, for instance, the maximize case of part (b). A move to the right, which increases the coefficient of $z_{1/2,a,c}$, helps the objective function—just as we concluded in [6.15](#).

What is different is that the more we increase the coefficient, the more rapidly the optimal value improves. In contrast to the decreasing returns we saw with constraint changes that helped the optimal value, favorable objective function changes produce increasing rates of return.

Principle 6.16 Objective function coefficient changes that help the optimal value in linear programs help more and more as the change becomes large. Changes that hurt the optimal value hurt less and less. The result is sensitivity plots shaped like the following:



As with [6.14], principle [6.16] is limited to linear programs with their convex feasible sets. Nonconvex optimization models may perform differently.

A full explanation of the incongruity between rates of change in constraint and objective function coefficients must await the duality development of the next few sections. For now, return to your struggling hero role, this time by manipulating the unit cost or benefit of some activity. Changing your objective function coefficient in the direction that helps the optimal value draws more and more action to your activity. The more it dominates the solution, the more its objective function coefficient affects the optimal value. On the other hand, changes that hurt the optimal value are cushioned by transfer of responsibility to other activities. They have less and less impact as the change becomes large.

EXAMPLE 6.7: QUALITATIVELY ASSESSING RATES OF OBJECTIVE CHANGE

Determine whether the objective function coefficient change indicated would increase or decrease the optimal objective function value in each of the following, and indicate whether the rate of change will steepen or become less steep with the magnitude of the change. Assume that all variables are nonnegative.

- Increase 42 on x_1 in $\max 42x_1 + 13x_2 - 9x_3$.
- Decrease -9 on x_3 in $\max 42x_1 + 13x_2 - 9x_3$.
- Increase -2 on x_2 in $\min 12x_1 - 2x_2 + x_4$.
- Decrease 0 on x_3 in $\min 12x_1 - 2x_2 + x_4$.

Solution:

(a) The optimal value improves (increases) with any objective function coefficient increase in a maximize model (principle [6.15]). Thus, under principle [6.16] the rate of increase should stay the same or steepen.

(b) A decrease would be to $c_3 < -9$. The optimal value worsens (decreases) with any objective function coefficient decrease in a maximize model (principle [6.15]). Thus, under principle [6.16] the rate of increase should stay the same or lessen.

(c) The optimal value worsens (increases) with any objective function coefficient increase in a minimize model (principle [6.15]). Thus, under principle [6.16] the rate of increase should stay the same or lessen.

(d) The optimal value improves (decreases) with any objective function coefficient decrease in a minimize model (principle [6.15]). Thus, under principle [6.16] the rate of increase should stay the same or steepen.

Effects of Adding or Dropping Variables

The final sort of sensitivity analysis we need to consider is the adding or dropping of activities (which is implemented mathematically as adding or dropping of decision variables). For example, what happens if we decide to consider a new scrap source in the Swedish Steel model (6.1)? What happens if we drop scrap 4?

Adding activities offers new choices; dropping them reduces possibilities. The direction of optimal value change should then be apparent.

Principle 6.17 Adding optimization model activities (variables) must leave the optimal value unchanged or improved. Dropping activities will leave the value unchanged or degraded.

EXAMPLE 6.8: QUALITATIVELY ASSESSING ADDED AND DROPPED VARIABLES

Determine the qualitative impact of adding or dropping a variable in linear programs with each of the following objective functions.

(a) $\max 27y_1 - y_2 + 4y_3$

(b) $\min 33y_1 + 11y_2 + 39y_3$

Solution: We apply principle [6.17].

(a) Adding a variable in this maximize LP can only help, leaving the optimal value the same or higher. Dropping a variable would make it the same or lower.

(b) Adding a variable in the minimize LP can only help, leaving the optimal value the same or lower. Dropping a variable would make it the same or higher.

6.3 QUANTIFYING SENSITIVITY TO CHANGES IN LP MODEL COEFFICIENTS: A DUAL MODEL

All the LP sensitivities of Section 6.2 related to directions of changes rather than magnitudes. It is certainly helpful to know whether, say, a right-hand-side modification will help or hurt the optimal value. Still, we would gain much more insight if we could quantify the rate of change.

The OR approach to computing quantities we would like to know is to represent them as variables, formulate a model of how they interrelate, and solve the model to obtain their values. In this and Section 6.4 we pursue that approach. More specifically, we define new sensitivity variables and formulate the conditions that a quantitative sensitivity analysis should fulfill.

Primals and Duals Defined

With one model being used to analyze sensitivity of another, it is important to distinguish between primal and dual.

Definition 6.18 | The **primal** is the given optimization model, the one formulating the application of primary interest.

All the linear programs formulated in earlier chapters were primals.

Definition 6.19 | The **dual** corresponding to a given primal formulation is a closely related LP with decision variables and constraints that quantify the sensitivity of primal results to changes in its parameters.

The remarkable fact is that the needed dual is an “orthogonal” LP over exactly the same parameter values as the primal, but with constraints of the dual corresponding to variables of the primal, and variables of the dual corresponding to constraints of the primal. We will see in Section 6.7 that primal and dual share a host of elegant connections and symmetries reaching well beyond sensitivity issues.

Dual Variables

To begin deriving the dual, take another look at the Figure 6.3 plots of the (primal) Swedish Steel model’s optimal value as a function of right-hand side. Dual variables quantify slopes of such curves.

Principle 6.20 | There is one **dual variable** for each main primal constraint. Each reflects the rate of change in primal optimal value per unit increase from the given right-hand-side value of the corresponding constraint.

We often denote the dual variable on primal main constraint i by v_i .

The right-hand side of primal constraint (6.2) is 75. Figure 6.3(a) shows the rate of change in the primal optimal value, and thus the value of the dual variable corresponding to this availability constraint is -3.38 kroner/kilogram.

Of course, we would prefer to know such slopes for all possible values of right-hand sides. For instance, in Figure 6.3(a) we have

$+\infty$	between	$-\infty$	and	0
-4.98	between	0	and	60.42
-3.38	between	60.42	and	83.33
0	between	83.33	and	$+\infty$

However, several LPs have to be solved to derive such plots.

Since the dual only provides subsidiary analysis of the primal, which is the model of real interest, definition [6.20](#) adopts a more manageable standard. Optimal dual variable values for any single LP yield rates of change only at the original RHS value. For instance, only the -3.38 rate at $\text{RHS} = 75$ would be obtained for the constraint of Figure 6.3(a).

This definition is somewhat unclear if the RHS value happens to fall at exactly one of the points where slopes change. For example, if the given RHS in Figure 6.3(a) had been 83.33, the corresponding dual variable value could come out either -3.38 or 0.00 (or anything in between). The first applies to the left of 83.33 and the second to the right. At exactly 83.33 the slope is ambiguous.

EXAMPLE 6.9: UNDERSTANDING DUAL VARIABLES

Refer to Figure 6.3(b)'s plot of sensitivity to minimum chromium constraint (6.3).

- (a) What is the corresponding optimal dual variable value?
- (b) What would it be if the constraint RHS had been 7.0?
- (c) What would it be if the constraint RHS had been 9.0?

Solution:

- (a) By definition [6.20](#), the dual variable corresponding to this minimum chromium constraints should equal the slope of the sensitivity curve at $\text{RHS} = 10.0$. That is, the dual variable has value 36.73 kroner/kilogram.
- (b) For $\text{RHS} = 7.0$, the corresponding dual variable value or rate of change is 8.57 kroner/kilogram.
- (c) At $\text{RHS} = 9.0$, the dual variable value is ambiguous. Computation might produce either the 8.57 to the left or the 36.73 to the right.

Dual Variable Types

We want to develop constraints and conditions that dual variables must satisfy to provide meaningful sensitivity information. Qualitative principles [6.9](#) and [6.10](#) already yield sign restrictions on rates of change in inequality right-hand sides. For example, an increase in the RHS of a \leq constraint in a minimize model relaxes the constraint, which leaves the optimal value unchanged or reduced. The corresponding $v \leq 0$.

Principle 6.21 The linear programming dual variable on constraint i has type as follows:

Primal	i is \leq	i is \geq	i is $=$
Minimize objective	$v_i \leq 0$	$v_i \geq 0$	Unrestricted
Maximize objective	$v_i \geq 0$	$v_i \leq 0$	Unrestricted

Notice that principle [6.21](#) specifies dual variables for $=$ constraints to be unrestricted in sign (URS). Since an equality can act as either a \leq or a \geq (principle [6.5](#)), the corresponding dual rate of change can be either positive or negative.

Two Crude Application Again

To have an easy example to follow, let us return to the familiar Two Crude refining model of Section 2.1. That primal linear program was

$$\begin{aligned}
 \min \quad & 100x_1 + 75x_2 \\
 \text{s.t.} \quad & 0.3x_1 + 0.4x_2 \geq 2 & : v_1 & \text{(gasoline demand)} \\
 & 0.4x_1 + 0.2x_2 \geq 1.5 & : v_2 & \text{(jet fuel demand)} \\
 & 0.2x_1 + 0.3x_2 \geq 0.5 & : v_3 & \text{(lubricant demand)} \\
 & 1x_1 \leq 9 & : v_4 & \text{(Saudi availability)} \\
 & \quad + 1x_2 \leq 6 & : v_5 & \text{(Venezuelan availability)} \\
 & x_1, x_2 \geq 0
 \end{aligned} \tag{6.4}$$

The unique primal optimal solution is $x_1^* = 2, x_2^* = 3.5$, with optimal value 92.5.

Notice that we have assigned dual variables to each main constraint. Applying principle [6.21](#), sign restriction constraints for the 5 Two Crude dual variables become

$$v_1 \geq 0, \quad v_2 \geq 0, \quad v_3 \geq 0, \quad v_4 \leq 0, \quad v_5 \leq 0 \tag{6.5}$$

Increasing \geq right-hand sides in a minimize problem can only raise the optimal cost or leave it unchanged. Increasing those of \leq 's can only lower it or leave it unchanged.

EXAMPLE 6.10: CHOOSING DUAL VARIABLE TYPE

For each of the following primal linear programs, determine whether the dual variables indicated should be nonnegative, nonpositive, or of URS type.

$$\begin{aligned}
 \text{(a) } \min \quad & + 5x_1 + 1x_2 + 4x_3 + 5x_4 \\
 \text{s.t.} \quad & + 1x_1 + 4x_2 + 2x_3 = 36 & : v_1 \\
 & + 3x_1 + 2x_2 + 8x_3 + 2x_4 \leq 250 & : v_2 \\
 & - 5x_1 - 2x_2 + 1x_3 + 1x_4 \leq 7 & : v_3 \\
 & \quad + 1x_3 + 1x_4 \geq 60 & : v_4 \\
 & x_1, x_2, x_3, x_4 \geq 0
 \end{aligned}$$

$$\begin{array}{rcllcl}
 \text{(b) max} & + 13x_1 & + 24x_2 & + 5x_3 & + 50x_4 & & \\
 \text{s.t.} & + 1x_1 & + 3x_2 & & & \geq 89 & : v_1 \\
 & & & + 1x_3 & + 5x_4 & \geq 60 & : v_2 \\
 & + 10x_1 & + 6x_2 & + 8x_3 & + 2x_4 & \leq 608 & : v_3 \\
 & & + 1x_2 & & + 1x_4 & = 28 & : v_4 \\
 & & & & & & x_1, x_2, x_3, x_4 \geq 0
 \end{array}$$

Solution: We apply rule [6.21](#).

(a) v_1 URS, $v_2 \leq 0$, $v_3 \leq 0$, $v_4 \geq 0$

(b) $v_1 \leq 0$, $v_2 \leq 0$, $v_3 \geq 0$, v_4 URS

Dual Variables as Implicit Marginal Resource Prices

Because the dual variables tell us the rate of change in the optimal value for another unit of each RHS, they provide a sort of implicit price on the resource of each constraint model. To be more precise, they yield what economists call **marginal prices**.

Principle 6.22 Dual variables provide **implicit prices** for the marginal unit of the resource modeled by each constraint as its right-hand-side limit is encountered.

In the Two Crude model (6.4), for instance, the first constraint models gasoline demand. Variable v_1 , which is in units of thousands of dollars per thousand barrels, will tell us the implicit price of gasoline at the margin when RHS = 2000 barrels are demanded, that is, how much the last 1000 barrels cost to produce in the optimal operating plan. Similarly, variable v_4 reflects the marginal impact of the Saudi availability constraint at its current level of 9000 barrels, which is the implicit value of another 1000 barrels of Saudi crude.

EXAMPLE 6.11: INTERPRETING DUAL VARIABLES

The Top Brass Trophy application model (5.1) of Section 5.1 is

$$\begin{array}{rcllcl}
 \text{max} & 12x_1 & + 9x_2 & & & \text{[profit (dollars)]} \\
 \text{s.t.} & x_1 & & \leq 1000 & : v_1 & \text{(brass footballs)} \\
 & x_2 & & \leq 1500 & : v_2 & \text{(brass soccer balls)} \\
 & x_1 & + x_2 & \leq 1750 & : v_3 & \text{(brass plaques)} \\
 & 4x_1 & + 2x_2 & \leq 4800 & : v_4 & \text{[wood (board feet)]} \\
 & & & & & x_1, x_2 \geq 0
 \end{array}$$

Interpret each of the 4 corresponding dual variables and determine their signs and units.

Solution: Applying interpretation [6.22], v_1 is the marginal value or contribution to profit of brass footballs at the current 1000 availability level, that is, how much we would pay for another one. Under principle [6.21], $v_1 \geq 0$, and its units are dollars per brass football. Similarly, $v_2 \geq 0$ is the marginal value of brass soccer balls at the current 1500 availability level (in dollars per soccer ball), $v_3 \geq 0$ is the marginal value of brass plaques at the current 1750 availability (in dollars per plaque), and $v_4 \geq 0$ is the marginal value of wood at the current 4800 availability (in dollars per board foot).

Implicit Activity Pricing in Terms of Resources Produced and Consumed

The activity associated with any decision variable both consumes and creates constraints' resources (review Figure 6.1). We also know that the activity's nonzero constraint coefficients can be interpreted as amounts consumed or created per unit of the activity (principle [6.8]).

By summing those coefficients times the implicit price v_i of the resources involved, we can obtain an implicit marginal value (minimize problems) or price (maximize problems) for the entire activity.

Principle 6.23 The implicit marginal value (minimize problems) or price (maximize problems) of a unit of LP activity (primal variable) j implied by dual variable values v_i is $\sum_i a_{i,j}v_i$, where $a_{i,j}$ denotes the coefficient of activity j in the left-hand side of constraint i .

Venezuelan activity $j = 2$ in the Two Crude application illustrates. Given resource prices v_1, \dots, v_5 , each unit of Venezuelan crude activity is implicitly worth

$$\sum_{i=1}^5 a_{i,2}v_i = 0.4v_1 + 0.2v_2 + 0.3v_3 + 1v_5$$

Noting sign restrictions (6.5), the first three terms measure the positive contribution of fulfilling demand constraints. The last deducts ($v_5 \leq 0$) the implicit worth of the availability resource consumed.

EXAMPLE 6.12: IMPLICITLY PRICING ACTIVITIES

Develop and interpret expressions for the price per unit of each activity in the Top Brass Trophy model reviewed in Example 6.11.

Solution: In the Top Brass example, x_1 represents production of football trophies, and x_2 , production of soccer trophies. Under principle [6.23], the implicit price or

marginal cost to produce a football trophy is the sum of coefficients on x_1 times the marginal value of the associated commodities, that is,

$$\sum_{i=1}^4 a_{i,1}v_i = 1v_1 + 1v_3 + 4v_4$$

This expression simply sums the marginal cost of the 1 football, 1 plaque, and 4 board feet of wood needed to make a football trophy. Similarly, the marginal cost of producing a soccer trophy is

$$\sum_{i=1}^4 a_{i,2}v_i = 1v_2 + 1v_3 + 2v_4$$

Main Dual Constraints to Enforce Activity Pricing

The [6.23](#) notion of implicit activity worth per unit also leads to the main dual constraints. If dual variables v_i are really to reflect the value of constraint resources in the primal optimal value, the implied activity worth must be consistent with explicit unit costs or benefits in the primal objective function.

To be specific, dual variables in a minimize (cost) problem would overvalue an activity if they priced it above its true cost coefficient c_j . An activity implicitly worth more than it costs at optimality should be used in greater quantity. But then we would be improving on an optimal solution—an impossibility. Similarly for maximize (benefit) problems, we would want to reject any v_i 's that priced the net value of resources that an activity uses below its real benefit c_j . Otherwise, it too would yield an improvement on an already optimal solution.

Principle 6.24 For each nonnegative variable activity x_j in a minimize linear program, there is a corresponding **main dual constraint** $\sum_i a_{i,j}v_i \leq c_j$ requiring the net marginal value of the activity not to exceed its given cost. In a maximize problem, main dual constraints for $x_j \geq 0$ are $\sum_i a_{i,j}v_i \geq c_j$, which keeps the net marginal cost of each activity at least equal to its given benefit.

Illustrating again with the Two Crude model (6.4), principle [6.24](#) leads to one constraint for primal variable x_1 and another for x_2 .

$$\begin{aligned} 0.3v_1 + 0.4v_2 + 0.2v_3 + 1v_4 &\leq 100 \\ 0.4v_1 + 0.2v_2 + 0.3v_3 + 1v_5 &\leq 75 \end{aligned} \tag{6.6}$$

Without such limits, say if the worth evaluation on the left-hand side of the first inequality were allowed to reach 105, the v_i would be telling us that another unit of activity $j = 1$ would produce a net impact on the optimal value of \$105 (thousand) yet costs only \$100 (thousand). Such bargains cannot be allowed if the v_i are to mean what we want them to mean.

EXAMPLE 6.13: FORMING MAIN DUAL CONSTRAINTS

Formulate and interpret the main dual constraints of the linear programs in Example 6.10.

Solution: We apply principle [6.24](#).

(a) For this minimize model, main dual constraints are

$$\begin{array}{rccccrcr} +1v_1 & +3v_2 & -5v_3 & & & & \leq 5 \\ +4v_1 & +2v_2 & -2v_3 & & & & \leq 1 \\ +2v_1 & +8v_2 & +1v_3 & +1v_4 & & & \leq 4 \\ & +2v_2 & +1v_3 & +1v_4 & & & \leq 5 \end{array}$$

They may be interpreted as requiring that the net marginal value of each activity under an optimal choice of dual variable values cannot exceed its given objective function cost.

(b) For this maximize model, main dual constraints are

$$\begin{array}{rccccrcr} +1v_1 & & +10v_3 & & & & \geq 13 \\ +3v_1 & & +6v_3 & +1v_4 & & & \geq 24 \\ & +1v_2 & +8v_3 & & & & \geq 5 \\ & +5v_2 & +2v_3 & +1v_4 & & & \geq 50 \end{array}$$

They may be interpreted as requiring that the net marginal cost of each activity under an optimal choice of dual variable values cannot be less than its given objective function benefit.

Optimal Value Equality between Primal and Dual

If dual variables are to price the resources associated with constraints correctly, the supplies and demands for those resources should evaluate to exactly the primal optimal value.

Principle 6.25 If a primal linear program has an optimal solution, its optimal value $\sum_j c_j x_j^*$ equals the corresponding optimal dual implicit total value $\sum_i b_i v_i^*$ of all constraint resources.

In the Two Crude case, requirement [6.25](#) implies that

$$100x_1^* + 75x_2^* = 2v_1^* + 1.5v_2^* + 0.5v_3^* + 9v_4^* + 6v_6^*$$

The sum of optimal prices on all constraints times current limiting values should recover the optimal primal objective value.

EXAMPLE 6.14: EXPRESSING PRIMAL–DUAL VALUE EQUALITY

Express and interpret optimal value equality requirement [6.25](#) for each of the linear programs in Example 6.10.

Solution: We want the primal optimal value to equal exactly the implicit total value of all constraint resources.

(a) For this minimize model

$$5x_1 + 1x_2 + 4x_3 + 5x_4 = 36v_1 + 250v_2 + 7v_3 + 60v_4$$

(b) For this maximize model

$$13x_1 + 24x_2 + 5x_3 + 50x_4 = 89v_1 + 60v_2 + 608v_3 + 28v_4$$

Primal Complementary Slackness between Primal Constraints and Dual Variable Values

Recall that an inequality is **active** if it is satisfied as an equality by a given solution, and **inactive** otherwise. If an inequality is inactive at the primal optimal solution, small changes in its right-hand side would not affect the optimal value at all; the constraint has slack. We can immediately deduce the value of the corresponding dual variable v_i .

Principle 6.26 Either the primal optimal solution makes main inequality constraint i active or the corresponding dual variable $v_i = 0$.

Equalities are always active, so there is no corresponding condition.

Principle [6.26](#) carries the somewhat bulky name **primal complementary slackness** because it asserts that either each primal inequality or its corresponding dual sign restriction ($v_i \geq 0$ or $v_i \leq 0$) will be slack (inactive) at optimality. To illustrate for the Two Crude model (6.4), we first substitute the primal optimum to determine which constraints are active.

$$\begin{array}{rcll}
 + 0.3(2) & + 0.4(3.5) & = & 2.0 \quad (\text{active}) \\
 + 0.4(2) & + 0.2(3.5) & = & 1.5 \quad (\text{active}) \\
 + 0.2(2) & + 0.3(3.5) & = & 1.45 > 0.5 \quad (\text{inactive}) \\
 + 1(2) & & = & 2 < 9 \quad (\text{inactive}) \\
 & + 1(3.5) & = & 3.5 < 6 \quad (\text{inactive})
 \end{array}$$

The last three have slack. Thus small changes in corresponding right-hand-side co-efficients would have no impact on optimal value. In accord with principle [6.26](#), associated dual variables must = 0.

EXAMPLE 6.15: EXPRESSING PRIMAL COMPLEMENTARY SLACKNESS

Express and interpret primal complementary slackness conditions for the linear programs of Example 6.10.

Solution: We apply principle [6.26].

(a) Primal complementary slackness conditions for this minimize model are

$$\begin{aligned} + 3x_1 + 2x_2 + 8x_3 + 2x_4 &= 250 & \text{or } v_2 &= 0 \\ - 5x_1 - 2x_2 + 1x_3 + 1x_4 &= 7 & \text{or } v_3 &= 0 \\ + 1x_3 + 1x_4 &= 60 & \text{or } v_4 &= 0 \end{aligned}$$

They specify that a main primal inequality's right-hand side can affect the optimal value only if the constraint is active. There is no condition for equality constraint 1 because it is always active.

(b) Primal complementary slackness conditions for this maximize model are

$$\begin{aligned} + 1x_1 + 3x_2 &= 89 & \text{or } v_1 &= 0 \\ + 1x_3 + 5x_4 &= 60 & \text{or } v_2 &= 0 \\ + 10x_1 + 6x_2 + 8x_3 + 2x_4 &= 608 & \text{or } v_3 &= 0 \end{aligned}$$

They, too, specify that changes in a main primal inequality's right-hand side can affect the optimal value only if the constraint is active. There is no such condition for equality constraint 4.

Dual Complementary Slackness between Dual Constraints and Primal Variable Values

Constraints [6.24] keep activity prices below true cost in minimize problems and above true benefit in maximize problems. At first glance, it might seem reasonable to ask for more (i.e., demand that activity prices match exactly corresponding c_j).

There would be a practical mathematical difficulty with such a requirement in the common case where we have many more primal variables than main constraints. Relatively few v_i would simultaneously have to satisfy constraints

$$\sum_i a_{i,j}v_i = c_j$$

for too many activities j .

More important, we want the dual variables to measure resource value at optimality. The only (nonnegative) primal variables involved in an optimal solution are those with optimal $x_j^* > 0$. Limiting perfect valuation to this more limited list of activities produces our final set of requirements—**dual complementary slackness** conditions.

Principle 6.27 | Either a nonnegative primal variable has optimal value $x_j = 0$ or the corresponding dual prices v_i must make the j th dual constraint [6.24] active.

This time the complementarity is between the primal nonnegativity $x_j \geq 0$ and the corresponding dual inequality [6.24].

For example, the tiny Two Crude optimum had both primal variables positive at optimality. Principle [6.27] now tells us that the dual values v_i we are trying to understand must make both constraints of (6.6) active.

EXAMPLE 6.16: EXPRESSING DUAL COMPLEMENTARY SLACKNESS

Express and interpret dual complementary slackness conditions for the linear programs of Example 6.10.

Solution: We apply principle [6.27] to the main dual constraints of Example 6.13.

(a) For this minimize model, dual complementary slackness conditions are

$$\begin{aligned}x_1 = 0 & \text{ or } 1v_1 + 3v_2 - 5v_3 = 5 \\x_2 = 0 & \text{ or } 4v_1 + 2v_2 - 2v_3 = 1 \\x_3 = 0 & \text{ or } 2v_1 + 8v_2 + 1v_3 + 1v_4 = 4 \\x_4 = 0 & \text{ or } 2v_2 + 1v_3 + 1v_4 = 5\end{aligned}$$

They may be interpreted to mean that marginal resource prices v_i must make the implicit value of each activity used in an optimal solution equal its given cost.

(b) For this maximize model, dual complementary slackness conditions are

$$\begin{aligned}x_1 = 0 & \text{ or } 1v_1 + 10v_3 = 13 \\x_2 = 0 & \text{ or } 3v_1 + 6v_3 + 1v_4 = 24 \\x_3 = 0 & \text{ or } 1v_2 + 8v_3 = 5 \\x_4 = 0 & \text{ or } 5v_2 + 2v_3 + 1v_4 = 50\end{aligned}$$

They may be interpreted to mean that marginal resource prices v_i must make the implicit cost of each activity used in an optimal solution equal to its given benefit.

6.4 FORMULATING LINEAR PROGRAMMING DUALS

The remarkable fact about linear programs is that all the quantitative sensitivity requirements of Section 6.3 can actually be achieved. Not only that, but they can be achieved with very minor computation as a by-product of the search for a primal optimum.

The secret behind these elegant results is deceptively simple: We think of dual variables v_i as decision variables of a new dual linear program defined on the same constants as the primal and determine their values by optimizing that dual. In this section we show how to formulate duals, and in the next section we verify the primal-to-dual relationships of Section 6.3 and more. As usual, x_j will always denote the j th primal variable, c_j its objective function coefficient, and $a_{i,j}$ its coefficient in the i th constraint; v_i is the dual variable for the i th constraint, and b_i is the right-hand side.

Form of the Dual for Nonnegative Primal Variables

To form a dual when primal decision variables are nonnegative, we optimize total resource value $\sum_i b_i v_i$ subject to the main dual constraints [6.24] and dual variable-type restrictions [6.21] derived in Section 6.3.

We can summarize in compact matrix format as follows:

Principle 6.28 | **Minimize** primal LPs over variables $\mathbf{x} \geq \mathbf{0}$ have duals over variables \mathbf{v} as follows.

$\begin{aligned} \min \quad & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \begin{cases} \leq \\ \geq \\ = \end{cases} \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$	$\begin{aligned} \max \quad & \mathbf{b} \cdot \mathbf{v} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{v} \leq \mathbf{c} \\ & \mathbf{v} \begin{cases} \leq \mathbf{0} \\ \geq \mathbf{0} \\ \text{URS} \end{cases} \end{aligned}$
--	--

where main constraint forms of the primal align with corresponding variable types of the dual.

Principle 6.29 | **Maximize** primal LPs over variables $\mathbf{x} \geq \mathbf{0}$ have duals over variables \mathbf{v} as follows.

$\begin{aligned} \max \quad & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \begin{cases} \leq \\ \geq \\ = \end{cases} \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$	$\begin{aligned} \min \quad & \mathbf{b} \cdot \mathbf{v} \\ \text{s.t.} \quad & \mathbf{A}^T \mathbf{v} \geq \mathbf{c} \\ & \mathbf{v} \begin{cases} \geq \mathbf{0} \\ \leq \mathbf{0} \\ \text{URS} \end{cases} \end{aligned}$
--	--

where main constraint forms of the primal align with corresponding variable types of the dual.

Our familiar Two Crude model (6.4) is a specific example. That minimize primal has dual

$$\begin{aligned} \max \quad & 2v_1 + 1.5v_2 + 0.5v_3 + 9v_4 + 6v_5 \\ \text{s.t.} \quad & 0.3v_1 + 0.4v_2 + 0.2v_3 + 1v_4 \leq 100 \\ & 0.4v_1 + 0.2v_2 + 0.3v_3 + 1v_5 \leq 75 \\ & v_1, v_2, v_3 \geq 0 \\ & v_4, v_5 \leq 0 \end{aligned} \tag{6.7}$$

An optimal solution is $v_1^* = 100$, $v_2^* = 175$, $v_3^* = v_4^* = v_5^* = 0$, with dual objective function value 462.5.

EXAMPLE 6.17: STATING DUALS WITH NONNEGATIVE x_j

State the duals of each of the following primal linear programs.

$$\begin{array}{llllll}
 \text{(a) min} & + 30x_1 & & + 5x_3 & & \\
 \text{s.t.} & + 1x_1 & - 1x_2 & + 1x_3 & \geq 1 & : v_1 \\
 & + 3x_1 & + 1x_2 & & = 4 & : v_2 \\
 & & + 4x_2 & + 1x_3 & \leq 10 & : v_3 \\
 & x_1, x_2, x_3 & \geq 0 & & &
 \end{array}$$

$$\begin{array}{llllll}
 \text{(b) max} & + 10x_1 & + 9x_2 & - 6x_3 & & \\
 \text{s.t.} & + 2x_1 & + 1x_2 & & \geq 3 & : v_1 \\
 & + 5x_1 & + 3x_2 & - 1x_3 & \leq 15 & : v_2 \\
 & & + 1x_2 & + 1x_3 & = 1 & : v_3 \\
 & x_1, x_2, x_3 & \geq 0 & & &
 \end{array}$$

Solution:

(a) From construction [6.28], the dual of this minimize model is a maximize model with objective function

$$\max \quad + 1v_1 \quad + 4v_2 \quad + 10v_3$$

derived from the RHS of the primal. Constraints include one main constraint for each primal variable (principle [6.24]) and type restrictions from table [6.21]. Specifically,

$$\begin{array}{llll}
 \text{s.t.} & + 1v_1 & + 3v_2 & \leq 30 \\
 & - 1v_1 & + 1v_2 & + 4v_3 \leq 0 \\
 & + 1v_1 & & + 1v_3 \leq 5 \\
 & v_1 \geq 0, v_2 \text{ URS}, v_3 \leq 0 & &
 \end{array}$$

(b) From construction [6.29], the dual of this maximize model is a minimize with objective function

$$\min \quad + 3v_1 \quad + 15v_2 \quad + 1v_3$$

derived from the RHS of the primal. Constraints include one main constraint for each primal variable (principle [6.24]) and type restrictions from table [6.21]. Specifically,

$$\begin{array}{llll}
 \text{s.t.} & + 2v_1 & + 5v_2 & \geq 10 \\
 & + 1v_1 & + 3v_2 & + 1v_3 \geq 9 \\
 & & - 1v_2 & + 1v_3 \geq -6 \\
 & v_1 \leq 0, v_2 \geq 0, v_3 \text{ URS} & &
 \end{array}$$

Duals of LP Models with Nonpositive and Unrestricted Variables

So far, all our duality results have assumed nonnegative primal variables. Not much is lost, because the majority of LP models have only nonnegative variables. Still, to obtain full symmetry between primal and dual, both should be allowed to have nonpositive, nonnegative, and unrestricted variables.

Table 6.1 shows the complete picture. Notice the elegant symmetries:

- A minimize primal yields a maximize dual and vice versa.
- Objective function coefficients of primal or dual become the right-hand sides of the other, and the right-hand sides of the the primal or dual become the objective function coefficients of the other.
- There is one dual variable for every primal constraint, and one dual constraint for every primal variable.
- Main constraint forms of primal or dual align with corresponding variable type constraints in the other.

TABLE 6.1 Corresponding Elements of Primal and Dual Linear Programs

	Primal Element	Corresponding Dual Element
max form	Objective $\max \sum_j c_j x_j$	Objective $\min \sum_i b_i v_i$
	Constraint $\sum_j a_{i,j} x_j \geq b_i$	Variable $v_i \leq 0$
	Constraint $\sum_j a_{i,j} x_j = b_i$	Variable v_i unrestricted
	Constraint $\sum_j a_{i,j} x_j \leq b_i$	Variable $v_i \geq 0$
	Variable $x_j \geq 0$	Constraint $\sum_i a_{i,j} v_j \geq c_j$
	Variable x_j URS	Constraint $\sum_i a_{i,j} v_j = c_j$
	Variable $x_j \leq 0$	Constraint $\sum_i a_{i,j} v_j \leq c_j$
min form	Objective $\min \sum_j c_j x_j$	Objective $\max \sum_i b_i v_i$
	Constraint $\sum_j a_{i,j} x_j \geq b_i$	Variable $v_i \geq 0$
	Constraint $\sum_j a_{i,j} x_j = b_i$	Variable v_i unrestricted
	Constraint $\sum_j a_{i,j} x_j \leq b_i$	Variable $v_i \leq 0$
	Variable $x_j \geq 0$	Constraint $\sum_i a_{i,j} v_i \leq c_j$
	Variable x_j URS	Constraint $\sum_i a_{i,j} v_i = c_j$
	Variable $x_j \leq 0$	Constraint $\sum_i a_{i,j} v_i \geq c_j$

EXAMPLE 6.18: FORMING DUALS OF ARBITRARY LPS

Form the dual of each of the following linear programs.

(a)

$$\begin{array}{rcll}
 \max & + 6x_1 & - 1x_2 & + 13x_3 \\
 \text{s.t.} & + 3x_1 & + 1x_2 & + 2x_3 & = 7 \\
 & + 5x_1 & - 1x_2 & & \leq 6 \\
 & & + 1x_2 & + 1x_3 & \geq 2 \\
 & x_1 \geq 0, & x_2 \leq 0 & &
 \end{array}$$

$$\begin{array}{ll}
 \text{(b)} & \min \quad + 7x_1 \qquad \qquad \qquad + 44x_3 \\
 & \text{s.t.} \quad - 2x_1 \quad - 4x_2 \quad + 1x_3 \leq 15 \\
 & \qquad \qquad + 1x_1 \quad + 4x_2 \qquad \qquad \geq 5 \\
 & \qquad \qquad + 5x_1 \quad - 1x_2 \quad + 3x_3 = -11 \\
 & \qquad \qquad x_1 \leq 0, x_3 \geq 0
 \end{array}$$

Solution:

(a) Assigning dual variables v_1, v_2, v_3 to the three main constraints, we apply the “max form” part of Table 6.1 to produce the following dual:

$$\begin{array}{ll}
 \min & + 7v_1 \quad + 6v_2 \quad + 2v_3 \\
 \text{s.t.} & + 3v_1 \quad + 5v_2 \qquad \qquad \geq 6 \\
 & + 1v_1 \quad - 1v_2 \quad + 1v_3 \leq -1 \\
 & + 2v_1 \qquad \qquad \qquad + 1v_3 = 13 \\
 & v_1 \text{ URS}, v_2 \geq 0, v_3 \leq 0
 \end{array}$$

Notice that unrestricted x_3 yields an equality dual constraint.

(b) Assigning dual variables v_1, v_2, v_3 to the three main constraints, we apply the “min form” part of Table 6.1 to produce the following dual:

$$\begin{array}{ll}
 \max & + 15v_1 \quad + 5v_2 \quad - 11v_3 \\
 \text{s.t.} & - 2v_1 \quad + 1v_2 \quad + 5v_3 \geq 7 \\
 & - 4v_1 \quad + 4v_2 \quad - 1v_3 = 0 \\
 & + 1v_1 \qquad \qquad \qquad + 3v_3 \leq 44 \\
 & v_1 \leq 0, v_2 \geq 0, v_3 \text{ URS}
 \end{array}$$

Again, unrestricted x_2 yields an equality dual constraint.

Dual of the Dual is the Primal

With this full symmetry of Table 6.1, we can also demonstrate one final form of primal-to-dual symmetry. Look back at the Two Crude model dual (6.7) and suppose that it were the primal:

$$\begin{array}{ll}
 \max & 2x_1 + 1.5x_2 + 0.5x_3 + 9x_4 + 6x_5 \\
 \text{s.t.} & 0.3x_1 + 0.4x_2 + 0.2x_3 + 1x_4 \leq 100 \\
 & 0.4x_1 + 0.2x_2 + 0.3x_3 + 1x_5 \leq 75 \\
 & x_1, x_2, x_3 \geq 0 \\
 & x_4, x_5 \leq 0
 \end{array}$$

Like all linear programs, this primal must have a dual. Applying Table 6.1, we obtain

$$\begin{array}{rllll}
 \min & + 100v_1 & + 75v_2 & & \\
 \text{s.t.} & + 0.3v_1 & + 0.4v_2 & \geq & 2 \\
 & + 0.4v_1 & + 0.2v_2 & \geq & 1.5 \\
 & + 0.2v_1 & + 0.3v_2 & \geq & 0.5 \\
 & + 1v_1 & & \leq & 9 \\
 & & + 1v_2 & \leq & 6 \\
 & & & & v_1, v_2 \geq 0
 \end{array}$$

Except for variable naming, this dual exactly matches the original primal (6.4). This will always be true.

Principle 6.30 The dual of the dual of any linear program is the LP itself.

EXAMPLE 6.19: FORMING THE DUAL OF THE DUAL

Show that the dual of the dual of the linear program in part (a) of Example 6.18 is the primal.

Solution: Assigning dual variables w_1, w_2, w_3 to the three main constraints of the dual part (a) of Example 6.18, the dual of that dual is

$$\begin{array}{rllll}
 \max & + 6w_1 & - 1w_2 & + 13w_3 & \\
 \text{s.t.} & + 3w_1 & + 1w_2 & + 2w_3 & = 7 \\
 & + 5w_1 & - 1w_2 & & \leq 6 \\
 & & + 1w_2 & + 1w_3 & \geq 2 \\
 & & & & w_1 \geq 0, w_2 \leq 0
 \end{array}$$

In accord with principle [6.30](#), this is exactly the primal (with different variable names).

6.5 COMPUTER OUTPUTS AND WHAT IF CHANGES OF SINGLE PARAMETERS

All the machinery for quantitative sensitivity analysis is now in place. It is time to do some.

CFPL Example Primal and Dual

We illustrate with the CFPL model formulated in Section 4.3 to plan operations of a plywood factory. Table 6.2 displays the full primal in terms of the decision variables

$$\begin{array}{ll}
 w_{q,v,t} \triangleq & \text{number of logs of quality } q \text{ bought from vendor } \\
 & v \text{ and peeled into green veneer of thickness } t \text{ per} \\
 & \text{month} \\
 x_{t,g} \triangleq & \text{number of square feet of grade } g \text{ green veneer of} \\
 & \text{thickness } t, \text{ purchased directly per month}
 \end{array}$$

$y_{t,g,g'}$ \triangleq number of sheets of thickness t used as grade g' veneer after drying and processing from grade g green veneer per month

$z_{t,g,g'}$ \triangleq number of sheets of front veneer grade g and back veneer grade g' plywood of thickness t , pressed and sold per month

A dual variable v_i has been assigned at the extreme right of each constraint. The optimal solution value is \$484,879 per month.

TABLE 6.2 CFPL Application Primal

maximize:		
(log costs)		
$-340w_{G,1,1/16} - 190w_{F,1,1/16} - 490w_{G,2,1/16} - 140w_{F,2,1/16}$		
$-340w_{G,1,1/8} - 190w_{F,1,1/8} - 490w_{G,2,1/8} - 140w_{F,2,1/8}$		
(green veneer costs)		
$-1.00x_{1/16,A} - 0.30x_{1/16,B} - 0.10x_{1/16,C} - 2.20x_{1/8,A} - 0.60x_{1/8,B} - 0.20x_{1/8,C}$		
(finished plywood sales)		
$+45z_{1/4,A,B} + 40z_{1/4,A,C} + 33z_{1/4,B,C} + 75z_{1/2,A,B} + 65z_{1/2,A,C} + 50z_{1/2,B,C}$		
subject to:		
(log availability)		
$w_{G,1,1/16} + w_{G,1,1/8}$	≤ 200	v_1
$w_{F,1,1/16} + w_{F,1,1/8}$	≤ 300	v_2
$w_{G,2,1/16} + w_{G,2,1/8}$	≤ 100	v_3
(purchased green veneer availability)		
$w_{F,2,1/16} + w_{F,2,1/8}$	≤ 1000	v_4
$x_{1/16,A}$	≤ 5000	v_5
$x_{1/16,B}$	$\leq 25,000$	v_6
$x_{1/16,C}$	$\leq 40,000$	v_7
$x_{1/8,A}$	$\leq 10,000$	v_8
$x_{1/8,B}$	$\leq 40,000$	v_9
$x_{1/8,C}$	$\leq 50,000$	v_{10}
(plywood market limits)		
$z_{1/4,A,B}$	≤ 1000	v_{11}
$z_{1/4,A,C}$	≤ 4000	v_{12}
$z_{1/4,B,C}$	≤ 8000	v_{13}
$z_{1/2,A,B}$	≤ 1000	v_{14}
$z_{1/2,A,C}$	≤ 5000	v_{15}
$z_{1/2,B,C}$	≤ 8000	v_{16}
(plywood pressing capacity)		
$0.25z_{1/4,A,B} + 0.25z_{1/4,A,C} + 0.25z_{1/4,B,C} + 0.40z_{1/2,A,B} + 0.40z_{1/2,A,C} + 0.40z_{1/2,B,C}$	≤ 4500	v_{17}
(green veneer balance)		
$400w_{G,1,1/16} + 200w_{F,1,1/16} + 400w_{G,2,1/16} + 200w_{F,2,1/16} + x_{1/16,A} - 35y_{1/16,A,A} - 35y_{1/16,A,B}$	≥ 0	v_{18}
$700w_{G,1,1/16} + 500w_{F,1,1/16} + 700w_{G,2,1/16} + 500w_{F,2,1/16} + x_{1/16,B} - 35y_{1/16,B,A} - 35y_{1/16,B,B} + 35y_{1/16,B,C}$	≥ 0	v_{19}

(continued)

TABLE 6.2 Continued

$900w_{G,1,1/16} + 1300w_{F,1,1/16} + 900w_{G,2,1/16} + 1300w_{F,2,1/16} + x_{1/16,C}$ $- 35y_{1/16,C,B} - 35y_{1/16,C,C}$	≥ 0	v_{20}
$200w_{G,1,1/8} + 100w_{F,1,1/8} + 200w_{G,2,1/8} + 100w_{F,2,1/8} + x_{1/8,A}$ $- 35y_{1/8,A,A} - 35y_{1/8,A,B}$	≥ 0	v_{21}
$350w_{G,1,1/8} + 250w_{F,1,1/8} + 350w_{G,2,1/8} + 250w_{F,2,1/8} + x_{1/8,B}$ $- 35y_{1/8,B,A} - 35y_{1/8,B,B} - 35y_{1/8,B,C}$	≥ 0	v_{22}
$450w_{G,1,1/8} + 650w_{F,1,1/8} + 450w_{G,2,1/8} + 650w_{F,2,1/8} + x_{1/8,C}$ $- 35y_{1/8,C,B} - 35y_{1/8,C,C}$	≥ 0	v_{23}
(finished veneer sheet balance)		
$y_{1/16,A,A} + y_{1/16,B,A} - z_{1/4,A,B} - z_{1/4,A,C} - z_{1/2,A,B} - z_{1/2,A,C}$	$= 0$	v_{24}
$y_{1/16,A,B} + y_{1/16,B,B} + z_{1/16,C,B} - z_{1/4,A,B} - z_{1/4,B,C} - z_{1/2,A,B} - z_{1/2,B,C}$	$= 0$	v_{25}
$y_{1/16,B,C} + y_{1/16,C,C} - z_{1/4,A,C} - z_{1/4,B,C} - z_{1/2,A,C} - z_{1/2,B,C}$	$= 0$	v_{26}
$y_{1/8,A,B} + y_{1/8,B,B} + z_{1/8,C,B} - z_{1/2,A,B} - z_{1/2,A,C} - z_{1/2,B,C}$	$= 0$	v_{27}
$y_{1/8,B,C} + y_{1/8,C,C} - z_{1/4,A,B} - z_{1/4,A,C} - z_{1/4,B,C} + 2z_{1/2,A,B}$ $+ 2z_{1/2,A,C} + 2z_{1/2,B,C}$	$= 0$	v_{28}
(nonnegativity)		
all variables ≥ 0		

Table 6.3 provides the corresponding dual. Key points in its derivation include:

- The dual minimizes because the primal maximizes. Objective function coefficients come directly from the primal right-hand side.
- There is one main constraint for each of the 32 primal variables. All such constraints have the \geq form because all primal variables are nonnegative. Dual right-hand sides come from the objective function coefficients of corresponding primal variables. Dual variables weighted on the left-hand side are those for constraints where the associated primal variable has nonzero coefficients.
- The first 17 dual variables are nonnegative because they correspond to \leq constraints in a maximize primal. The next 6 are nonpositive because they relate to \geq 's, and the last 5 are unrestricted because they correspond to $=$'s.

Constraint Sensitivity Outputs

No two linear programming codes are identical, but all report the primal optimal solution, the corresponding dual solution, and some related sensitivity information:

Typ	Whether the constraint is of L = \leq , or G = \geq , or E = equality form
Optimal Dual	The optimal value of the dual variable for the constraint (AMPL dual)
RHS Coef	The specified right-hand side for the constraint (AMPL current)
Slack	The amount of slack in the constraint at primal optimality (AMPL slack)
Lower Range	The lowest right-hand-side value for which the optimal dual solution must remain unchanged (AMPL down)
Upper Range	The highest right-hand-side value for which the optimal dual solution must remain unchanged (AMPL up)

The following modified version of the Two Crude Example model (6.4) AMPL encoding in Table 2.3 shows how these values are requested in that modeling language using the CPLEX solver. It is only necessary to add new display commands

TABLE 6.3 CFPL Example Dual

minimize:

$$200v_1 + 300v_2 + 100v_3 + 1000v_4 + 5000v_5 + 25,000v_6$$

$$+ 40,000v_7 + 10,000v_8 + 40,000v_9 + 50,000v_{10} + 1000v_{11} + 4000v_{12}$$

$$+ 8000v_{13} + 1000v_{14} + 5000v_{15} + 8000v_{16} + 4500v_{17}$$

subject to:

(w-variable columns)

$$v_1 + 400v_{18} + 700v_{19} + 900v_{20} \cong -340$$

$$v_2 + 200v_{18} + 500v_{19} + 1300v_{20} \cong -190$$

$$v_3 + 400v_{18} + 700v_{19} + 900v_{20} \cong -490$$

$$v_4 + 200v_{18} + 500v_{19} + 1300v_{20} \cong -140$$

$$v_1 + 200v_{21} + 350v_{22} + 450v_{23} \cong -340$$

$$v_2 + 100v_{21} + 250v_{22} + 650v_{23} \cong -190$$

$$v_3 + 200v_{21} + 350v_{22} + 450v_{23} \cong -490$$

$$v_4 + 100v_{21} + 250v_{22} + 650v_{23} \cong -140$$

(x-variable columns)

$$v_5 + v_{18} \cong -1.00$$

$$v_6 + v_{19} \cong -0.30$$

$$v_7 + v_{20} \cong -0.10$$

$$v_8 + v_{21} \cong -2.2$$

$$v_9 + v_{22} \cong -0.60$$

$$v_{10} + v_{23} \cong -0.20$$

(y-variable columns)

$$v_{24} - 35v_{18} \cong 0$$

$$v_{24} - 35v_{19} \cong 0$$

$$v_{25} - 35v_{18} \cong 0$$

$$v_{25} - 35v_{19} \cong 0$$

$$v_{25} - 35v_{20} \cong 0$$

$$v_{26} - 35v_{19} \cong 0$$

$$v_{26} - 35v_{20} \cong 0$$

$$v_{27} - 35v_{21} \cong 0$$

$$v_{27} - 35v_{22} \cong 0$$

$$v_{27} - 35v_{23} \cong 0$$

$$v_{28} - 35v_{22} \cong 0$$

$$v_{28} - 35v_{23} \cong 0$$

(z-variable columns)

$$v_{11} + 0.25v_{17} - v_{24} - v_{25} - v_{28} \cong 45$$

$$v_{12} + 0.25v_{17} - v_{24} - v_{26} - v_{28} \cong 40$$

$$v_{13} + 0.25v_{17} - v_{25} - v_{26} - v_{28} \cong 33$$

$$v_{14} + 0.40v_{17} - v_{24} - v_{25} - v_{27} - 2v_{28} \cong 75$$

$$v_{15} + 0.40v_{17} - v_{24} - v_{26} - v_{27} - 2v_{28} \cong 65$$

$$v_{16} + 0.40v_{17} - v_{25} - v_{26} - v_{27} - 2v_{28} \cong 50$$

(sign restrictions)

$$v_1, v_2, \dots, v_{17} \geq 0$$

$$v_{18}, v_{19}, \dots, v_{23} \leq 0$$

$$v_{24}, v_{25}, \dots, v_{28} \text{ unrestricted}$$

after the `solve` referencing the names of main constraints and the AMPL suffix shown above.

```
var x1 >= 0; # decision variables and types
var x2 >= 0;
minimize tcost: 100*x1+75*x2; # objective function
subject to # main constraints
gas: 0.3*x1+0.4*x2 >= 2.0;
jet: 0.4*x1+0.2*x2 >= 1.5;
lubr: 0.2*x1+0.3*x2 >= 0.5;
saudi: x1 <= 9;
venez: x2 <= 6;
option solver cplex; # choose and call solver
solve;
display cost,x1,x2; # report primal optimum
# added displays for constraint sensitivity
display gas.dual,gas.slack,gas.down,gas.current,gas.up;
display jet.dual,jet.slack,jet.down,jet.current,jet.up;
display lubr.dual,lubr.slack,lubr.down,lubr.current,lubr.up;
display saudi.dual,saudi.slack,saudi.down,saudi.current,saudi.up;
display venez.dual,venez.slack,venez.down,venez.current,venez.up;
```

Table 6.4 displays corresponding output from the **GAMS**¹ modeling language for the 28 constraints of the larger CFPL Example in Table 6.2.

Note that slack and dual values conform to primal complementary slackness conditions [6.26]. Dual variables are positive only when the corresponding slack is zero.

Right-Hand-Side Ranges

The last two items for each constraint are new to our discussion. To understand their meaning, look back at Figure 6.3. We know that optimal dual variable values quantify the slope or rate of change in such relations between the right-hand side of a constraint and the overall optimal value. In the CFPL output, for instance, $v_2^* = 122.156$ implies that the optimal value will improve by \$122.156 per log increase in $RHS = 300$ of the corresponding log availability constraint.

The RHS range question is how much we could change the right-hand side before that slope would no longer apply.

Principle 6.31 **Right-hand-side ranges** in LP sensitivity outputs show the interval within which the corresponding dual variable value provides the exact rate of change in optimal value per unit change in RHS (all other data held constant).

For example, output Table 6.4 tells us that slope v_2^* would remain unchanged for any RHS in the interval [242.5, 331.972]. Outside the range, we have only qualitative principle [6.14].

The idea behind range computation is to identify the maximum change in any RHS for which the optimal basis remains (primal) feasible. In rows with slack, such as the first constraint in Table 6.4, feasibility persists exactly until the slack is

¹A. Brooke, D. Kendrick, and A. Meeraus (1988) *GAMS: A User's Guide*, Scientific Press.

TABLE 6.4 Typical Constraint Sensitivity Analysis Output for CFPL Model

Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range
c1	L	-0.000	200.000	158.727	41.273	+infin
c2	L	122.156	300.000	0.000	242.500	331.972
c3	L	-0.000	100.000	100.000	0.000	+infin
c4	L	172.156	1000.000	0.000	942.500	1031.972
c5	L	-0.000	5000.000	5000.000	0.000	+infin
c6	L	-0.000	25,000.000	25,000.000	0.000	+infin
c7	L	0.032	40,000.000	0.000	0.000	85,858.586
c8	L	-0.000	10,000.000	10,000.000	0.000	+infin
c9	L	-0.000	40,000.000	40,000.000	0.000	+infin
c10	L	0.112	50,000.000	0.000	0.000	81,971.831
c11	L	9.564	1000.000	0.000	342.857	2621.429
c12	L	4.564	4000.000	0.000	3342.857	5621.429
c13	L	-0.000	8000.000	3644.156	4355.844	+infin
c14	L	10.000	1000.000	0.000	402.597	5402.597
c15	L	-0.000	5000.000	597.403	4402.597	+infin
c16	L	-0.000	8000.000	8000.000	0.000	+infin
c17	L	51.418	4500.000	0.000	4404.622	4609.524
c18	G	-0.201	0.000	-0.000	-51,111.111	131,759.465
c19	G	-0.201	0.000	-0.000	-51,111.111	194,861.111
c20	G	-0.132	0.000	-0.000	-45,858.586	41,818.182
c21	G	-0.312	0.000	-0.000	-31,971.831	57,500.000
c22	G	-0.312	0.000	-0.000	-31,971.831	57,500.000
c23	G	-0.312	0.000	-0.000	-31,971.831	57,500.000
c24	E	-7.046	0.000	0.000	-1460.317	5567.460
c25	E	-4.610	0.000	0.000	-1310.245	1194.805
c26	E	-4.610	0.000	0.000	-1310.245	1194.805
c27	E	-10.925	0.000	0.000	-913.481	1642.857
c28	E	-10.925	0.000	0.000	-913.481	1642.857

eliminated. Thus with the original constraint one RHS of 200 permitting 158.727 units of slack, dual slope $v_1^* = 0$ applies for any RHS value at least

$$\text{RHS} - \text{slack} = 200 - 158.727 = 41.273$$

The upper range is $+\infty$ because increasing the RHS only relaxes an already slack constraint.

The computation is a bit more complex when a constraint is active, but the issue remains keeping the primal solution feasible. We omit details.

EXAMPLE 6.20: INTERPRETING RHS RANGES

Suppose that solution of the LP

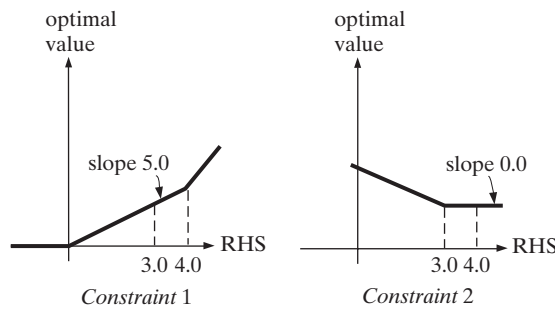
$$\begin{array}{llll}
 \min & + 5x_1 & + 9x_2 & \\
 \text{s.t.} & + 1x_1 & + 1x_2 & \geq 3 \\
 & + 1x_1 & - 1x_2 & \leq 4 \\
 & x_1, x_2 & \geq 0 &
 \end{array}$$

produces the constraint sensitivity output

Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range
c1	G	5.000	3.000	-0.000	0.000	4.000
c2	L	0.000	4.000	1.000	3.000	+infin

Sketch what can be deduced from this output about how the optimal value varies with each RHS.

Solution: What we can know about how optimal value varies with each RHS is summarized in the following two plots:



The printout’s RHS range for constraint 1 shows that the rate of change in optimal value with its right-hand side is dual variable value 5.0 in the range 0.0 to 4.0 (principle [6.30]). Above that range qualitative principle [6.14] implies that the rate can only increase because we are tightening a \geq constraint. Below the range the rate can only decrease.

For constraint 2 the RHS range is $[3.0, +\infty)$. Within this range there will be no change in optimal value because the corresponding dual variable is 0.0. Below the range the tightening \leq may produce a steeper rate of change (principle [6.14]).

Constraint What If’s

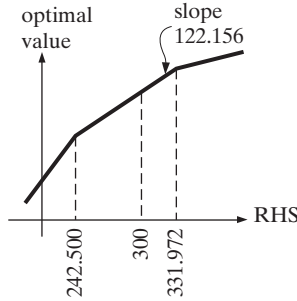
There are hundreds of possible “what if” questions about the CFPL model that can be asked and answered with the aid of results in Table 6.4. Much can be said if we consider changing only one coefficient while holding all others constant. We illustrate with a few examples.

- Question: How sensitive is the optimal value to our 8000 estimate of the market for $\frac{1}{2}$ -inch BC plywood?

Solution: The market constraint for $\frac{1}{2}$ -inch BC has optimal dual $v_{16}^* = 0$ and RHS range $[0, +\infty)$. Thus the optimal value will remain unchanged no matter what (nonnegative) market estimate we employ.

- Question: How sensitive is the optimal value to our 300 estimate of the supply of “fair” logs available from supplier 1?

Solution: The availability constraint for fair logs from supplier 1 has optimal dual $v_2^* = 122.156$ and the RHS range [242.5, 331.972]. Thus, if the estimate of 300 is too high, every log deducted will reduce the optimal value by at least \$122.156. If it is too low, every log added will increase the optimal value by at most the same amount.



The \$122.156 rate is exact within the RHS range. Below the range, the rate of change will become steeper because we are tightening a constraint (principle 6.14). Above the range, the rate will diminish.

- Question: What is the marginal value of plywood pressing capacity?

Solution: The marginal value is the optimal dual on the pressing capacity constraint, $v_{17}^* = \$51.418$ per sheet.

- Question: How much should we be willing to spend in promotional costs to increase the market for $\frac{1}{4}$ -inch AB plywood from 1000 to 2000 sheets per month?

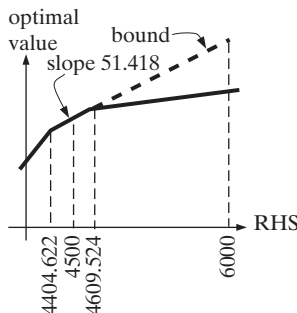
Solution: Because 2000 is within the RHS range on the market constraint for $\frac{1}{4}$ -inch AB, dual $v_{11}^* = \$9.564$ gives the exact rate of change. An increase from 1000 to 2000 would be worth

$$\begin{aligned} (\text{new value} - \text{current RHS}) v_{11}^* &= (2000 - 1000)(9.564) \\ &= \$9564 \text{ per month} \end{aligned}$$

Any promotional expense up to that amount would be justified.

- Question: How much should we be willing to pay to increase plant pressing capacity from 4500 to 6000 sheets per month?

Solution: New value 6000 falls well beyond the upper range limit of 4609.624 on the pressing capacity RHS, thus we can only bound the value of the proposed capacity increase.



At a minimum, we know that the optimal value would improve:

$$\begin{aligned} (\text{range limit} - \text{current RHS}) v_{17}^* &= (4609.624 - 4500)(51.418) \\ &\approx \$5637 \text{ per month} \end{aligned}$$

The dual provides an exact rate of change through the upper range limit.

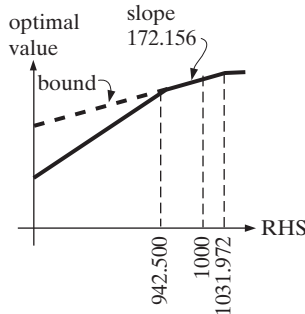
Above the range limit, the rate of change may decline (principle 6.14); we are relaxing a constraint. Still, the gain from raising capacity to 6000 cannot exceed the estimate provided by the dual-variable value:

$$\begin{aligned} (\text{new value} - \text{current RHS}) v_{17}^* &= (6000 - 4500)(51.418) \\ &= \$77,127 \text{ per month} \end{aligned}$$

We conclude that any equivalent monthly capacity expansion cost up to \$5637 could be justified and any over \$77,127 could not. In between, we cannot be definitive.

- Question: How much loss would we incur if we stopped using “good” logs from supplier 2?
Solution: The dual variable on supplier 2 good log availability has $v_3^* = 0.0$ and RHS range $[0.0, +\infty)$. Thus no change in optimal value would result if we reduced availability all the way to zero.
- Question: How much loss would we experience if we stopped using fair logs from supplier 2?

Solution: The dual variable on supplier 2 fair log availability 1000 has $v_4^* = 172.156$ and RHS range $[942.5, 1031.972]$. Eliminating such log purchases amount to changing the RHS to zero. For such a value, which is well outside the RHS range, we can only bound the effect.



We are tightening the availability constraint, so principle 6.14 shows that the true rate of change will be v_4^* (actually, its negative) or worse. Thus, closing out supplier 2 fair logs would produce a loss of at least

$$\begin{aligned} (\text{new value} - \text{current RHS}) v_3^* &= (0 - 1000)(172.156) \\ &= -\$172,156 \text{ per month} \end{aligned}$$

There is no way to use Table 6.4 information to set an outer limit on the loss (except the full optimal value, \$484,878).

Variable Sensitivity Outputs

We saw in Section 6.1 that sensitivity analysis encompasses much more than just changes in right-hand-side values. Table 6.5 provides an example of the variable

TABLE 6.5 Typical Variable Sensitivity Analysis Output for CFPL Model

Name	Optimal Value	Bas Sts	Lower Bound	Upper Bound	Object Coef	Reduced Object	Lower Range	Upper Range
wG1s	41.273	BAS	0.000	+infin	-340.000	0.000	-361.569	-304.762
wF1s	300.000	BAS	0.000	+infin	-190.000	0.000	-190.000	+infin
wG2s	0.000	NBL	0.000	+infin	-490.000	150.000	-infin	-340.000
wF2s	155.273	BAS	0.000	+infin	-140.000	0.000	-185.306	-140.000
wG1e	0.000	NBL	0.000	+infin	-340.000	27.844	-infin	-312.156
wF1e	0.000	NBL	0.000	+infin	-190.000	0.000	-infin	-190.000
wG2e	0.000	NBL	0.000	+infin	-490.000	177.844	-infin	-312.156
wF2e	844.727	BAS	0.000	+infin	-140.000	0.000	-140.000	-94.694
xsA	0.000	NBL	0.000	+infin	-1.000	0.799	-infin	-0.201
xsB	0.000	NBL	0.000	+infin	-0.300	0.099	-infin	-0.201
xsC	40,000.000	BAS	0.000	+infin	-0.100	0.000	-0.132	+infin
xeA	0.000	NBL	0.000	+infin	-2.200	1.888	-infin	-0.312
xeB	0.000	NBL	0.000	+infin	-0.600	0.288	-infin	-0.312
xeC	50,000.000	BAS	0.000	+infin	-0.200	0.000	-0.312	+infin
zqAB	1000.000	BAS	0.000	+infin	45.000	0.000	35.436	+infin
zqAC	4000.000	BAS	0.000	+infin	40.000	0.000	35.436	+infin
zqBC	4355.844	BAS	0.000	+infin	33.000	0.000	31.612	34.675
zhAB	1000.000	BAS	0.000	+infin	75.000	0.000	65.000	+infin
zhAC	4402.597	BAS	0.000	+infin	65.000	0.000	62.320	67.220
zhBC	0.000	NBL	0.000	+infin	50.000	12.564	-infin	62.564
yeBB	0.000	NBL	0.000	+infin	-0.000	0.000	-infin	0.000
ysAA	3073.247	BAS	0.000	+infin	0.000	0.000	-2.351	5.045
ysAB	0.000	NBL	0.000	+infin	0.000	2.436	-infin	2.436
ysBA	7329.351	BAS	0.000	+infin	0.000	0.000	-2.792	3.964
ysBB	0.000	NBL	0.000	+infin	0.000	2.436	-infin	2.436
ysBC	0.000	NBL	0.000	+infin	0.000	2.436	-infin	2.436
ysCB	6355.844	BAS	0.000	+infin	0.000	0.000	-1.388	1.675
ysCC	12,758.442	BAS	0.000	+infin	0.000	0.000	-3.700	4.467
yeAB	2413.506	BAS	0.000	+infin	0.000	0.000	-12.867	15.857

or column-oriented sensitivity information that is part of almost any LP optimizer output. The following are displayed for each primal variable:

Item	AMPL
Name	The name of the primal variable
Optimal Value	The optimal value for the variable
Bas Sts	Whether the variable is BAS = basic, NBL = nonbasic lower-bounded, or NBU = nonbasic upper-bounded in the optimal solution (see Section 5.9)
Lower Bound	The specified lower bound on the variable
Upper Bound	The specified upper bound on the variable
Object Coef	The specified objective function coefficient for the variable (AMPL current)

(continued)

Item	AMPL
Reduced Object	The reduced objective function coefficient for the variable at optimality (AMPL rc)
Lower Range	The lowest objective function coefficient value for which the optimal primal solution must remain unchanged (AMPL down)
Upper Range	The highest objective function coefficient value for which the optimal primal solution must remain unchanged (AMPL up)

Most of these items are familiar from our previous discussion. Optimal values, objective function coefficients, and reduced objective function coefficients have usually been denoted x_j^* , c_j , and \bar{c}_j , respectively. Notice that given values satisfy dual complementary slackness conditions [6.27]. Whenever `Optimal Value` > 0, the corresponding dual constraint slack `Reduced Object` = 0.

Just as the constraint sensitivity information shown above, AMPL display commands are easily added to capture new results about objective function coefficients from that modeling system and the CPLEX solver. Parallel with the earlier case, the items are identified by a model variable name followed by the AMPL suffix from the above list. For the 2 primal variables of the Two-Crude example those additions would be as follows:

```
display x1.current,x1.down,x1.up,x1.rc;
display x2.current,x2.down,x2.up,x1.rc;
```

Objective Coefficient Ranges

As with constraint information, the two completely new items are **ranges**, this time on the objective. Items `Lower Range` and `Upper Range` delimit changes in objective coefficient value that would leave the primal optimal solution unchanged.

Figure 6.5 illustrates how keeping the same optimal solution does not necessarily mean keeping the same optimal value. The optimal level for sales variable $z_{1/4,A,B}$ (`zqAB` in the computer output) is 1000. Thus the optimal objective value

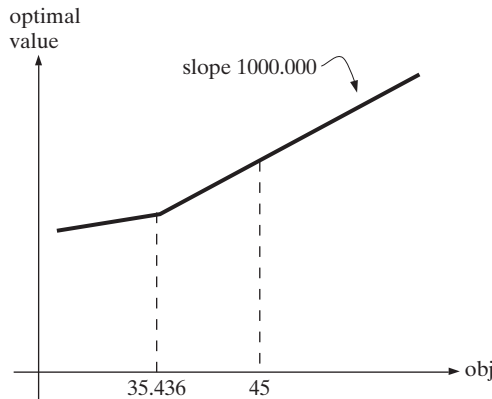


FIGURE 6.5 Optimal Value Impact of CFPL Objective Changes

will increase at the rate of \$1000 per \$1 increase in the $z_{1/4, A, B}$ objective function coefficient as long as the primal optimal solution remains unchanged.

Principle 6.32 | Optimal primal LP variable values show the rate of change in optimal value per unit increase in the corresponding objective function coefficient.

Principle 6.33 | **Objective coefficient ranges** in LP sensitivity outputs display the interval within which the current primal solution remains optimal and individual variable values continue to show exactly how optimal value changes with their objective coefficients (all other data held constant).

In Figure 6.5 the range is $[35.436, +\infty)$. Outside the range we must rely on qualitative principle [6.16](#) to impute rates of change. Since selling prices below \$35.436 help the objective less, the rate ≤ 1000 .

Like the RHS case, objective coefficient range computations involve details inappropriate for this book. However, calculations center on keeping the dual solution feasible so that optimality conditions will continue to hold for the current primal optimum.

The computation is easy when dual slack \bar{c}_j is nonzero. Certainly, dual feasibility will not be lost until all slack is eliminated. For instance, sales variable $z_{1/2, B, C}$ (zhBC), with reduced objective coefficient $-\$12.564$ on an original coefficient of \$50, has objective coefficient range $(-\infty, 62.564]$. Reductions from \$50 only relax an already slack dual constraint, and increases eliminate slack only at the coefficient value

$$\begin{aligned} (\text{current coefficient} - \text{dual slack}) &= 50 - (-12.564) \\ &= \$62.564 \text{ per month} \end{aligned}$$

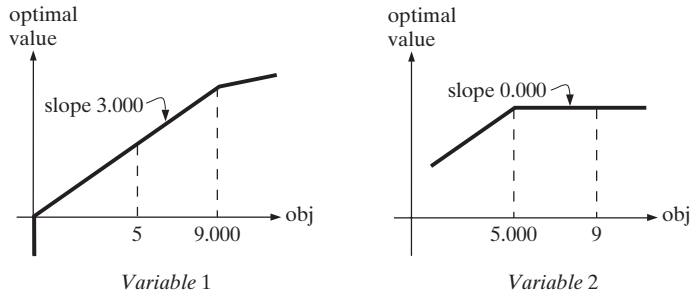
EXAMPLE 6.21: INTERPRETING OBJECTIVE COEFFICIENT RANGES

Return to the LP of Example 6.20, and assume that the variable part of sensitivity output is as follows:

Name	Optimal Value	Bas Sts	Object Coef	Reduced Object	Lower Range	Upper Range
x1	5.000	BAS	5.000	0.000	0.000	9.000
x2	0.000	NBL	9.000	4.000	5.000	+infin

Sketch what can be deduced from this output about how the optimal value will change with the two objective function coefficients.

Solution: What we can know about how the optimal value varies with objective function coefficients is summarized in the following two plots:



The range on the first coefficient is $[0.0, 9.0]$, meaning that the current primal solution will remain optimal for any c_1 in the range and that the optimal value varies within that range at rate $x_1^* = 3.0$ (principle 6.33). Increasing that coefficient hurts the optimal value, so the rate may decline for $c_1 > 9.0$ (principle 6.16). Below the range the rate becomes steeper.

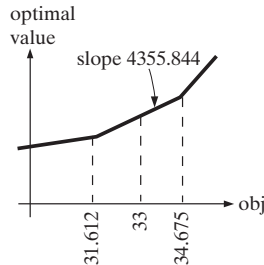
Similarly, the output above shows that the range for c_2 is $[5.0, +\infty)$. Any cost in that range will leave $x_2^* = 0.0$ and the optimal value unchanged. Below the range the objective may decline because helping the objective will steepen the rate (principle 6.16).

Variable What If's

The value of output information in Table 6.5 is best illustrated by considering some of the “what if” questions that it might help to answer. Again we assume only one parameter changes with all else held constant.

- Question: How sensitive are the optimal solution and value to our \$33 estimate of the selling price for $\frac{1}{4}$ -inch BC plywood?

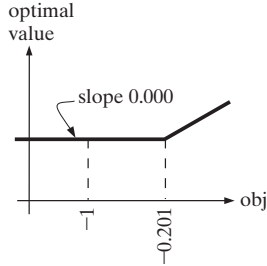
Solution: The objective range of $[31.612, 34.675]$ on z_{qBC} implies that the optimal operating plan would remain unchanged for any price between those values. However, the optimal value would vary.



Every price increase of \$1 would add $z_{1/4, B, C}^* = \$4355.844$ per week to the optimal value within the range and that much or more above (principle 6.16). Every \$1 decrease would reduce the optimal value at the same rate through limit \$31.612 and that much or less thereafter.

- Question: How sensitive are the optimal solution and value to our \$1.00 per square foot (objective coefficient -1.00) estimate of the cost of $\frac{1}{16}$ -inch A green veneer?

Solution: Coefficient range $(-\infty, -0.201]$ on x_{SA} implies that the optimal solution would remain unchanged for any coefficient below -0.201 (cost above \$0.201) per square foot. Within that range, optimal value would also remain unchanged because $x_{1/16, A}^* = 0$.

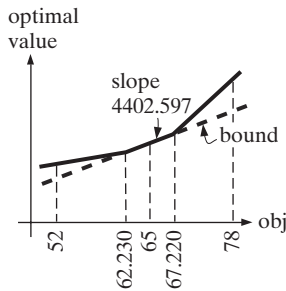


Coefficient values above -0.201 (costs below \$0.201) might improve the optimal value because the rate of change cannot decrease.

- Question: How much difference in the optimal value would it make if we increased or decreased the \$65 per sheet selling price of $\frac{1}{2}$ -inch AC plywood by 20%?

Solution: A 20% increase would bring the price to $(1.2)(65) = \$78$. Because this value is well beyond the upper range limit of \$67.220 for variable z_{hAC} , we can only bound the optimal value impact. It will equal or exceed

$$\begin{aligned} (\text{new coefficient} - \text{original coefficient}) z_{1/2,A,C}^* &= (78 - 65)(4402.597) \\ &\approx \$57,234 \text{ per month} \end{aligned}$$



A 20% price decrease also takes us well outside the range. We can compute the least possible impact by considering only the range where we know the exact rate of change.

$$\begin{aligned} (\text{range limit} - \text{original coefficient}) z_{1/2,A,C}^* &= (62.320 - 65)(4402.597) \\ &\approx -\$11,799 \text{ per month} \end{aligned}$$

Extending the current rate of change all the way to $(0.8)(65) = \$52$ bounds the maximum impact:

$$\begin{aligned} (\text{new coefficient} - \text{original coefficient}) z_{1/2,A,C}^* &= (52 - 65)(4402.597) \\ &\approx -\$57,234 \text{ per month} \end{aligned}$$

- Question: The current optimal plan produces no $\frac{1}{2}$ -inch BC plywood at our estimated \$50 per sheet selling price. At what price would producing that product become profitable?

Solution: Coefficient ranges that indicate $z_{1/2,B,C}^* = 0$ would remain the optimal choice for any price up to \$62.564. At prices above that value a positive profit is possible because the rate of change can only increase as we move a coefficient to help the objective (principle [6.16](#)).

- Question: The current optimal plan buys $x_{1/16,C}^* = 40,000$ square feet of $\frac{1}{16}$ -inch C green veneer per week at cost \$0.10 (objective coefficient -0.10) per square foot. At what price would such purchases become unprofitable?

Solution: The coefficient range for $x_{1/16,C}$ is $[-0.132, +\infty)$. Thus we know that the 40,000-square foot value will remain optimal at coefficients above -0.132 (i.e., cost below \$0.132 per square foot). Beyond \$0.132 it may be optimal to buy less.

Dropping and Adding Constraint What If's

Another familiar form of “what if” questions involve dropping or adding constraints. Results in Figures 6.4 and 6.5 help here, too. The necessary insights are almost obvious:

Principle 6.34 Dropping a constraint can change the optimal solution only if the constraint is active at optimality.

Principle 6.35 Adding a constraint can change the optimal solution only if that optimum violates the constraint.

Some examples (assuming the rest of the model is held constant):

- Question: Would dropping the press capacity constraint change the optimal plan?
Solution: Yes. The constraint has no slack at the optimal solution.
- Question: Would dropping the supplier 1 good log availability constraint change the optimal plan?
Solution: No. At the optimal solution that constraint has a slack of 158.727 logs.
- Question: We have an informal commitment to buy at least 325 logs per month from supplier 1. Would including that constraint change the optimal solution?
Solution: No. Such a constraint would have the form

$$w_{G,1,1/16} + w_{G,1,1/8} + w_{F,1,1/16} + w_{F,1,1/8} \geq 325$$

Substituting the current optimal solution yields

$$\begin{aligned} w_{G,1,1/16}^* + w_{G,1,1/8}^* + w_{F,1,1/16}^* + w_{F,1,1/8}^* &= 41.273 + 0 + 300 + 0 \\ &= 341.273 \\ &> 325 \end{aligned}$$

The constraint would not be active if it were included.

- Question: Would a policy limiting green veneer purchase to \$10,000 per month change the current optimal plan?
Solution: Yes. A new constraint enforcing this policy would have the form

$$1.00x_{1/16,A} + 0.30x_{1/16,B} + 0.10x_{1/16,C} + 2.20x_{1/8,A} + 0.60x_{1/8,B} + 0.20x_{1/8,C} \leq 10,000$$

Substituting the current plan yields

$$\begin{aligned} 1.00x_{1/16,A}^* + 0.30x_{1/16,B}^* + 0.10x_{1/16,C}^* + 2.20x_{1/8,A}^* + 0.60x_{1/8,B}^* + 0.20x_{1/8,C}^* \\ = 1.00(0) + 0.30(0) + 0.10(40,000) + 2.20(0) + 0.60(0) + 0.20(50,000) \\ = 14,000 \\ \neq 10,000 \end{aligned}$$

Our current optimal solution violates the proposed constraint.

EXAMPLE 6.22: ANALYZING DROPPED AND ADDED CONSTRAINTS

Suppose that a linear program has optimal solution $x_1^* = 3$, $x_2^* = 0$, $x_3^* = 1$. Determine whether each of the following modifications in the model will change this optimal solution.

- (a) Dropping constraint $6x_1 - x_2 + 2x_3 \geq 20$
- (b) Dropping constraint $4x_1 - 3x_3 \leq 15$
- (c) Adding constraint $x_1 + x_2 + x_3 \leq 2$
- (d) Adding constraint $2x_1 + 7x_2 + x_3 \leq 7$

Solution:

- (a) Substituting the optimal solution gives us

$$6(3) - (0) + 2(1) = 20$$

so that the constraint is active. Under principle [6.34](#), dropping the constraint may change the optimal solution.

- (b) Substituting the optimal solution, we have

$$4(3) - 3(1) = 9 < 15$$

and the constraint is inactive. Dropping the constraint will not change the optimal solution (principle [6.34](#)).

- (c) Substituting the optimal solution, we have

$$(3) + (0) + (1) = 4 \not\leq 2$$

Since the constraint is violated, adding it will change the optimal solution (principle [6.35](#)).

- (d) Substituting the optimal solution, we have

$$2(3) + 7(0) + (1) = 7 \leq 7$$

Adding the constraint will not change the optimal solution because it is already satisfied (principle [6.35](#)).

Dropping and Adding Variable What If's

The final category of “what if” questions that we can answer using Figures 6.4 and 6.5 involves dropping and adding variables. The dropping case is trivial:

Principle 6.36 | An LP variable can be dropped without changing the optimal solution only if its optimal value is zero.

For example, Table 6.5 implies that the optimal plan would remain unchanged if we eliminated $\frac{1}{2}$ -inch BC producing activity $z_{1/2,B,C}$. Its optimal value $z_{1/2,B,C}^* = 0$.

Adding is a bit more complex. We need to decide whether the optimal value for an added variable would be zero, implying that it is just as well not to include it. But this depends on whether current optimal dual prices v_i^* produce a reduced objective function coefficient with the proper sign (i.e., whether the main dual constraint associated with the new variable is satisfied).

Principle 6.37 A new LP variable can change the current primal optimal solution only if its dual constraint is violated by the current dual optimum.

Some examples (assuming the rest of the model is held constant):

- Question: A new $\frac{1}{4}$ -inch AA plywood may be introduced that could be pressed in 0.25 hour using two $\frac{1}{16}$ -inch sheets of A veneer and one $\frac{1}{8}$ -inch sheet of B veneer. At what selling price would this product enter the optimal plan?

Solution: The corresponding dual constraint would be

$$0.25v_{17} - 2v_{24} - v_{27} \geq c$$

Substituting the current dual solution yields

$$\begin{aligned} 0.25v_{17}^* - 2v_{24}^* - v_{27}^* &= 0.25(51.418) - 2(-7.046) - (-10.925) \\ &\approx \$37.87 \end{aligned}$$

At any price above this value, the new product would be profitable.

- Question: A new supplier offers $\frac{1}{16}$ -inch B green veneer at \$0.40 per square foot. Would adding the possibility of purchasing this veneer change the optimal solution?

Solution: No. The only nonzero coefficient on this new activity would appear in the balance constraint for $\frac{1}{16}$ -inch B veneer. Thus the constraint would be

$$v_{19} \geq -0.40$$

The present $v_{19}^* = -0.201$ satisfies this constraint, so it would be optimal to use none of the new activity.

EXAMPLE 6.23: ANALYZING DROPPED AND ADDED VARIABLES

Suppose that a minimize LP over nonnegative variables has primal optimal solution $x_1 = 0, x_2 = 4, x_3 = 2$ and corresponding dual optimal solution $v_1 = 1, v_2 = 0, v_3 = 6$. Determine whether each of the following modifications in the model will change this optimal solution.

- Dropping variable x_1
- Dropping variable x_2
- Adding a variable with coefficients 2, -1 , and 3 in the main constraints and cost 18
- Adding a variable with coefficients 1, 0, and 2 in the main constraints and cost 50

Solution:

(a) In accord with principle [6.36], dropping x_1 will not change the optimal solution because its optimal value = 0.

(b) Since x_2 has a nonzero optimal value, dropping it would change the optimal solution (principle [6.36]).

(c) The main dual constraint for the new primal variable would be

$$2v_1 - 1v_2 + 3v_3 \leq 18$$

Substituting the optimal dual solution, we obtain

$$2(1) - 1(0) + 3(6) = 20 \not\leq 18$$

Adding this variable would change the primal optimum because the current dual solution violates the new constraint (principle [6.37]).

(d) The main dual constraint for this new primal variable would be

$$1v_1 + 2v_3 \leq 50$$

Substituting the optimal dual solution, we obtain

$$1(1) + 2(6) = 13 \leq 50$$

Since the constraint is satisfied, adding the new variable would not change the primal optimum (principle [6.37]).

6.6 BIGGER MODEL CHANGES, REOPTIMIZATION, AND PARAMETRIC PROGRAMMING

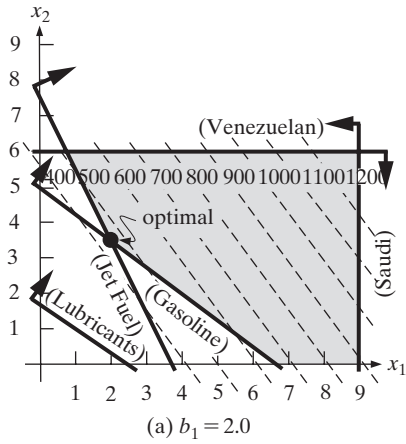
Although standard LP sensitivity outputs lend insight on a host of minor “what if” questions, optimization by-products can go only so far. In this section we review briefly some of the practical limits and introduce more informative approaches requiring reoptimization.

Ambiguity at Limits of the RHS and Objective Coefficient Ranges

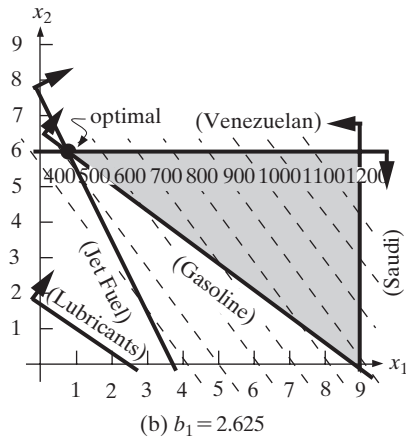
In Section 6.5 (principles [6.31] and [6.33]) we explained that the optimal primal solution provides reliable quantitative sensitivity information only as long as an objective coefficient change stays within the range provided in LP outputs, and the optimal dual solution yields precise rates of change only if a right-hand-side change is restricted to the RHS range displayed. Beyond the ranges, we have only upper or lower bounds on rates of change from qualitative principles [6.14] and [6.16].

Figure 6.6 illustrates how the ambiguity about rates of change, which we first encountered in Section 6.2, occurs at the limits of RHS ranges. Three different versions of our venerable Two Crude refining model (6.4) are solved graphically using varying gasoline requirement RHS values $b_1 = 2.0, 2.625, \text{ and } 3.25$, respectively.

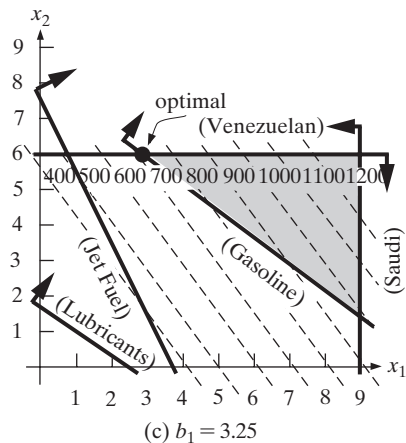
Computer output for the 2.0 case shows optimal dual slope $v_1^* = 20$ holding within the RHS range $1.125 \leq b_1 \leq 2.625$. A similar output for the $b_1 = 3.25$ choice indicates that a rate of change $v_1^* = 333.333$ applies over the range $2.625 \leq b_1 \leq 5.100$.



Optimal Dual	Lower Range	Upper Range
100.000	1.125	2.625



Optimal Dual	Lower Range	Upper Range
100.000	1.125	2.625
<i>or</i>		
333.333	2.625	5.100



Optimal Dual	Lower Range	Upper Range
333.333	2.625	5.100

FIGURE 6.6 Sensitivity Ranges for Varying Two Crude Gasoline Demand

One or the other of these dual and range outputs would print out if we solved with b_1 exactly 2.625, but which one it would be is unpredictable. Either would mislead. The correct rate for an increase from $b_1 = 2.625$ is 333.333, and for a decrease it is 100.00, but neither possible output provides both.

Principle 6.38 | At the limits of the RHS and objective function sensitivity ranges rates of optimal value change are ambiguous, with one value applying below the limit and another above. Computer outputs may show either value.

EXAMPLE 6.24: INTERPRETING SENSITIVITY AT RANGE LIMITS

Return to the Swedish Steel sensitivity plot of Figure 6.3(a). Determine from the plot the RHS ranges and optimal dual values that might result if an LP optimization code were invoked with the given RHS (a) 75.0; (b) 60.4.

Solution:

(a) RHS value 75.0 falls within an unambiguous range. Output would show dual variable -3.38 and range $[60.4, 83.3]$.

(b) RHS value 60.4 forms the boundary between two ranges. Output might show dual variable -3.38 and range $[60.4, 83.3]$, or it might produce dual -4.98 with range $[0.0, 60.4]$.

Connection between Rate Changes and Degeneracy

A closer look at the three cases in Figure 6.6 shows what causes rates to change at $b_1 = 2.625$. Below that value, the optimal solution is defined by active gasoline and jet fuel requirement constraints

$$\begin{aligned} 0.3x_1 + 0.4x_2 &\geq b_1 \\ 0.4x_1 + 0.2x_2 &\geq 1.5 \end{aligned}$$

The corresponding dual slope is $v_1^* = 100$. Above 2.625, the Venezuelan availability limit replaces jet fuel to give active set

$$\begin{aligned} 0.3x_1 + 0.4x_2 &\geq b_1 \\ x_2 &\leq 6 \end{aligned}$$

Rate $v_1^* = 333.333$.

Principle 6.39 | Rates of variation in optimal value with model constants change when the collection of active primal or dual constraints changes.

At exactly $b_1 = 2.625$ all three constraints are active, but any two define primal solution $x_1^* = 0.75$, $x_2^* = 6$. This is the degenerate case of more constraints being active than are needed to settle the primal solution that we introduced in

Section 5.6. Rates of change displayed in sensitivity outputs will reflect whichever pair the optimization search happens to focus on.

Many, perhaps most, large linear programs will have degenerate optimal solutions. It follows that we will often find that the given value of a model parameter also forms one or the other end of its range. Even if the base-case value lies strictly within a range, that interval of reliable quantitative sensitivity information is likely to be rather narrow.

Principle 6.40 Degeneracy, which is extremely common in large-scale LP models, limits the usefulness of sensitivity by-products from primal optimization because it leads to narrow RHS and objective coefficient ranges and ambiguity at the range limits.

Reoptimization to Make Sensitivity Exact

The obvious remedy for weaknesses in sensitivity analysis based on optimization byproducts is reoptimization—repeating the optimization search with changed model constants. For example, if we want to know how the optimal plan at Two Crude would change if gasoline demand increases from 2.0 to 3.5, we could just run again with the revised RHS.

Principle 6.41 If the number of “what if” variations does not grow too big, **reoptimization** using different values of model input parameters often provides the most practical avenue to good sensitivity analysis.

Practical OR analyses rely heavily on running a variety of cases, and many linear programming optimization codes provide for inputting several alternative RHSs or objective functions at the same time. Still, the approach has its limits. All combinations of even 10 parameter variations leads to $2^{10} = 1024$ cases to try; 11 variations imply twice as many.

Parametric Variation of One Coefficient

To do sensitivity analysis by trying different cases, we must, of course, know what cases to try. For example, we may know we are interested in how sensitive optimal results are to changes in an RHS or objective coefficient, yet be unclear about exactly what values to consider.

Parametric studies track the optimal value as a function of model inputs. Figures 6.3 and 6.4 displayed just such parametric functions in our qualitative discussion of Section 6.2. Figure 6.7 adds another—the Two Crude optimal value as a function of the demand for gasoline.

Many LP optimization codes include automatic features to produce such parametric analysis, but we are now in a position to see how it can be done with the tools already at hand. Just four carefully chosen runs can construct the entire Figure 6.7 curve. In particular, we employ the following cases and sensitivity outputs:

Case	RHS	Dual	Lower Range	Upper Range
Base model	2.000	20.000	1.125	2.625
Variant 1	$2.625 + \varepsilon$	333.333	2.626	5.100
Variant 2	$5.100 + \varepsilon$	$+\infty$	5.100	$+\infty$
Variant 3	$1.125 - \varepsilon$	0.000	$-\infty$	1.125

where ε is a small positive number.

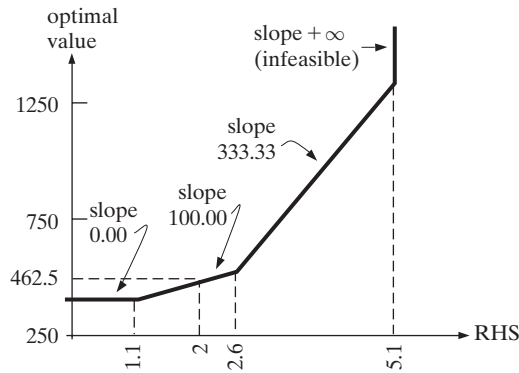


FIGURE 6.7 Parametric Variation of Optimal Value with Two Crude Gasoline Demand

Parametric analysis begins with the base model case of $b_1 = 2.000$ and its optimal value of 92.5. The first line of table values shows that the slope or rate of change in optimal value at that point is $v_1^* = 20.000$. Range information tells us that this slope holds from $b_1 = 1.125$ to 2.625. The result is the segment of the Figure 6.7 curve passing through $b_1 = 2.000$, optimal value = 92.5.

New rates of change arise only outside that range. Our first variation changes b_1 to $2.625 + \varepsilon$, just beyond the upper limit. The result is new slope 66.667 and an indication that it holds through upper limit $b_1 = 5.100$. Repeating for $b_1 = 5.100 + \varepsilon$ shows that the model is infeasible beyond 5.100. We complete the analysis by moving b_1 below the base-case range limit of 1.125. Reoptimization with $b_1 = 1.125 - \varepsilon$ indicates that the slope $v_1^* = 0.000$ will hold for all further decreases.

Principle 6.42 Parametric studies of optimal value as a function of a single-model RHS or objective function coefficient can be constructed by repeated optimization using new coefficient values just outside the previously applicable sensitivity range.

EXAMPLE 6.25: PARAMETRICALLY ANALYZING ONE COEFFICIENT

Return to the parametric plots of Figures 6.3 and 6.4.

(a) Tabulate the separate optimizations that would have to be run to produce RHS parametric analysis Figure 6.3(a), beginning from the base the RHS value 75.0.

(b) Tabulate the separate optimizations that would have to be run to produce objective coefficient parametric analysis Figure 6.4(a), beginning from base cost 9.0.

Solution:

(a) Proceeding from the figure according to principle 6.42 produces the following cases:

Case	RHS	Dual	Lower Range	Upper Range
Base model	75.0	-3.38	60.4	83.3
Variant 1	$83.3 + \epsilon$	0.00	83.3	$-\infty$
Variant 2	$60.4 + \epsilon$	-4.98	0.0	60.4
Variant 3	$0.0 - \epsilon$	$-\infty$	$-\infty$	0.0

(b) Proceeding from the figure according to principle 6.42 produces the following cases:

Case	Cost	Primal	Lower Range	Upper Range
Base model	9.0	156.1	8.4	10.1
Variant 1	$10.1 + \epsilon$	109.56	10.1	19.9
Variant 2	$19.9 + \epsilon$	82.03	19.9	29.2
Variant 3	$29.2 + \epsilon$	72.59	29.2	31.8
Variant 4	$31.8 + \epsilon$	72.21	31.8	35.5
Variant 5	$35.5 + \epsilon$	68.43	35.5	$+\infty$
Variant 6	$8.4 - \epsilon$	178.76	8.0	8.4
Variant 7	$8.0 - \epsilon$	400.98	$-\infty$	8.0

Assessing Effects of Multiple Parameter Changes

The greatest limitation of sensitivity studies we have explained so far is that all vary only one model input at a time. They increase or decrease a single RHS or objective coefficient, or add or drop a single variable or constraint. The implicit assumption is that all other data are held constant.

Computations behind sensitivity printouts like those of Figures 6.4 and 6.5 depend explicitly on this “single change” assumption.

Principle 6.43 Elementary LP sensitivity rates of change and ranges hold only for a single coefficient change, with all other data held constant.

Unfortunately, one-at-a-time analysis often does not suffice. Many “what if” questions that arise in operations research studies involve multiple changes, with several model constants varying simultaneously. For example, we may wish to know the impact on Two Crude refinery’s optimal plan if demand for gasoline increases by some percent at the same time as demand for jet fuel increases at twice that percent because of an anticipated upturn in the economy; two RHSs would be changing at the same time.

As with single variations, we can deal with a modest number of such multiple-change questions simply by reoptimizing with new coefficients. For example, a fraction θ increase in both gasoline and jet fuel demand at Two Crude would be implemented by a new run with

$$\begin{aligned} b_1 &= (1 + \theta)2.0 \\ b_2 &= (1 + 2\theta)1.5 \end{aligned} \tag{6.8}$$

and all other data as before.

Parametric Multiple-RHS Change

To see how to track the parametric effect of such multiple changes over a variety of change magnitudes θ , we need to think of right-hand sides in the form

$$b_i^{\text{new}} = b_i^{\text{base}} + \theta \Delta b_i$$

where each Δb_i shows how much its RHS b_i is changing per unit variation, and θ defines the size of the step. In the example of (6.8), change components for the five model RHSs are

$$\Delta b_1 = 2.0, \quad \Delta b_2 = 2(1.5) = 3, \quad \Delta b_3 = \Delta b_4 = \Delta b_5 = 0$$

Thinking of θ as a decision variable, adding $(\theta \Delta b_i)$ on the right-hand side of each constraint translates as $-(\Delta b_i)\theta$ on the left. Therein lies the key insight.

Principle 6.44 The effect of a multiple change in right-hand sides with step θ can be analyzed parametrically by treating θ as a new decision variable with constraint coefficients $-\Delta b_i$ that detail the rates of change in RHSs b_i and a value fixed by a new equality constraint.

For example, the revised Two Crude model with varying rate of gasoline and jet fuel demand growth would be

$$\begin{aligned} \min \quad & + 100x_1 \quad + 75x_2 \\ \text{s.t.} \quad & + 0.3x_1 \quad + 0.4x_2 \quad - 2\theta \quad \geq 2 \\ & + 0.4x_1 \quad + 0.2x_2 \quad - 3\theta \quad \geq 1.5 \\ & + 0.2x_1 \quad + 0.3x_2 \quad \geq 0.5 \\ & + 1x_1 \quad \leq 9 \\ & \quad \quad + 1x_2 \quad \leq 6 \\ & \quad \quad \quad + 1\theta \quad = b_6 \\ & x_1, x_2 \geq 0. \theta \text{ URS} \end{aligned}$$

We can now proceed exactly as we did to construct Figure 6.7. New equality constraint

$$\theta = b_6$$

reduces the task of parametric analysis to one of seeing how optimal value changes with RHS b_6 .

Using RHS range outputs for that constraint leads us to the following sequence of runs and the parametric curve of Figure 6.8.

Case	RHS	Dual	Lower Range	Upper Range
Base model	2.000	20.000	1.125	2.625
Variant 1	$2.625 + \epsilon$	66.667	2.626	5.100
Variant 2	$5.100 + \epsilon$	$+\infty$	5.100	$+\infty$
Variant 3	$1.125 - \epsilon$	0.000	$-\infty$	1.125

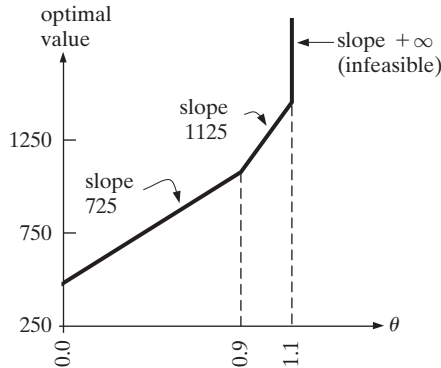


FIGURE 6.8 Parametric Multiple-RHS Change of the Two Crude Example

EXAMPLE 6.26: ANALYZING MULTIPLE-RHS CHANGES

Consider the linear program

$$\begin{aligned}
 \max \quad & + 5x_1 + 2x_2 \\
 \text{s.t.} \quad & + 1x_1 + 1x_2 \leq 3 \\
 & + 1x_1 \leq 2 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Show how to modify the model to parametrically analyze the effect of simultaneously increasing the first RHS and decreasing the second by the same amount.

Solution: The indicated change involves $\Delta b_1 = +1, \Delta b_2 = -1$. Thus, applying principle [6.44], we may parametrically analyze the change by varying new RHS b_3 in the modified model:

$$\begin{aligned}
 \max \quad & + 5x_1 + 2x_2 \\
 \text{s.t.} \quad & + 1x_1 + 1x_2 - 1\theta \leq 3 \\
 & + 1x_1 + 1\theta \leq 2 \\
 & + 1\theta = b_3 \\
 & x_1, x_2 \geq 0, \theta \text{ URS}
 \end{aligned}$$

Parametric Change of Multiple Objective Function Coefficients

To see how to do the same sort of parametric multiple-change analysis when objective function coefficients are changing, we must think back to the relation between primal and dual. Parametric revision

$$c_j^{\text{new}} = c_j^{\text{base}} + \theta \Delta c_j$$

modifies the objective coefficients of the primal, which are the right-hand sides of the dual. Thus the analog of adding a new constraint $-\Delta \mathbf{b}$ column in the RHS case is the addition of a new constraint row $-\Delta \mathbf{c}$ to vary the c_j . To be more specific,

Principle 6.45 The effect of a multiple change in objective function with step θ can be analyzed parametrically by treating objective rates of change $-\Delta c_j$ as coefficients in a new equality constraint having right-hand side zero and a new unrestricted variable with objective coefficient θ .

Again we illustrate using the Two Crude model (6.4). If we wish to explore parametrically the impact of a uniform crude price rise by a fraction θ ,

$$c_1^{\text{new}} = 100 + \theta \Delta c_1 = 20 + \theta(20)$$

$$c_2^{\text{new}} = 75 + \theta \Delta c_2 = 15 + \theta(15)$$

The revised primal model to manipulate is

$$\begin{array}{llll} \min & + 100x_1 & + 75x_2 & + \theta x_3 \\ \text{s.t.} & + 0.3x_1 & + 0.4x_2 & \geq 2 \\ & + 0.4x_1 & + 0.2x_2 & \geq 1.5 \\ & + 0.2x_1 & + 0.3x_2 & \geq 0.5 \\ & + 1x_1 & & \leq 9 \\ & & + 1x_2 & \leq 6 \\ & -20x_1 & - 15x_2 & + 1x_3 = 0 \\ & & & x_1, x_2 \geq 0, x_3 \text{ URS} \end{array}$$

Negatives of rates Δc_j form coefficients for a new equality constraint with RHS = 0.0, and new variable x_3 appears in that constraint and the objective function.

We can now determine parametric effects of our multiple changes by analyzing the effect on optimal value of changes in the single objective coefficient θ . The process works because the main dual constraint corresponding to new variable x_3 is

$$\nu_6 = \theta$$

Thus θ affects all other main dual constraints by a term $-\Delta c_j \nu_6 = -\Delta c_j \theta$ on the left-hand side, which is equivalent to increasing at the same rate in the dual right-hand side \mathbf{c} . With primal and dual guaranteed to yield equal optimal values, manipulating the dual in this way does to the primal exactly what we want.

Using objective function range outputs for parameter θ leads us to the following sequence of runs and the parametric curve of Figure 6.9.

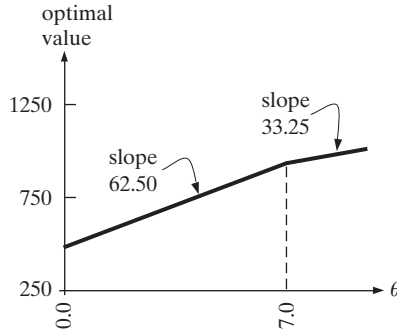


FIGURE 6.9 Parametric Multiple Objective Coefficient Change of the Two Crude Example

Case	Coefficient θ	Optimal x_3	Lower Range	Upper Range
Base model	2.000	100.000	1.125	2.625
Variant 1	$2.625 + \varepsilon$	333.333	2.626	5.100
Variant 2	$5.100 + \varepsilon$	$+\infty$ (infeasible)	5.100	$+\infty$
Variant 3	$1.125 - \varepsilon$	0.000	$-\infty$	1.125

EXAMPLE 6.27: ANALYZING MULTIPLE OBJECTIVE CHANGES

Show how to modify the model in Example 6.26 to parametrically analyze the effect of simultaneously increasing the first objective coefficient and decreasing the second at twice the rate.

Solution: The indicated change involves $\Delta c_1 = +1, \Delta c_2 = -2$. Thus, applying principle [6.45], we may parametrically analyze the change by varying new objective coefficient θ in the modified model:

$$\begin{aligned}
 \max \quad & + 5x_1 + 2x_2 + \theta x_3 \\
 \text{s.t.} \quad & + 1x_1 + 1x_2 \leq 3 \\
 & + 1x_1 \leq 2 \\
 & -1x_1 + 2x_2 + 1x_3 = 0 \\
 & x_1, x_2 \geq 0, x_3 \text{ URS}
 \end{aligned}$$

6.7 DUALITY AND OPTIMALITY IN LINEAR PROGRAMMING

Relationships between primal linear programs and their duals have import well beyond the intuitive sensitivity results developed in Sections 6.3–6.6. In this section we investigate those important theoretical relationships and their consequences of LP algorithm strategies.

To begin, it will be useful to use a single concise form for primal and dual. We know there is no loss of generality (Section 5.1) in adopting the following inequality forms:

$$\begin{array}{ll} \min & \mathbf{c} \cdot \mathbf{x} \\ \text{(Primal) s.t.} & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \quad \begin{array}{ll} \max & \mathbf{v} \cdot \mathbf{b} \\ \text{(Dual) s.t.} & \mathbf{vA} \leq \mathbf{c} \\ & \mathbf{v} \geq \mathbf{0} \end{array} \quad (6.9)$$

Dual of the Dual

We begin with one of the easiest relationships to see.

Principle 6.46 The dual of the dual of any linear program is the primal.

We can see that this must be true by simply rearranging the primal-dual pair in (6.9).

$$\begin{array}{ll} \max & \mathbf{v} \cdot \mathbf{b} \\ \text{(Dual) s.t.} & \mathbf{vA} \leq \mathbf{c} \\ & \mathbf{v} \geq \mathbf{0} \end{array} \quad \begin{array}{ll} \min & \mathbf{c} \cdot \mathbf{y} \\ \text{(DualDual) s.t.} & \mathbf{Ay} \geq \mathbf{b} \\ & \mathbf{y} \geq \mathbf{0} \end{array}$$

On the left we now have the former dual playing the role of the primal. Now let the variables of its dual be denoted \mathbf{y} . The corresponding objective function and constraints will then be those shown on the right above. But comparison to the (6.9) primal demonstrates this dual of the dual is identical to the original primal except for the name of the variables.

Weak Duality between Objective Values

The next relationship to investigate is **weak duality** – how objective values of feasible solutions to one of the problems bound the values of feasible solutions to the other.

Principle 6.47 The primal objective function evaluated at any feasible solution to a minimize primal is \geq the objective function value of the corresponding dual evaluated at any dual feasible solution. For a maximize primal, it is \leq .

To see why this weak duality must hold, let $\bar{\mathbf{x}}$ be a feasible solution to the above primal, and $\bar{\mathbf{v}}$ a feasible solution to the corresponding dual. Then adding and subtracting a common quantity and regrouping shows

$$\begin{aligned} \mathbf{c} \cdot \bar{\mathbf{x}} - \bar{\mathbf{v}} \cdot \mathbf{b} &= \mathbf{c} \cdot \bar{\mathbf{x}} - \bar{\mathbf{v}} \mathbf{A} \bar{\mathbf{x}} + \bar{\mathbf{v}} \mathbf{A} \bar{\mathbf{x}} - \bar{\mathbf{v}} \cdot \mathbf{b} \\ &= (\mathbf{c} - \bar{\mathbf{v}} \mathbf{A}) \cdot \bar{\mathbf{x}} + \bar{\mathbf{v}} \cdot (\mathbf{A} \bar{\mathbf{x}} - \mathbf{b}) \end{aligned} \quad (6.10)$$

If $\bar{\mathbf{x}}$ is primal feasible in (6.9), and $\bar{\mathbf{v}}$ is feasible in the corresponding dual, then $(\mathbf{c} - \bar{\mathbf{v}} \mathbf{A})$, $\bar{\mathbf{x}}$, $\bar{\mathbf{v}}$, and $(\mathbf{A} \bar{\mathbf{x}} - \mathbf{b})$ are all nonnegative. Thus (6.10) implies the difference in the optimal solution values is nonnegative, and primal value $\mathbf{c} \cdot \bar{\mathbf{x}}$ is indeed \geq dual solution value $\bar{\mathbf{v}} \cdot \mathbf{b}$.

EXAMPLE 6.28: VERIFYING WEAK DUALITY

For the Two linear programs in Example 6.17, verify that the following primal and dual solutions are feasible, and show that they conform to weak duality property [6.47](#).

(a) Primal solution: $x_1 = 1, x_2 = 1, x_3 = 1$

Dual solution: $v_1 = 2, v_2 = 6, v_3 = -1$

(b) Primal solution: $x_1 = 2, x_2 = 1, x_3 = 0$

Dual solution: $v_1 = -2, v_2 = 4, v_3 = -1$

Solution:

(a) To check primal feasibility, we first note all specified $x_j \geq 0$. For main primal constraints,

$$\begin{aligned} +1x_1 - 1x_2 - 1x_3 &= +1(1) - 1(1) + 1(1) = 1 \geq 1 \\ +3x_1 + 1x_2 &= +3(1) + 1(1) = 4 \\ +4x_2 + 1x_3 &= +4(1) + 1(1) = 5 \leq 10 \end{aligned}$$

Turning to dual feasibility, main constraints (see Example 6.17) have

$$\begin{aligned} +1v_1 + 3v_2 &= +1(2) + 3(6) = 20 \leq 30 \\ -1v_1 + 1v_2 + 4v_3 &= -1(2) + 1(6) + 4(-1) = 0 \leq 0 \\ +1v_1 + 1v_3 &= +1(2) + 1(-1) = 1 \leq 5 \end{aligned}$$

The given solution also satisfies type restrictions because

$$v_1 = 2 \geq 0 \text{ and } v_3 = -1 \leq 0$$

Objective function values for the given solutions are

$$\begin{aligned} +30x_1 + 5x_3 &= +30(1) + 5(1) = 35 \\ +1v_1 + 4v_2 + 10v_3 &= +1(2) + 4(6) + 10(-1) = 16 \end{aligned}$$

As implied by weak duality principle [6.47](#), the primal value is \geq the dual.

(b) To check primal feasibility, we first note again that all specified $x_j \geq 0$. Verifying main primal constraints,

$$\begin{aligned} +2x_1 + 1x_2 &= +2(2) + 1(1) = 5 \geq 3 \\ +5x_1 + 3x_2 - 1x_3 &= 5(2) + 3(1) - 1(0) = 13 \leq 15 \\ +1x_2 + 1x_3 &= +1(1) + 1(0) = 1 \end{aligned}$$

Turning to dual feasibility, main constraints have

$$\begin{aligned} +2v_1 + 5v_2 &= 2(-2) + 5(4) = 16 \geq 10 \\ +1v_1 + 3v_2 + 1v_3 &= +1(-2) + 3(4) + 1(-1) = 9 \geq 9 \\ -1v_2 + 1v_3 &= -1(4) + 1(-1) = -5 \geq -6 \end{aligned}$$

The given solution also satisfies type restrictions because

$$v_1 = -2 \leq 0 \text{ and } v_2 = 4 \geq 0$$

Objective function values for the given solutions are

$$\begin{aligned} +10x_1 + 9x_2 - 6x_3 &= +10(2) + 9(1) - 6(0) &= 29 \\ +3v_1 + 15v_2 + 1v_3 &= +3(-2) + 15(4) + 1(-1) &= 53 \end{aligned}$$

Consistent with weak duality principle [6.47], the primal value is \leq the dual.

Unbounded and Infeasible Cases

Strong duality and complementary slackness conditions depend on the primal (and thus the dual) having optimal solutions. We would hardly undertake post-optimality analysis if there were no optimal solutions. Still, other cases are sometimes of interest.

We learned very early that LPs can prove infeasible or unbounded. Weak duality condition [6.47] provides a key tool in infeasible and unbounded instances. Suppose, for example, that a primal minimize problem is unbounded. Weak duality says that any dual feasible solution yields a lower bound on primal objective function values. But if the primal is unbounded, no such bound exists. The only possible explanation is that no dual feasible solutions exist.

We could make a similar argument for unbounded duals. Any primal feasible solution would limit dual objective function values, so none can exist.

Principle 6.48 If either a primal LP model or its dual is unbounded, the other is infeasible.

After seeing so much elegant symmetry between primal and dual, one might guess, conversely, that infeasibility in one problem also implies unboundedness in the other. This is false! Both primal and dual in the following are obviously infeasible:

$$\begin{array}{ll} \min & -x_1 \\ \text{(Primal) s.t.} & x_1 - x_2 \geq 1 \\ & x_1 - x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{array} \quad \begin{array}{ll} \max & v_1 \\ \text{(Dual) s.t.} & v_1 + v_2 \leq -1 \\ & -v_1 - v_2 \leq 0 \\ & v_1 \geq 0, v_2 \leq 0 \end{array}$$

Principle 6.49 Possible outcomes for pairs of primal and dual linear programs may be summarized as follows:

Primal	Dual		
	Optimal	Infeasible	Unbounded
optimal	possible	never	never
infeasible	never	possible	possible
unbounded	never	possible	never

Notice one other elegance of case listing table [6.49]. Both the row for a primal optimum and the column for a dual one have only one possible choice. This can easily be deduced by process of elimination. If the primal has an optimal solution, then weak

duality [6.47] assures the corresponding dual cannot be unbounded. Furthermore, with primal neither infeasible nor unbounded, the dual cannot also not be infeasible. This leaves only the possibility that the dual is optimal. A similar argument demonstrates that a dual optimum must imply a primal one.

EXAMPLE 6.29: RELATING UNBOUNDEDNESS AND INFEASIBILITY

(a) Show graphically that the following linear program is unbounded and verify the implication for its dual.

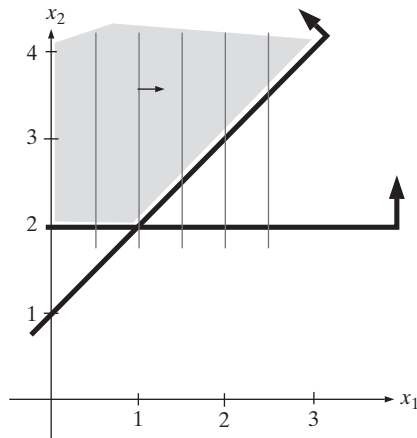
$$\begin{aligned} \max \quad & +1x_1 \\ \text{s.t.} \quad & -1x_1 + 1x_2 \geq 1 \\ & + 1x_2 \geq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

(b) Show graphically that the following linear program has an unbounded dual and verify the implication for the primal.

$$\begin{aligned} \min \quad & +1x_1 + 1x_2 \\ \text{s.t.} \quad & +2x_1 + 1x_2 \geq 3 \\ & +5x_1 + 1x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution:

(a) The following plot clearly shows the model is unbounded:



Its dual is

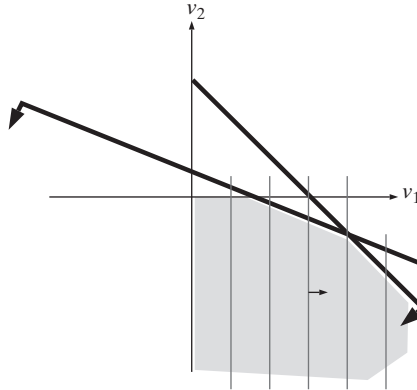
$$\begin{aligned} \min \quad & +1v_1 \\ \text{s.t.} \quad & -1v_1 \geq 1 \\ & +1v_1 + 1v_2 \geq 0 \\ & v_1, v_2 \leq 0 \end{aligned}$$

which is infeasible as implied by principle [6.48].

(b) The dual of this linear program is

$$\begin{aligned} \max \quad & +3v_1 \\ \text{s.t.} \quad & +2v_1 + 5v_2 \leq 1 \\ & +1v_1 + 1v_2 \leq 1 \\ & v_1 \geq 0, v_2 \leq 0 \end{aligned}$$

Solving graphically, gives



which is clearly unbounded. It follows from principle [6.48] that the primal is infeasible.

Complementary Slackness and Optimality

Recall from Section 6.3 principles [6.26] and [6.27] that **complementary slackness** relates slack in primal inequalities to nonzero values of corresponding dual variables and vice versa. Another look at weak duality calculation (6.10) reveals more. The two final (nonnegative) expressions in the difference between feasible primal and dual solutions values there are exactly total complementary slackness.

$$\begin{aligned} (\mathbf{c} - \bar{\mathbf{v}}\mathbf{A}) \cdot \bar{\mathbf{x}} &= \sum_j (c_j - \sum_i \bar{v}_i a_{i,j}) \bar{x}_j \text{ and} \\ \bar{\mathbf{v}} \cdot (\mathbf{A}\bar{\mathbf{x}} - \mathbf{b}) &= \sum_i \bar{v}_i (\sum_j a_{i,j} \bar{x}_j - b_i) \end{aligned} \quad (6.11)$$

Principle 6.50 If $\bar{\mathbf{x}}$ is a feasible solution to a primal linear program, and $\bar{\mathbf{v}}$ is a feasible solution to the corresponding dual, then the difference in their objective values is exactly the total complementary slackness between the solutions. Furthermore, if objective function values agree, that is, $\mathbf{c} \cdot \bar{\mathbf{x}} = \bar{\mathbf{v}} \cdot \mathbf{b}$, then both $\bar{\mathbf{x}}$ and $\bar{\mathbf{v}}$ are optimal in their respective problems, and complementary slackness conditions

$$(\mathbf{c} - \bar{\mathbf{v}}\mathbf{A}) \cdot \bar{\mathbf{x}} = \mathbf{0} \text{ and } \bar{\mathbf{v}} \cdot (\mathbf{A}\bar{\mathbf{x}} - \mathbf{b}) = \mathbf{0}$$

between the optimal solutions must hold at optimality as in [6.26] and [6.27].

We have already established with weak duality [6.47] that values of feasible solutions to either problem bound those of the other. If solution value equality is achieved, both must be the best possible in their respective problems and thus optimal. Furthermore, equation (6.10) shows that the difference in primal and dual solution value is exactly the sum (equations ([6.50])) of terms relating slack in constraints of one problem to values of corresponding variables in the other. If there is no solution value difference, all such complementarity terms must = 0, justifying principles [6.26] and [6.27].

EXAMPLE 6.30: VERIFYING COMPLEMENTARY SLACKNESS AT OPTIMALITY

Consider the LP

$$\begin{aligned} \max \quad & 5x_1 + 7x_2 + 10x_3 \\ \text{s.t.} \quad & +2x_1 - 1x_2 + 5x_3 \leq 10 \\ & +1x_1 + 3x_2 \leq 15 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Using dual variables v_1, v_2 the dual is

$$\begin{aligned} \min \quad & 10v_1 + 15v_2 \\ \text{s.t.} \quad & 2v_1 + 1v_2 \geq 5 \\ & -1v_1 + 3v_2 \geq 7 \\ & 5v_1 \geq 10 \\ & v_1, v_2 \geq 0 \end{aligned}$$

An optimal primal solution is $\mathbf{x}^* = (0, 5, 3)$ and a corresponding optimal dual is $\mathbf{v}^* = (2, 3)$.

- (a) Write all primal and dual complementary slackness conditions for this maximize model in product form (6.11).
- (b) Verify that the conditions are satisfied by the given primal and dual optimal solution.

Solution:

(a) Primal complementary slackness [6.26] for tills maximize instance relates primal inequality slack to dual variable values:

$$\begin{aligned} (10 - 2x_1 + 1x_2 - 5x_3)v_1 &= 0 \\ (15 - 1x_1 - 3x_2)v_2 &= 0 \end{aligned}$$

Dual complementary slackness [6.27] relates dual inequality slack to primal variable values:

$$\begin{aligned} (2v_1 + 1v_2 - 5)x_1 &= 0 \\ (v_1 - 3v_2 - 7)x_2 &= 0 \\ (5v_1 - 10)x_3 &= 0 \end{aligned}$$

(b) Checking primal complementary slackness conditions,

$$\begin{aligned}(10 - 2x_1 + 1x_2 - 5x_3)v_1 &= (10 - 2(0) + 1(5) - 5(3))(2) = 0 \\ (15 - 1x_1 - 3x_2)v_2 &= (15 - 1(0) - 3(5))(3) = 0\end{aligned}$$

Similarly in dual complementary slackness conditions,

$$\begin{aligned}(2v_1 + 1v_2 - 5)x_1 &= (2(2) + 1(3) - 5)(0) = 0 \\ (v_1 - 3v_2 - 7)x_2 &= (1(2) - 3(3) - 7)(5) = 0 \\ (5v_1 - 10)x_3 &= (5(2) - 10)(3) = 0\end{aligned}$$

Strong Duality and Karush-Kuhn-Tucker (KKT) Optimality Conditions for Linear Programs

We addressed in [6.47] how weak duality relationships between objective values of feasible solutions to primal and dual bound each other. But when both problems have feasible solutions, more powerful **strong duality** can be established.

Principle 6.51 If either a primal linear program or its dual has a finite optimal solution, they both do, and their optimal solution values are equal.

The underlying reasons strong duality must hold derive from the fundamental 3-part **Karush-Kuhn-Tucker (KKT) optimality** conditions that guide all algorithms for linear programming. (See section 17.4 for NLP generalizations of these conditions.)

Principle 6.52 If primal and dual solutions \bar{x} and \bar{v} are respectively primal and dual feasible in a given LP and its dual, and they mutually satisfy complementary slackness conditions [6.50], then both solutions are optimal in their respective problems, and their optimal solution values are equal. That is, KKT conditions (i) primal feasibility, (ii) dual feasibility, and (iii) complementary slackness are necessary and sufficient for LP optimality.

Verification of the necessary part of this proposition that there must exist complementary primal and dual optima for every feasible and bounded LP will be delayed to the discussion of basic solution optima below (see principle [6.60]). But the sufficient part of the 3-part conditions that solutions satisfying the conditions are necessarily optimal follows immediately from what has been established in principle [6.50]. If both primal and dual solutions are feasible, and their objective function value difference, which is equivalent to the sum of complementary slackness conditions (6.11), is $= 0$, then both solutions are optimal in their respective problems, and their solution values agree.

EXAMPLE 6.31: FORMULATING KKT OPTIMALITY CONDITIONS

Formulate KKT optimal conditions for the (primal) linear program

$$\begin{aligned}\min & +5x_1 + 13x_2 + 9x_3 \\ \text{s.t.} & +2x_1 + 4x_2 + x_3 \geq 4 \\ & -1x_1 + 3x_2 + x_3 \geq 1 \\ & x_1, x_2, x_3 \geq 0\end{aligned}$$

Solution:

Using dual variables v_1, v_2 for the two main constraints, the corresponding dual is

$$\begin{aligned} \max \quad & 4v_1 + v_2 \\ \text{s.t.} \quad & 2v_1 - 1v_2 \leq 5 \\ & 4v_1 + 3v_2 \leq 13 \\ & v_1 + v_2 \leq 9 \\ & v_1, v_2 \geq 0 \end{aligned}$$

Then by [6.52] the required KKT conditions for optimality of solutions $\bar{x}_1, \bar{x}_2, \bar{x}_3$ and \bar{v}_1, \bar{v}_2 are

[Primal Feasibility]	[Dual Feasibility]	[Complementary Slackness]
$2\bar{x}_1 + 4\bar{x}_2 + \bar{x}_3 \geq 4$	$2\bar{v}_1 - \bar{v}_2 \leq 5$	$(2\bar{x}_1 + 4\bar{x}_2 + \bar{x}_3 - 4)\bar{v}_1 = 0$
$-\bar{x}_1 + 3\bar{x}_2 + \bar{x}_3 \geq 1$	$4\bar{v}_1 + 3\bar{v}_2 \leq 13$	$(-\bar{x}_1 - 3\bar{x}_2 + \bar{x}_3 - 1)\bar{v}_2 = 0$
$\bar{x}_1, \bar{x}_2, \bar{x}_3 \geq 0$	$\bar{v}_1 + \bar{v}_2 \leq 9$	$(5 - 2\bar{v}_1 + \bar{v}_2)\bar{x}_1 = 0$
	$\bar{v}_1, \bar{v}_2 \geq 0$	$(13 - 4\bar{v}_1 - 3\bar{v}_2)\bar{x}_2 = 0$
		$(9 - \bar{v}_1 - \bar{v}_2)\bar{x}_3 = 0$

Models in Standard Form

Recall from definition [5.7] that the **standard form** for linear programs has only equality main constraints and nonnegative decision variables. It is easy to represent that standard form in matrix notation and formulate its dual.

Principle 6.53 Assuming a minimizing primal objective, primal and dual linear programs in standard form can be represented

min	cx		max	vb
[Primal]	s.t.	Ax = b	[Dual]	s.t.
		x ≥ 0		vA ≤ c
				v URS

Notice that all dual variables are unrestricted because primal main constraints are equalities.

EXAMPLE 6.32: FORMULATING PRIMAL AND DUAL IN STANDARD FORM

- Return to the primal linear program of Example 6.31.
- (a) Subtract nonnegative slack variables x_4 and x_5 to place the primal in standard form.
 - (b) Identify the corresponding matrix/vector parameters **A**, **b**, and **c** of format [6.53].
 - (c) Formulate the dual of your standard form primal in (a),

Solution:

(a) With slacks the standard form becomes

$$\begin{aligned} \min \quad & 5x_1 + 13x_2 + 9x_3 \\ \text{s.t.} \quad & 2x_1 + 4x_2 + x_3 - x_4 = 4 \\ & -x_1 + 3x_2 + x_3 - x_5 = 1 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

(b) The parameters in matrix form are

$$\mathbf{A} = \begin{pmatrix} 2 & 4 & 1 & -1 & 0 \\ -1 & 3 & 1 & 0 & -1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}, \mathbf{c} = (5 \ 13 \ 9 \ 0 \ 0)$$

(c) Using dual variables v_1, v_2 the dual is

$$\begin{aligned} \max \quad & 4v_1 + v_2 \\ \text{s.t.} \quad & 2v_1 - v_2 \leq 5 \\ & 4v_1 + 3v_2 \leq 13 \\ & v_1 + v_2 \leq 9 \\ & -v_1 \leq 0 \\ & -v_2 \leq 0 \end{aligned}$$

Adding complementary slackness to the primal and dual constraints of [6.53] gives full KKT optimality conditions for a linear program in matrix standard form.

Definition 6.54 KKT optimality conditions for LP standard form solutions $\bar{\mathbf{x}}$ and $\bar{\mathbf{v}}$ in [6.53] are

[Primal Feasibility]	[Dual Feasibility]	[Complementary Slackness]
$\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$	$\bar{\mathbf{v}}\mathbf{A} \leq \mathbf{c}$	$(\mathbf{c} - \bar{\mathbf{v}}\mathbf{A})\bar{\mathbf{x}} = \mathbf{0}$
$\bar{\mathbf{x}} \geq \mathbf{0}$		

EXAMPLE 6.33: FORMULATING KKT CONDITIONS FOR STANDARD FORM

Return to the standard form linear program of Example 6.32 and formulate KKT conditions for optimality of solutions $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5$ and \bar{v}_1, \bar{v}_2 .

Solution: Direct substitution in conditions [6.54] yields

[Primal Feasibility]	[Dual Feasibility]	[Complementary Slackness]
$2\bar{x}_1 + 4\bar{x}_2 + \bar{x}_3 - \bar{x}_4 = 4$	$2\bar{v}_1 - \bar{v}_2 \leq 5$	$(5 - 2\bar{v}_1 + \bar{v}_2)\bar{x}_1 = 0$
$-\bar{x}_1 + 3\bar{x}_2 + \bar{x}_3 - \bar{x}_5 = 1$	$4\bar{v}_1 + 3\bar{v}_2 \leq 13$	$(13 - 4\bar{v}_1 - 3\bar{v}_2)\bar{x}_2 = 0$
$\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5 \geq 0$	$\bar{v}_1 + \bar{v}_2 \leq 9$	$(9 - \bar{v}_1 - \bar{v}_2)\bar{x}_3 = 0$
	$-\bar{v}_1 \leq 0$	$(\bar{v}_1)\bar{x}_4 = 0$
	$-\bar{v}_2 \leq 0$	$(\bar{v}_2)\bar{x}_5 = 0$

Standard Form LPs in Partitioned Basic Format

Like the simplex algorithms of Chapter 5, most of the analysis to follow in this chapter will deal with constraints and parameters rearranged to partition all the elements for a current basis versus the others for nonbasics.

Principle 6.55 Given primal and dual linear programs in standard form [6.53], and a current basis submatrix \mathbf{B} collecting constraint columns for basic variables, the corresponding partitioned forms of primal and dual are

$$\begin{array}{ll}
 \min & \mathbf{c}^B \mathbf{x}^B + \mathbf{c}^N \mathbf{x}^N \\
 \text{[Primal] s.t.} & \mathbf{B} \mathbf{x}^B + \mathbf{N} \mathbf{x}^N = \mathbf{b} \\
 & \mathbf{x}^B \geq \mathbf{0}, \mathbf{x}^N \geq \mathbf{0} \\
 \max & \mathbf{v} \mathbf{b} \\
 \text{[Dual] s.t.} & \mathbf{v} \mathbf{B} \leq \mathbf{c}^B \\
 & \mathbf{v} \mathbf{N} \leq \mathbf{c}^N \\
 & \mathbf{v} \text{ URS}
 \end{array}$$

Here \mathbf{x}^B and \mathbf{x}^N are the basic and nonbasic components of the decision vector \mathbf{x} , \mathbf{B} , and \mathbf{N} are the corresponding submatrices of the main constraints, \mathbf{c}^B and \mathbf{c}^N are the vectors of basic and nonbasic objective function coefficients, and \mathbf{b} is the instance right-hand-side.

EXAMPLE 6.34: FORMULATING LPs IN PARTITIONED STANDARD FORM

Return to the primal and dual LPs of Example 6.33 and identify the basic and nonbasic elements of partitioned format [6.55] using basis $\{x_1, x_3\}$.

Solution: For the given basis, $\mathbf{x}^B = (x_1, x_3)$ and $\mathbf{x}^N = (x_2, x_4, x_5)$. The corresponding parameters are

$$\begin{array}{lll}
 \mathbf{c}^B = (5, 9) & \mathbf{c}^N = (13, 0, 0) & \mathbf{B} = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix} \\
 \mathbf{N} = \begin{pmatrix} 4 & -1 & 0 \\ 3 & 0 & -1 \end{pmatrix}, & \text{and} & \mathbf{b} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}
 \end{array}$$

As above, we can elaborate formulations [6.55] to derive 3-part KKT optimality conditions for partitioned standard-form linear programs.

Principle 6.56 KKT optimality conditions for solutions $\bar{\mathbf{x}}^B, \bar{\mathbf{x}}^N$ and $\bar{\mathbf{v}}$ of primal and dual linear programs in partitioned standard form [6.55] can be expressed

$$\begin{array}{lll}
 \text{[Primal Feasibility]} & \text{[Dual Feasibility]} & \text{[Complementary Slackness]} \\
 \mathbf{B} \bar{\mathbf{x}}^B + \mathbf{N} \bar{\mathbf{x}}^N = \mathbf{b} & \bar{\mathbf{v}} \mathbf{B} \leq \mathbf{c}^B & (\mathbf{c}^B - \bar{\mathbf{v}} \mathbf{B}) \bar{\mathbf{x}}^B = \mathbf{0} \\
 \bar{\mathbf{x}}^B \geq \mathbf{0}, \bar{\mathbf{x}}^N \geq \mathbf{0} & \bar{\mathbf{v}} \mathbf{N} \leq \mathbf{c}^N & (\mathbf{c}^N - \bar{\mathbf{v}} \mathbf{N}) \bar{\mathbf{x}}^N = \mathbf{0}
 \end{array}$$

Basic Solutions in Partitioned Form

Recall from definition [5.16](#) that basic solutions to linear programs in standard form are constructed by fixing the values of nonbasic variables at their lower bound $= 0$ and solving main constraints to compute the corresponding values of basic variables.

Principle 6.57 Solution $(\bar{\mathbf{x}}^B, \bar{\mathbf{x}}^N)$ is a **primal basic solution** to the partitioned standard-form in [6.55](#), if $\bar{\mathbf{x}}^N = 0$, $\bar{\mathbf{x}}^B = \mathbf{B}^{-1}\mathbf{b}$. Its objective function value is $\mathbf{c}^B \mathbf{B}^{-1}\mathbf{b}$. The solution is primal feasible if $\bar{\mathbf{x}}^B \geq 0$.

These results follow because fixing nonbasics $= 0$ leaves basics as the solution to $\mathbf{B}\mathbf{x}^B = \mathbf{b}$ or $\bar{\mathbf{x}}^B = \mathbf{B}^{-1}\mathbf{b}$, which must be nonnegative to be feasible. Applying objective coefficients \mathbf{c}^B to the only non-zero part of the basic solution yields objective value $\mathbf{c}\bar{\mathbf{x}} = \mathbf{c}^B \bar{\mathbf{x}}^B = \mathbf{c}^B \mathbf{B}^{-1}\mathbf{b}$.

EXAMPLE 6.35: COMPUTING BASIC SOLUTIONS IN PARTITIONED FORM

Return to the standard for primal problem and basis $\{x_1, x_3\}$ of Example 6.34. Compute the corresponding primal basic solution, determine its objective value, and establish whether it is primal feasible.

Solution: From the previous Example, we have

$$\mathbf{B} = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix}, \text{ inverse } \mathbf{B}^{-1} = \begin{pmatrix} 1/3 & -1/3 \\ 1/3 & 2/3 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}, \mathbf{c}^B = \begin{pmatrix} 5 \\ 9 \end{pmatrix}$$

Thus from [6.57](#)

$$\bar{\mathbf{x}}^N = (\bar{x}_2, \bar{x}_4, \bar{x}_5) = 0, \bar{\mathbf{x}}^B = (\bar{x}_1, \bar{x}_3) = \mathbf{B}^{-1}\mathbf{b} = (1, 2)$$

The solution is feasible because both basic components are nonnegative, nonbasic components $= 0$, and the main equality constraints are satisfied by construction. Its objective value is $\mathbf{c}^B \bar{\mathbf{x}}^B = \mathbf{c}^B \mathbf{B}^{-1}\mathbf{b} = 23$.

Complementary Dual Basic Solutions

Discussion of the Revised Simplex in Section 5.8 constructed a pricing vector [5.45](#) denoted \mathbf{v} . It is easy to see that its name is no accident. The pricing vector is exactly a standard dual basic solution to LPs in partitioned standard form.

Principle 6.58 The **complementary dual basic solution** corresponding to basis matrix \mathbf{B} is given by the solution to $\bar{\mathbf{v}}\mathbf{B} = \mathbf{c}^B$ or $\bar{\mathbf{v}} = \mathbf{c}^B \mathbf{B}^{-1}$. The solution is dual feasible if it satisfies $\bar{\mathbf{v}}\mathbf{N} \leq \mathbf{c}^N$. Whether or not feasible, its dual objective value matches that of primal solution $(\bar{\mathbf{x}}^B, \bar{\mathbf{x}}^N)$ in [6.57](#) at $\bar{\mathbf{v}}\mathbf{b} = \mathbf{c}^B \mathbf{B}^{-1}\mathbf{b} = \mathbf{c}\bar{\mathbf{x}}$, and the two mutually satisfy all complementary slackness of KKT conditions [6.56](#).

To verify these computations, note that the dual basic solution construction of [6.58](#) automatically satisfies the first basic-variables part of dual feasibility in

conditions [6.56] by constructing $\bar{\mathbf{v}}\mathbf{B} = \mathbf{c}^B$ with $\bar{\mathbf{v}} = \mathbf{c}^B\mathbf{B}^{-1}$. What remains for dual feasibility is to check dual constraints for nonbasics variables, that is, $\bar{\mathbf{v}}\mathbf{N} \leq \mathbf{c}^N$. Regardless of whether $\bar{\mathbf{v}}$ is dual feasible, however, both primal and dual objective function values match at $\bar{\mathbf{v}}\mathbf{b} = \mathbf{c}^B\mathbf{B}^{-1}\mathbf{b} = \mathbf{c}\bar{\mathbf{x}}$. Also, making $\bar{\mathbf{v}}\mathbf{B} = \mathbf{c}^B$ assures the first, basic part of complementary slackness is satisfied in [6.56]. The second is automatic with $\bar{\mathbf{x}}^N = 0$.

EXAMPLE 6.36: COMPUTING COMPLEMENTARY DUAL BASIC SOLUTIONS

Return to the partitioned standard form primal and dual linear programs of Examples 6.34 and 6.35, with basis $\{x_1, x_3\}$.

- (a) Compute the dual basic solution of [6.58], and determine if it is dual feasible.
- (b) Verify that the solution of (a) has the same objective function value as previously computed for the primal.
- (c) Verify that dual basic solution and the primal solution $(\bar{\mathbf{x}}^B, \bar{\mathbf{x}}^N)$ of Example 6.35 mutually satisfy all required complementary slackness conditions.

Solution:

(a) With $\mathbf{c}^B = (5, 9)$, and $\mathbf{B}^{-1} = \begin{pmatrix} 1/3 & -1/3 \\ 1/3 & 2/3 \end{pmatrix}$, dual solution $\bar{\mathbf{v}} = \mathbf{c}^B\mathbf{B}^{-1} = (14/3, 13/3)$. To determine whether it is dual feasible, we must check dual constraints for nonbasics primal variables. One of those for x_2 requires $4\bar{v}_1 + 3\bar{v}_2 \leq 13$, which is violated because $4(14/3) + 3(13/3) > 13$. Thus solution $\bar{\mathbf{v}}$ is dual infeasible.

(b) Despite solution $\bar{\mathbf{v}}$ being infeasible, its objective function value $\bar{\mathbf{v}}\mathbf{b} = (14/3, 13/3) \cdot (4, 1) = 23$, which matches the primal objective value of Example 6.35.

(c) Referring back to Example 6.33, there is one complementary slackness condition for each primal variable. Despite the dual solution being infeasible, complementary slackness constraints for nonbasics are automatically satisfied by $\bar{\mathbf{x}}^N = (\bar{x}_2, \bar{x}_4, \bar{x}_5) = 0$. Those for basic variables have $(5 - 2\bar{v}_1 + \bar{v}_2)\bar{x}_1 = 0$ and $(9 - \bar{v}_1 - \bar{v}_2)\bar{x}_3 = 0$ are also automatically satisfied because $\mathbf{c}^B - \bar{\mathbf{v}}\mathbf{B} = (5, 9) - (14/3, 13/3)\begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix} = (0, 0)$.

There is another big benefit of computing complementary dual solutions as in [6.58]. Since both primal and dual LPs were defined over the same parameters in Sections 6.3 and 6.4, it would be natural to believe that two searches would be needed to obtain optimal solutions for both, one for the primal and another for the dual. But principle [6.58] now makes clear that only one improving search is required (here a primal). We know from principles [5.6] and [5.19] that every LP having a finite optimal solution has one at an extreme point of its feasible set, and that the basic feasible solutions for LPs are the extreme-point solutions. Thus if we can find an optimal primal basis, both the primal optimum and the corresponding dual can be

readily computed from that same basis. All that is needed to make KKT optimality operational is an algorithm to find an optimal primal basic solution and complementary dual that can be proved feasible. The familiar simplex algorithms of Chapter 5 will do the job.

Primal Simplex Optimality and Necessity of KKT Conditions

When Primal Simplex search is being applied, we can show that the algorithm's stopping rule corresponds exactly to meeting that last, dual feasibility requirement $\bar{\mathbf{v}}\mathbf{N} \leq \mathbf{c}^N$ for minimize primals (respectively $\bar{\mathbf{v}}\mathbf{N} \geq \mathbf{c}^N$ for maximize). First, recall that a simplex direction is constructed for nonbasic variable x_j by letting its direction component $\Delta x_j = 1$, setting components $\Delta x_k = 0$ for other nonbasics x_k , then solving for the basic part of the direction $\Delta \mathbf{x}^B$ to satisfy feasibility requirement $\mathbf{A} \Delta \mathbf{x} = 0$. Specifically, for \mathbf{a}^j the j th constraint column, we need

$$\begin{aligned} \mathbf{B} \Delta \mathbf{x}^B &= -\mathbf{a}^j && \text{or multiplying through by } \mathbf{B}^{-1} \\ \Delta \mathbf{x}^B &= -\mathbf{B}^{-1}\mathbf{a}^j \end{aligned}$$

Then the direction's reduced cost \bar{c}_j is just $\mathbf{c} \cdot \Delta \mathbf{x}$ or

$$\begin{aligned} \bar{c}_j &= c_j - \mathbf{c}^B \Delta \mathbf{x}^B && \text{which is} \\ &= c_j - \mathbf{c}^B (\mathbf{B}^{-1}\mathbf{a}^j) && \text{and regrouping} \\ &= c_j - \mathbf{c}^B \mathbf{B}^{-1}(\mathbf{a}^j) && \text{which becomes} \\ &= c_j - \bar{\mathbf{v}}\mathbf{a}^j \end{aligned}$$

That is, the \bar{c} test for primal simplex optimality is exactly what we need to establish the one remaining part of KKT conditions.

$$\begin{aligned} \bar{\mathbf{c}}^N \geq 0 & \quad \text{implies} \quad \bar{\mathbf{v}}\mathbf{N} \leq \mathbf{c}^N && \text{for minimize problems, and} \\ \bar{\mathbf{c}}^N \leq 0 & \quad \text{implies} \quad \bar{\mathbf{v}}\mathbf{N} \geq \mathbf{c}^N && \text{for maximize problems} \end{aligned}$$

Principle 6.59 | Primal simplex algorithms can be understood as pursuing an improving **primal search strategy** through a series of basic solutions to fulfill KKT optimality conditions [6.56] by (i) starting with and maintaining primal feasibility of solutions visited, (ii) implicitly or explicitly supplementing each with a basic dual solution that satisfies complementary slackness with the primal and facilitates pricing of simplex directions, and (iii) terminating when unboundedness is detected or the current dual basic solution proves dual feasible. That is, primal simplex algorithms maintain primal feasibility and complementary slackness while seeking dual feasibility.

To see an example, refer to the Table 5.9 Revised Simplex solution of Top Brass Application 5.1. It starts with the all-slack primal basic solution having $\bar{\mathbf{x}}^N = (\bar{x}_1, \bar{x}_2) = (0, 0)$ and primal feasible $\bar{\mathbf{x}}^B = (\bar{x}_3, \bar{x}_4, \bar{x}_5, \bar{x}_6) = (1000, 1500, 1750, 4800)$. Subsequent iterations improve the solution value of the basic solution, but always keep primal feasibility by choosing step size λ to move the search just up to the point where feasibility would be lost with a basic variable going negative.

The dual (pricing) vector \mathbf{v} at each iteration is constructed as shown in principle [6.58] as $\bar{\mathbf{v}} = \mathbf{c}^B \mathbf{B}^{-1}$. We know this maintains complementarity with current primal solutions.

Since Top Brass is a maximize primal, dual constraints for nonbasics variables are $\mathbf{vN} \geq \mathbf{c}^N$, or in terms of reduced costs, $\bar{\mathbf{c}}^N = \mathbf{c}^N - \bar{\mathbf{v}}\mathbf{N} \leq \mathbf{0}$. But this is exactly the test for absence of any improving simplex direction. For example, at iteration $t = 2$, nonbasic x_3 is chosen to enter because $\bar{x}_3 = c_3 - \bar{\mathbf{v}} \cdot \mathbf{a}^3 = 0 - (-6, 0, 0, 4.5) = 6 > 0$, i.e. because it represents a violation of dual feasibility. On the other hand at iteration $t = 3$, when reduced costs \bar{c}_j for both nonbasics are ≤ 0 , the algorithm terminates with a conclusion that the last primal solution $\mathbf{x}^{(3)} = (650, 1100, 350, 400, 0, 0)$ is optimal. Dual feasibility has been achieved. Primal simplex always maintains primal feasibility (principle [5.24]). Note also (principle [6.58]) that primal and dual objective values are kept equal at each step and complementary slackness is maintained. Thus when the algorithm stops, we have complementary feasible solutions for both primal and dual that share the same objective function value and are thus optimal in the respective problems (principle [6.50]).

Finally note that these Primal Simplex results also allow us to resolve an outstanding theoretical issue. Our discussion of KKT optimality conditions [6.56] above stated that the conditions are both necessary for LP optimality and sufficient. However, we were able at that time only to verify sufficiency. The realization that every feasible and bounded LP has an optimal basic solution that leads immediately to a complementary dual optimum with the same objective function value establishes that KKT conditions hold for every such LP.

Principle 6.60 If there is a primal optimal solution to a given linear program, then there is a basic one \mathbf{x}^* with corresponding dual optimal solution \mathbf{v}^* which has the same objective value and the two mutually satisfy complementary slackness. That is, KKT optimality conditions [6.56] are necessary for existence of optimal solutions to either primal or dual LPs.

EXAMPLE 6.37: PROVING KKT OPTIMALITY FROM AN OPTIMAL PRIMAL BASIS

Consider the following standard-form primal linear program

$$\begin{array}{ll} \min & 8x_1 + 10x_2 + 20x_3 + 2x_4 \\ \text{s.t.} & -2x_1 + 2x_2 + 2x_3 - 1x_4 = 8 \\ & 7x_1 + 1x_2 + 5x_3 + 3x_4 = 11 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

- Demonstrate that primal simplex algorithms could terminate with basis $\{x_2, x_4\}$.
- Formulate the dual of the given primal.
- Compute an optimal solution to the dual of (b) using only the results of (a).

Solution:

(a) The corresponding basic solution solves

$$\begin{aligned} 2\bar{x}_2 - 1\bar{x}_4 &= 8 \\ 1\bar{x}_2 + 3\bar{x}_4 &= 11 \end{aligned}$$

giving full primal basic solution $\bar{\mathbf{x}} = (0, 5, 0, 2)$, which is primal feasible. Corresponding simplex directions are $\Delta \mathbf{x}^{(1)} = (1, 13/7, 0, 12/7)$ with $\bar{c}_1 = 201/7$ and $\Delta \mathbf{x}^{(2)} = (0, 6/5, 1, -7/5)$ with $\bar{c}_2 = 146/5$. Neither improves for a minimize problem, so $\bar{\mathbf{x}}$ is optimal with objective value = 54.

(b) Using dual variables v_1, v_2 , the dual is

$$\begin{aligned} \max \quad & 8v_1 + 11v_2 \\ \text{s.t.} \quad & -2v_1 + 7v_2 \leq 8 \\ & 2v_1 + 1v_2 \leq 10 \\ & 2v_1 + 5v_2 \leq 20 \\ & -1v_1 + 3v_2 \leq 2 \\ & v_1, v_2 \text{ URS} \end{aligned}$$

(c) We can compute the corresponding dual solution by solving $\bar{\mathbf{v}}\mathbf{B} = \mathbf{c}^B$ on basic column, that is,

$$\begin{aligned} 2\bar{v}_1 + 1\bar{v}_2 &= 10 \\ -1\bar{v}_1 + 3\bar{v}_2 &= 2 \end{aligned}$$

The result is $\mathbf{v} = (4, 2)$ with objective value = 54. Reduced costs in (a) demonstrate the solution is dual feasible, and it achieves the primal optimal solution value. Thus it must satisfy complementary slackness conditions [6.50] as well, and both primal and dual solutions must be optimal.

6.8 DUAL SIMPLEX SEARCH

We have seen in principle [6.59] how Primal Simplex search can be interpreted as pursuing a strategy of maintaining primal feasibility while constructing complementary dual solutions and seeking their feasibility to complete KKT optimality conditions. This is not the only simplex path to LP optimality. In this section, we develop an alternative Dual Simplex search of improving dual solutions.

Principle 6.61 | Dual simplex algorithms can be understood as pursuing an improving dual search strategy through a series of basic solutions to fulfill KKT optimality conditions [6.56] by (i) starting with and maintaining dual feasibility of solutions visited, (ii) generating for each a primal basic solution that satisfies complementary slackness with the dual and facilitates direction choice, and (iii) terminating when dual unboundedness (primal infeasibility) is detected or the current primal basic solution proves primal feasible. That is, the algorithms maintain dual feasibility and complementary slackness while seeking primal feasibility.

One reason it might be attractive to approach LPs this way is that duals often have many fewer variables than primals. But more important is getting started. If a feasible primal basic solution is not apparent, there is often no choice but to use Phase I to find one, which may be an effort-consuming diversion. On the other hand, it is frequently rather easy to find a feasible dual from which to start, or restart in investigating “what if” changes as discussed in Sections 6.5 and 6.6. After all, we know that with standard form LPs (principle 6.56) that main dual variables \mathbf{v} are unrestricted. Dual feasibility then requires only $\mathbf{v}\mathbf{A} \geq \mathbf{c}$ for a maximize primal or $\mathbf{v}\mathbf{A} \leq \mathbf{c}$ for a minimize, which are equivalent to

$$\begin{aligned} \bar{\mathbf{c}} &= \mathbf{c} - \mathbf{v}\mathbf{A} \leq 0 \text{ for a maximizing primal, and} \\ \bar{\mathbf{c}} &= \mathbf{c} - \mathbf{v}\mathbf{A} \geq 0 \text{ for a minimizing primal} \end{aligned} \tag{6.11}$$

EXAMPLE 6.38: APPRECIATING THE CONVENIENCE OF THE DUAL SIMPLEX

Consider the following primal linear program

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 3x_1 - 2x_2 \geq 4 \\ & x_1 + 2x_2 \geq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- (a) Formulate the corresponding dual.
- (b) Place the primal in standard form by subtracting slack variables x_3 and x_4 .
- (c) Taking x_3 and x_4 as the basic variables, compute the primal basic solution and the corresponding dual.
- (d) Explain why this basis is not suitable for starting the Primal Simplex, but can readily begin the Dual Simplex.

Solution:

- (a) Using variables v_1 and v_2 , the dual problem is

$$\begin{aligned} \max \quad & 4v_1 + 3v_2 \\ \text{s.t.} \quad & 3v_1 + v_2 \leq 2 \\ & -2v_1 + 2v_2 \leq 3 \\ & v_1, v_2 \geq 0 \end{aligned}$$

- (b) Now placing the primal in standard form with two slack variables x_3 and x_4 and choosing them basic gives

	x_1	x_2	x_3	x_4	
min \mathbf{c}	2	3	0	0	\mathbf{b}
\mathbf{A}	3	-2	-1	0	4
	1	2	0	-1	3
$\bar{\mathbf{c}}_j$	2	3	0	0	
Basis:	N	N	1st	2nd	

(c) Solving for the primal and dual basic solution gives the primal infeasible result

$$\mathbf{B} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \bar{\mathbf{x}}^B = \mathbf{B}^{-1}\mathbf{b} = \begin{pmatrix} -4 \\ -3 \end{pmatrix}$$

(d) Still the complementary dual solution $\bar{\mathbf{v}} = \mathbf{c}^B \mathbf{B}^{-1} = (0, 0) \mathbf{B}^{-1} = (0, 0)$ is feasible because the only constraints in the dual for standard form for a minimize primal are equivalent to $\bar{\mathbf{c}} \geq 0$ (equation 6.11), which part (b) of the above table shows holds for the current dual. Starting Primal Simplex might well require Phase I, but we can start Dual Simplex immediately.

Choosing an Improving Direction

Each iteration of the Dual Simplex focuses on a component r of the current primal basic solution with $\bar{x}_r < 0$ and thus infeasible. A direction of change $\Delta \mathbf{v}$ in the current dual solution $\bar{\mathbf{v}}$ must be chosen that will produce progress in both primal and dual. This is achieved by leveraging the r th row of basis inverse \mathbf{B}^{-1} .

Principle 6.62 Dual simplex search adopts direction $\Delta \mathbf{v} \leftarrow \pm \mathbf{r}$ (+ for a maximizing primal, and $-$ for a minimizing primal), where \mathbf{r} is the row of \mathbf{B}^{-1} corresponding to infeasible basic value \bar{x}_r . This choice will improve the current dual objective function value while moving the associated primal solution closer to feasibility.

To illustrate, choose infeasible component $r = 1$ in the minimizing model of Example 6.38. The corresponding move direction of principle [6.62] would be

$$\Delta \mathbf{v} \leftarrow -\mathbf{r}^{(1)} = -(-1, 0) = (1, 0) \quad (6.12)$$

More generally, we know the current dual solution is feasible, so improving the dual objective value should lead to primal objective function values worse than the current solution (lower for a maximizing primal and higher for a minimizing one). The direction choice of principle [6.62] does exactly that. A step $\lambda \geq 0$ in direction $\Delta \mathbf{v} \leftarrow \pm \mathbf{r}$, will move dual solution value, and thus the complementary primal, as

$$\bar{\mathbf{v}}^{new} \mathbf{b} \leftarrow (\bar{\mathbf{v}}^{old} + \lambda \Delta \mathbf{v}) \cdot \mathbf{b} = (\bar{\mathbf{v}}^{old} \pm \lambda \mathbf{r}) \cdot \mathbf{b} = \mathbf{v}^{old} \cdot \mathbf{b} \pm \lambda \mathbf{r} \cdot \mathbf{b}$$

With r th primal basic $\bar{x}_r = \mathbf{r} \cdot \mathbf{b} < 0$, maximize primal solution values will decrease, which improves the associated minimizing dual, and minimize primals will increase while improving the associated maximizing dual.

In the case of (6.12) this would yield improving result

$$\bar{\mathbf{v}}^{new} \mathbf{b} \leftarrow (\bar{\mathbf{v}}^{old} + \lambda \Delta \mathbf{v}) \cdot \mathbf{b} = ((0, 0) - \lambda \mathbf{r}) \cdot \mathbf{b} = 0 - \lambda \mathbf{r} \cdot \mathbf{b} = -\lambda(-4) = 4\lambda$$

The maximizing dual will improve, and the corresponding primal will be adjusted toward feasibility.

Determining a Dual Step Size to Retain Dual Feasibility

The step size to apply to the direction of [6.62] follows from the need to retain dual feasibility (6.11).

Principle 6.63 A step $\lambda > 0$ in direction $\Delta \mathbf{v}$ will change current reduced costs by $\Delta \bar{\mathbf{c}} = -\Delta \mathbf{v} \mathbf{A}$. The limiting choice for λ will occur when dual feasibility is about to be lost on a nonbasic variable, that is,

$$\lambda \leftarrow \frac{-\bar{c}_p}{\Delta \bar{c}_p} = \min \left\{ \frac{-\bar{c}_j}{\Delta \bar{c}_j} : \Delta \bar{c}_j > 0, j \text{ nonbasic} \right\}$$

for a maximizing primal

$$\lambda \leftarrow \frac{\bar{c}_p}{-\Delta \bar{c}_p} = \min \left\{ \frac{\bar{c}_j}{-\Delta \bar{c}_j} : \Delta \bar{c}_j < 0, j \text{ nonbasic} \right\}$$

for a minimizing primal

If no limit is encountered, the dual is unbounded, which implies the primal is infeasible.

Returning to the minimizing example of Example 6.38 and equation (6.12), we have

$$\Delta \bar{\mathbf{c}} = -\Delta \mathbf{v} \mathbf{A} = -(1, 0) \begin{pmatrix} 3 & -2 & -1 & 0 \\ 1 & 2 & 0 & -1 \end{pmatrix} = (-3, 2, 1, 0)$$

With only one of the new nonbasic components < 0 , we obtain $\lambda = \bar{c}_1 / -\Delta \bar{c}_1 = 2/3$.

Changing the Primal Solution and Basis Update

With direction and step size established, we are ready to update the basis and corresponding primal and dual solutions.

Principle 6.64 Restoring a basic solution after updating dual solution $\bar{\mathbf{v}} \leftarrow \bar{\mathbf{v}} + \lambda \Delta \mathbf{v}$ is accomplished by replacing infeasible x_r in the basis by the x_p identified in step choice [6.63]. Updated primal and dual basic solutions will retain dual feasibility and complementary slackness while increasing previously negative x_r to $\mathbf{0}$.

Thinking first about dual feasibility, step size choice [6.63] will assure satisfaction on constraints (6.11) for currently nonbasic variables. What about the current basic variables? Requirements of complementarity make $\mathbf{c}^B = \bar{\mathbf{v}} \mathbf{B}$ at the start of the iteration, or $\bar{\mathbf{c}}^B = \mathbf{0}$. Then $\Delta \bar{\mathbf{c}}^B = -\Delta \mathbf{v} \mathbf{B} = \mp \mathbf{r} \mathbf{B}$. But with \mathbf{r} the r th row of the basic inverse, $\mp \mathbf{r} \mathbf{B} = (0, 0, \dots, \mp 1, \dots, 0, 0)$, that is, a unit vector with a -1 for maximizing or a $+1$ for minimizing primals on the component for x_r . Only \bar{c}_r will change with a step $\lambda > 0$, and the change will retain dual feasibility.

As for the primal and complementary slackness, step size choice [6.63] identifies a nonbasic variable x_p with updated $\bar{c}_p = 0$ after the move. Making variable $x_p > 0$ in a new basis would not violate complementarity. Furthermore, x_r now nonbasic makes its value $= 0$ which is complementary with its changed \bar{c}_r . It follows

that the new dual solution $\bar{\mathbf{v}} \leftarrow \bar{\mathbf{v}} + \lambda \Delta \mathbf{v}$ is exactly the complementary dual basic solution required for the next iteration.

Algorithm 6A combines all these steps in a full statement of the Dual Simplex algorithm.

ALGORITHM 6A: DUAL SIMPLEX SEARCH FOR LINEAR PROGRAMS

Step 0: Initialization. Choose any starting dual-feasible basis, partition the main constraint columns into basic submatrix \mathbf{B} and nonbasic submatrix \mathbf{N} , and derive a representation of inverse \mathbf{B}^{-1} . Then set nonbasic components of starting nonbasic primal solution $\mathbf{x}^{N(0)} \leftarrow 0$, compute basic components $\mathbf{x}^{B(0)}$ of the primal solution in by solving $\mathbf{B}\mathbf{x}^{B(0)} = \mathbf{b}$ as $\mathbf{x}^{B(0)} \leftarrow \mathbf{B}^{-1}\mathbf{b}$. Also, derive starting dual basic solution $\mathbf{v}^{(0)}$ by solving $\mathbf{v}^{(0)}\mathbf{B} = \mathbf{c}^B$ as $\mathbf{v}^{(0)} \leftarrow \mathbf{c}^B\mathbf{B}^{-1}$, where \mathbf{c}^B is the vector of current basic objective function coefficients, and initialize solution index $t \leftarrow 0$.

Step 1: Optimality. If the current primal basic solution satisfies $\mathbf{x}^{B(t)} \geq 0$, then stop; primal feasibility has been achieved, and current primal and dual solutions $\mathbf{x}^{(t)}$ and $\mathbf{v}^{(t)}$ are optimal in their respective problems with common objective value $\mathbf{c} \cdot \mathbf{x}^{(t)} = \mathbf{v}^{(t)} \cdot \mathbf{b}$. Otherwise choose a primal infeasible basic variable r , with $x_r^{(t)} < 0$.

Step 2: Dual Simplex Directions. Construct an improving dual simplex direction $\Delta \mathbf{v} \leftarrow \pm \mathbf{r}$ (+ for a maximizing primal and for a minimizing primal) where \mathbf{r} is the row of \mathbf{B}^{-1} corresponding to infeasible primal basic variable $x_r^{(t)}$. Also compute corresponding changes in nonbasic reduced costs $\Delta \bar{\mathbf{c}}^N \leftarrow -\Delta \mathbf{v}\mathbf{N}$.

Step 3. Step Size. If $\Delta \bar{\mathbf{c}}^N \geq 0$ for a minimizing primal or $\Delta \bar{\mathbf{c}}^N \leq 0$ for a maximizing primal, stop; progress is unlimited and the dual is unbounded, which implies the primal is infeasible. Otherwise, choose step size λ and nonbasic index p by

$$\lambda \leftarrow \frac{-\bar{c}_p}{\Delta \bar{c}_p} = \min \left\{ \frac{-\bar{c}_j}{\Delta \bar{c}_j} : \Delta \bar{c}_j > 0, j \text{ nonbasic} \right\}$$

for a maximizing primal

$$\lambda \leftarrow \frac{\bar{c}_p}{-\Delta \bar{c}_p} = \min \left\{ \frac{\bar{c}_j}{-\Delta \bar{c}_j} : \Delta \bar{c}_j < 0, j \text{ nonbasic} \right\}$$

for a minimizing primal

Step 4: New Solutions and Basis. Replace x_r in the basis by x_p , update submatrices \mathbf{B} , \mathbf{N} , and \mathbf{B}^{-1} , and compute new primal basic solution $\mathbf{x}^{(t+1)}$. The corresponding dual can be similarly recomputed or obtained by step $\mathbf{v}^{(t+1)} \leftarrow \mathbf{v}^{(t)} + \lambda \Delta \mathbf{v}$. Next advance $t \leftarrow t + 1$, and return to Step 1.

EXAMPLE 6.39: APPLYING DUAL SIMPLEX ALGORITHM 6A

Return to the example of Example 6.38.

- (a) Apply Algorithm 6A to compute optimal primal and dual basic solutions, showing quantities $\mathbf{x}^{(t)}$, $\mathbf{v}^{(t)}$, \mathbf{r} , $\Delta\mathbf{v}$, \mathbf{B} and \mathbf{B}^{-1} at each iteration t .
- (b) Illustrate progress of the algorithm in a plot of the original two variables, and comment on the evolution of primal and dual solutions.

Solution:

(a) Starting with x_3 and x_4 basic we have

	x_1	x_2	x_3	x_4	
min \mathbf{c}	2	3	0	0	\mathbf{b}
\mathbf{A}	3	-2	-1	0	4
	1	2	0	-1	3
\bar{c}_j	2	3	0	0	
Basis:	N	N	1st	2nd	

$$\mathbf{B} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\mathbf{v}^{(0)} = \mathbf{c}^B \mathbf{B}^{-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{x}^{B(0)} = \mathbf{B}^{-1} \mathbf{b} = \begin{pmatrix} -4 \\ -3 \end{pmatrix}$$

Both objection function values are $\mathbf{c}\mathbf{x}^{(0)} = \mathbf{v}^{(0)}\mathbf{b} = \mathbf{0}$.

Choosing infeasible $x_r = -4$, the improving direction is $\Delta\mathbf{v} = -\mathbf{r}^{(1)} = (1, 0)$ and $\Delta\bar{\mathbf{c}} = (-3, 2, 1, 0)$. Then step size is established on component x_1 by $\lambda = 2 / -(-3) = 2/3$, making $p = 1$.

At $t = 1$, we update the basis to

	x_1	x_2	x_3	x_4	
min \mathbf{c}	2	3	0	0	\mathbf{b}
\mathbf{A}	3	-2	-1	0	4
	1	2	0	-1	3
\bar{c}_j	0	13/3	2/3	0	
Basis:	1st	N	N	2nd	

$$\mathbf{B} = \begin{pmatrix} 3 & 0 \\ 1 & -1 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} 1/3 & 0 \\ 1/3 & -1 \end{pmatrix}$$

$$\mathbf{v}^{(1)} = \mathbf{c}^B \mathbf{B}^{-1} = \begin{pmatrix} 2/3 \\ 0 \end{pmatrix}, \mathbf{x}^{B(1)} = \mathbf{B}^{-1} \mathbf{b} = \begin{pmatrix} 4/3 \\ -5/3 \end{pmatrix}$$

Both objection function values are $\mathbf{c}\mathbf{x}^{(1)} = \mathbf{v}^{(1)}\mathbf{b} = 8/3$.

This time basic variable x_4 is infeasible, making $r = 4$, and $\Delta\mathbf{v} = -\mathbf{r}^{(2)} = (-1/3, 1)$ and $\Delta\bar{\mathbf{c}} = (0, -8/3, -1/3, 1)$. Step size rule [\[6.63\]](#) leaves $\lambda = \min \{ (13/3)/(8/3), (2/3)/(1/3) \} = 13/8$, and $p = 2$.

At $t = 2$, we update again to

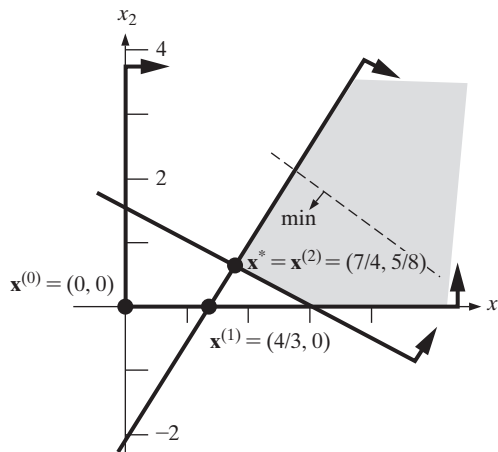
	x_1	x_2	x_3	x_4	
min \mathbf{c}	2	3	0	0	\mathbf{b}
\mathbf{A}	3	-2	-1	0	4
	1	2	0	-1	3
\bar{c}_j	0	0	1/8	13/8	
Basis:	1st	2nd	N	N	

$$\mathbf{B} = \begin{pmatrix} 3 & -2 \\ 1 & 2 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} 1/4 & 1/4 \\ -1/8 & 3/8 \end{pmatrix}$$

$$\mathbf{v}^{(2)} = \mathbf{c}^B \mathbf{B}^{-1} = \begin{pmatrix} 1/8 \\ 13/8 \end{pmatrix}, \mathbf{x}^{B(2)} = \mathbf{B}^{-1} \mathbf{b} = \begin{pmatrix} 7/4 \\ 5/8 \end{pmatrix}$$

Now the primal solution is fully feasible, so the algorithm terminates with $\mathbf{x}^* = (7/4, 5/8, 0, 0)$ and $\mathbf{v}^* = (1/8, 13/8)$, both with objective value $43/8$.

(b) Graphic solution is depicted below



Dual simplex solution begins at primal infeasible $\mathbf{x}^{(0)} = (0, 0)$, then proceeds to still infeasible $\mathbf{x}^{(1)} = (4/3, 0)$, and terminates optimal at the first primal feasible solution reached $\mathbf{x}^* = \mathbf{x}^{(2)} = (7/4, 5/8)$. Notice that each of the solutions is basic, defined by two active constraints. But only the last is an extreme-point of the feasible set.

6.9 PRIMAL-DUAL SIMPLEX SEARCH

We have seen in principle [6.59] how Primal Simplex search can be interpreted as pursuing a strategy of maintaining primal basic feasibility while constructing complementary basic dual solutions and seeking their feasibility to complete KKT optimality conditions. In principle [6.61] we encountered Dual Simplex search which

maintains dual feasible basic solutions, while updating the corresponding complementary primal basic solution, until primal feasibility is attained. Here we explore a variant known as **Primal-Dual search** which maintains dual feasible solutions that are not necessarily basic, but adds new flexibility in the search for a complementary primal feasible solution to go with it.

Principle 6.65 Primal-dual simplex algorithms can be understood as pursuing an improving primal-dual search strategy through a series of solution pairs seeking to fulfill KKT optimality conditions [6.54] by (i) starting with and maintaining dual feasibility of solutions visited (although not necessarily basic dual solutions), (ii) seeking a primal feasible basic solution among those of **restricted primals** admitting all that satisfy complementary slackness with the each dual, and when blocked, generating an improving direction for the dual, then (iii) terminating when dual unboundedness (primal infeasibility) is detected or the current primal basic solution proves primal feasible. That is, the algorithms maintain dual feasibility and complementary slackness while seeking primal feasibility in a new way.

Like the Dual Simplex, part of the appeal of Primal-Dual Simplex is that only a dual feasible solution is required to get started. In the Primal-Dual case, even more freedom is allowed because the dual solution need not even be basic. In standard form LPs (principle [6.53]), which are the focus of this section, main dual variables \mathbf{v} are unrestricted. Dual feasibility requires only $\mathbf{v}\mathbf{A} \geq \mathbf{c}$ for a maximize primal or $\mathbf{v}\mathbf{A} \leq \mathbf{c}$ for a minimize, which are equivalent to $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{v}\mathbf{A} \leq 0$, and $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{v}\mathbf{A} \geq 0$, respectively.

The other issue is flexibility and efficiency in discovering appropriate primal solutions through a Restricted Primal strategy.

Principle 6.66 Each major step of primal-dual search solves a **restricted primal** model in the form

$$\begin{aligned} \min \quad & \sum_i q_i \\ \text{s.t.} \quad & \sum_{j \in J} \mathbf{a}^j x_j + \mathbf{q} = \mathbf{b} \\ & \mathbf{q} \geq \mathbf{0}, x_j \geq \mathbf{0} \text{ for all } j \in J \end{aligned}$$

Here \mathbf{q} is a vector of artificial variables, and $J = \{j : c_j - \bar{\mathbf{v}}\mathbf{a}^j = \bar{c}_j = 0\}$ for current dual feasible solution $\bar{\mathbf{v}}$.

As detailed in principle [6.66], the restricted primal strategy brings back Phase I Primal Simplex search (Section 3.5) in a new way dealing with fewer primal variables. A primal search ensues at each major step over just the subset of complementary x_j with $\bar{c}_j = 0$ to find a solution that is primal feasible ($\bar{\mathbf{q}}$, the artificial optimum, = $\mathbf{0}$). Dual change, which has to consider the \bar{c}_j limits on every primal variable, arises only periodically when the restricted problem fails. Although we will limit our discussion here to simplex-based methods for dealing with restricted

primals, many others are possible for special LPs (e.g., the Hungarian Assignment Algorithm of Chapter 10).

EXAMPLE 6.40: STARTING PRIMAL-DUAL SIMPLEX SEARCH

Consider the following standard-form primal linear program

$$\begin{aligned} \min \quad & 2x_1 + 1x_2 + 4x_3 + 11x_4 \\ \text{s.t.} \quad & 2x_1 + 1x_2 + 2x_3 + 4x_4 = 4 \\ & + 1x_2 + 3x_3 + 6x_4 = 5 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

- Formulate the corresponding dual over variables v_1 and v_2 .
- Demonstrate that solution $\bar{\mathbf{v}} = (1, 0)$ is feasible in the dual.
- Compute reduced costs \bar{c}_j for all primal variables, and identify those qualifying for restricted primal set J .
- Formulate the corresponding restricted primal problem.

Solution:

- Using variables v_1 and v_2 , the dual problem is

$$\begin{aligned} \max \quad & 4v_1 + 5v_2 \\ \text{s.t.} \quad & 2v_1 \leq 2 \\ & 1v_1 + 1v_2 \leq 1 \\ & 2v_1 + 3v_2 \leq 4 \\ & 4v_1 + 6v_2 \leq 11 \\ & v_1, v_2 \text{ URS} \end{aligned}$$

- Substituting $v_1 = 1, v_2 = 0$ satisfies all 4 main constraints, so it is dual feasible.

(c) Using $v_1 = 1$ and $v_2 = 0$ to weight the main constraint rows, corresponding reduced costs are $\bar{c}_1 = 0, \bar{c}_2 = 0, \bar{c}_3 = 2, \bar{c}_5 = 6$. This makes restricted set $J = \{1, 2\}$, those with $\bar{\mathbf{c}} = 0$.

- Now restricting original variables to those in J , and introducing artificial variables q_1 and q_2 , the starting restricted primal is

$$\begin{aligned} \min \quad & q_1 \\ \text{s.t.} \quad & 2x_1 + 1x_2 + q_1 = 4 \\ & + 1x_2 + q_2 = 5 \\ & x_1, x_2, q_1, q_2 \geq 0 \end{aligned}$$

Choosing an Improving Dual Direction

Each major round of primal-dual search involves solving the current restricted primal, often passing through a number of primal solutions to reach phase I optimality. If the optimal $\bar{\mathbf{q}} = 0$, a primal feasible and complementary solution is at hand, and the

whole search can terminate. Otherwise, we must find a way to change main dual solution \mathbf{v} to improve its objective value and admit new variables to the restricted set.

Principle 6.67 If optimality is obtained in the current restricted primal problem without reaching primal feasibility over the original variables, primal-dual simplex search adopts direction $\Delta \mathbf{v} \leftarrow \pm \bar{\mathbf{w}}$ (+ for a minimizing primal, and – for a maximizing primal), where $\bar{\mathbf{w}}$ is an optimal dual solution to the current restricted primal. This direction will improve the dual solution value in the original model and offer the opportunity to add new variables to the next restricted primal.

To see that this choice of $\Delta \mathbf{v}$ improves the main dual objective value, note that

$$\begin{pmatrix} \text{new dual} \\ \text{objective} \\ \text{value} \end{pmatrix} = (\bar{\mathbf{v}} + \lambda \Delta \mathbf{v}) \mathbf{b} = \bar{\mathbf{v}} \cdot \mathbf{b} + \lambda \Delta \cdot \mathbf{b} = \begin{pmatrix} \text{old dual} \\ \text{objective} \\ \text{value} \end{pmatrix} \pm \lambda \bar{\mathbf{w}} \cdot \mathbf{b}$$

But $\bar{\mathbf{w}} \cdot \mathbf{b}$ is exactly the optimal solution value of the last restricted primal, and it is > 0 if nonzero artificials remain in its optimal solution.

Determining a Dual Step Size

We would like to pursue this improving direction $\Delta \mathbf{v} = \pm \bar{\mathbf{w}}$ of principle [6.67](#) as long as the modified $\bar{\mathbf{v}}$ remains dual feasible. If there is a limit, we will see it is reached exactly when one or more new primal variables are added to the restricted primal.

The step size to apply to the direction of [6.67](#) follows from the need to retain dual feasibility.

Principle 6.68 A step $\lambda > 0$ in direction $\Delta \mathbf{v}$ will add $-\Delta \mathbf{v} \cdot \mathbf{a}^j$ to the current reduced cost \bar{c}_j on each primal variable x_j . The limiting choice for λ can occur only when dual feasibility is about to be lost on a variable x_p not part of the current restricted primal, that is,

$$\lambda \leftarrow \frac{-\bar{c}_p}{-\Delta \mathbf{v} \cdot \mathbf{a}^p} = \min \left\{ \frac{-\bar{c}_j}{-\Delta \mathbf{v} \cdot \mathbf{a}^j} : \Delta \mathbf{v} \cdot \mathbf{a}^j < 0, j \notin J \right\}$$

for a maximizing primal

$$\lambda \leftarrow \frac{\bar{c}_p}{\Delta \mathbf{v} \cdot \mathbf{a}^p} = \min \left\{ \frac{-\bar{c}_j}{-\Delta \mathbf{v} \cdot \mathbf{a}^j} : \Delta \mathbf{v} \cdot \mathbf{a}^j > 0, j \notin J \right\}$$

for a minimizing primal

The next restricted primal will add those blocking variables to J and retain any variables that were basic in the restricted optimum. Otherwise, if no limit is encountered, the dual is unbounded, which implies the primal is infeasible.

To understand why rule [6.68](#) works, consider the case of a minimizing primal. Then feasibility of the current dual solution assures all $\bar{c}_j \geq 0$. The step limit arises over those j with positive reduced cost and change < 0 meaning $\Delta \mathbf{v} \cdot \mathbf{a}^j > 0$. Now, dual

optimality in the restricted (always minimizing) primal, where all costs on original variables = 0, demands phase I reduced costs = $-\bar{\mathbf{w}} \cdot \mathbf{a}^j = -\Delta \mathbf{v} \cdot \mathbf{a}^j \geq 0$ for all $j \in J$. Furthermore that restricted problem reduced cost must = 0 for those j in the optimal restricted solution. We can conclude that all of those J -variables in the reduced problem will have $\bar{c}_j \geq 0$ not declining after the dual change. In particular, those that were in the optimal restricted primal solution will continue to have $\bar{c}_j = 0$ and remain a part of the next reduced model. Only $j \in J$ can limit the step. When one or more of them reaches $\bar{c}_j = 0$, those new variables can be added to the next restricted primal. Otherwise, there is no limit on dual improvement and the primal is infeasible (principle [6.48](#)).

All the elements are now in place to detail the Primal-Dual Simplex Algorithm 6B. It will continue until either some restricted primal produces a primal feasible solution to complete the primal and dual pair, or primal infeasibility (dual unboundedness) is established.

ALGORITHM 6B: PRIMAL-DUAL SIMPLEX SEARCH FOR LP

Step 0: Initialization. Place the given primal model in standard form. Then choose any dual feasible $\mathbf{v}^{(0)}$ (i.e., $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{v}^{(0)}\mathbf{A} \leq 0$ for a maximize primal and $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{v}^{(0)}\mathbf{A} \geq 0$ for a minimize) that also makes at least one corresponding primal reduced cost $\bar{c}_j = 0$, and set $t \leftarrow 0$.

Step 1: Restricted Primal. Construct the restricted primal [6.66](#) over $j \in J = \{j : \bar{c}_j = 0\}$ for current dual solution $\mathbf{v}^{(t)}$, and solve it to optimality for solution $\bar{\mathbf{x}}, \bar{\mathbf{q}}$.

Step 2: Optimality. If the optimal artificial variables $\bar{\mathbf{q}} = \mathbf{0}$ in the restricted primal, stop. Restricted primal optimum $\bar{\mathbf{x}}$ (including $\bar{x}_j \leftarrow 0$ for $j \notin J$) and the current $\mathbf{v}^{(t)}$ are respectively primal and dual optimal in the original problem with common objective value $\mathbf{c} \cdot \bar{\mathbf{x}} = \mathbf{v}^{(t)} \cdot \mathbf{b}$.

Step 3: Dual Simplex Directions. Construct an improving dual simplex direction $\Delta \mathbf{v} \leftarrow \pm \bar{\mathbf{w}}$ (+ for a minimizing primal and – for a maximizing primal) where $\bar{\mathbf{w}}$ is an optimal dual solution to the restricted primal of Step 1.

Step 4: Step Size. If $\Delta \mathbf{v} \cdot \mathbf{a}^j \geq 0$ for all $j \notin J$ of a maximize primal, or all $\Delta \mathbf{v} \cdot \mathbf{a}^j \leq 0$ for a minimize, stop; progress is unlimited and the dual is unbounded which implies the primal is infeasible. Otherwise pick step size $\lambda > 0$ according to

$$\lambda \leftarrow \frac{-\bar{c}_p}{-\Delta \mathbf{v} \cdot \mathbf{a}^p} = \min \left\{ \frac{-\bar{c}_j}{-\Delta \mathbf{v} \cdot \mathbf{a}^j} : \Delta \mathbf{v} \cdot \mathbf{a}^j < 0, j \notin J \right\}$$

for a maximizing primal

$$\lambda \leftarrow \frac{\bar{c}_p}{\Delta \mathbf{v} \cdot \mathbf{a}^p} = \min \left\{ \frac{\bar{c}_j}{\Delta \mathbf{v} \cdot \mathbf{a}^j} : \Delta \mathbf{v} \cdot \mathbf{a}^j > 0, j \notin J \right\}$$

for a minimizing primal

Step 5: New Solutions. Advance $\mathbf{v}^{(t+1)} \leftarrow \mathbf{v}^{(t)} + \lambda \Delta \mathbf{v}$, update J to include all x_j with modified $\bar{c}_j = 0$, increment $t \leftarrow t + 1$ and return to Step 1.

EXAMPLE 6.41: APPLYING PRIMAL-DUAL SIMPLEX ALGORITHM 6B

Return to the LP of Example 6.40. Starting from the same dual $\mathbf{v}^{(0)} = (1, 0)$, apply Algorithm 6B to compute optimal primal and dual solutions, showing quantities \mathbf{x} , \mathbf{v} , \mathbf{J} , $\Delta \mathbf{v}$, \mathbf{B} and \mathbf{B}^{-1} as they change. Also show each restricted primal solved.

Solution:

(a) For simplicity, the solution will be tracked in tabular form. Variables not available in particular restricted primals are shaded in gray. New vector \mathbf{d} is used to denote the restricted problem (Phase I) objective, as opposed from the original \mathbf{c} .

$t = 0$	x_1	x_2	x_3	x_4	q_1	q_2	
min $\mathbf{c} =$	2	1	4	11	-	-	
min $\mathbf{d} =$	0	0	0	0	1	1	\mathbf{b}
$\mathbf{A} =$	2	1	2	4	1	0	4
	0	1	3	6	0	1	5
$\bar{\mathbf{c}} =$	0	0	2	7			$J = \{1, 2\}$
basis =	1st	N			N	$2nd$	
	$\mathbf{B} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{B}^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix}$ $\mathbf{d}^B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \bar{\mathbf{w}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$						
$\bar{\mathbf{x}}, \bar{\mathbf{q}} =$	2	0	0	0	0	5	
$\mathbf{d} =$	0	-1	-3	-6	1	0	
$\Delta \mathbf{x}, \Delta \mathbf{q} =$	-0.5	1	0	0	0	-1	
step =	2/0.5	-			-	5/1	
$\bar{\mathbf{x}}, \bar{\mathbf{q}} =$	0	4	0	0	0	1	
basis =	N	1st	-	-	N	$2nd$	
	$\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$ $\mathbf{d}^B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \bar{\mathbf{w}} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$						
$\mathbf{d} =$	2	0	-1	-2	2	0	optimal

(b) We begin as in Example 6.41 with $\mathbf{v}^{(0)} = (1, 0)$ and objective value $\mathbf{v}^{(0)} \cdot \mathbf{b} = 4$. As often happens, this dual choice allows more than one primal variable to join the restricted problem. Here, there are two with $J = \{1, 2\}$. In combination with the artificials, these present many options for a starting basis in the restricted problem. Computations above choose x_1 and q_2 . Corresponding basis elements in the table lead to a restricted problem solution of $\bar{x}_2 = 2, \bar{x}_3 = 0, \bar{q}_1 = 0$, and $\bar{q}_5 = 5$, but pricing nonbasic columns with \bar{d}_j s demonstrates nonbasic x_2 should enter.

An appropriate step in the corresponding simplex direction leads to new basis x_2, q_2 with restricted primal solution $\bar{x}_1 = 0, \bar{x}_2 = 4, \bar{q}_1 = 0$, and $\bar{q}_2 = 1$. Updated basis elements now show the last solution is optimal in the restricted problem. All \bar{d}_j are nonnegative. However, artificial variable \bar{q}_2 still remains

positive, which establishes that a complementary x -solution cannot be constructed using only the variables now available in the restricted problem. Dual solution \mathbf{v} must be changed.

Following principle [6.67], the direction of dual improvement is derived from the optimal dual solution in the last restricted problem, that is, $\Delta \mathbf{v} \leftarrow \bar{\mathbf{w}} = (-1, 1)$. Applying step rule [6.68], there are two variables x_3 and x_4 not now in J , and both have $\Delta \mathbf{v} \cdot \mathbf{a}^j > 0$. Testing ratios with corresponding \bar{c}_j gives $\lambda = \min\{2/1, 6/2\} = 2$. The result is $\mathbf{v}^{(1)} = \mathbf{v}^{(0)} + \lambda \Delta \mathbf{v} = (1, 0) + 2(-1, 1) = (-1, 2)$ with improved objective value $\mathbf{v}^{(1)} \cdot \mathbf{b} = 6$.

(c) As shown in the table below, updated \bar{c}_j now admit x_3 to the restricted problem. Notice that previously basic x_2 remains because the chosen dual change has net effect = 0 on its reduced cost. However, nonbasic x_1 has dropped out of the restricted primal, which creates no difficulty since its last value was = 0.

We restart directly from the last restricted primal basic solution, with newly added variables nonbasic. Dual solution $\bar{\mathbf{w}}$ and corresponding \bar{d}_j are unchanged. Now that x_3 has joined the restricted problem, it qualifies to enter the basis with $\bar{d}_3 = -1$. Applying the usual simplex direction and step size rules leads to new restricted primal solution $\bar{x}_2 = 2, \bar{x}_3 = 1, \bar{q}_1 = 0$, and $\bar{q}_2 = 0$.

$t = 1$	x_1	x_2	x_3	x_4	q_1	q_2	$\mathbf{v}^{(1)}$
$\bar{c} =$	4	0	0	3	–	–	–1
basis =		1st	N		N	2nd	2
$\Delta \mathbf{x}, \Delta \mathbf{q} =$	0	–2	1	0	0	–1	$J = \{2, 3\}$
step =		4/2	–		–	1/1	
$\bar{\mathbf{x}}, \bar{\mathbf{q}} =$	0	2	1	0	0	0	feasible

This time a solution to the restricted primal has been reached with all artificial variables = 0. The corresponding optimal $\bar{\mathbf{x}}$ is optimal in the original primal because it is feasible and complementary with the last \mathbf{v} . Filling in 0s on primal variables not in the restricted problem, we have $\mathbf{x}^* = (0, 2, 1, 0)$ and $\mathbf{v}^* = (-1, 2)$, both with objective value = 6.

EXERCISES

6-1 As a result of a recent decision to stop production of toy guns that look too real, the Super Slayer Toy Company is planning to focus its production on two futuristic models: beta zappers and freeze phasers. Beta zappers produce \$2.50 in profit for the company, and freeze phasers, \$1.60. The company is contracted to sell 10 thousand beta zappers and 15 thousand freeze phasers in the next month, but all that are produced can be sold. Production of either model involves three crucial steps: extrusion, trimming, and assembly. Beta zappers use 5 hours of extrusion time per thousand units, 1 hour of trimming time, and 12 hours of assembly. Corresponding values per thousand units of freeze phasers are 9, 2, and 15.

There are 320 hours of extrusion time, 300 hours of trimming time, and 480 hours of assembly time available over the next month. (An optimization output appears in Table 6.6.)

(a) Briefly explain how this problem can be modeled by the linear program:

$$\begin{aligned}
 \max \quad & 2500x_1 + 1600x_2 \\
 \text{s.t.} \quad & x_1 \geq 10 \\
 & x_2 \geq 15 \\
 & 5x_1 + 9x_2 \leq 320 \\
 & 1x_1 + 2x_2 \leq 300 \\
 & 12x_1 + 15x_2 \leq 480 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

TABLE 6.6 Optimization Output for Super Slayer Exercise 6-1

Solution value (max) = 77125.000								
VARIABLE SENSITIVITY ANALYSIS:								
Name	Optimal Value	Bas Sts	Lower Bound	Upper Bound	Object Coef	Reduced Object	Lower Range	Upper Range
x1	21.250	BAS	0.000	+infin	2500.00	0.000	1280.000	+infin
x2	15.000	BAS	0.000	+infin	1600.00	0.000	-infin	3125.000
CONSTRAINT SENSITIVITY ANALYSIS:								
Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range		
c1	G	0.000	10.000	11.250	-infin	21.250		
c2	G	-1525.000	15.000	-0.000	0.000	24.000		
c3	L	-0.000	320.000	78.750	241.250	+infin		
c4	L	-0.000	300.000	248.750	51.250	+infin		
c5	L	208.333	480.000	0.000	345.000	669.000		

- ✓ (b) Identify the resource associated with the objective function and each main constraint in part (a).
- ✓ (c) Identify the activity associated with each decision variable in part (a).
- ✓ (d) Interpret the left-hand-side coefficients of each decision variable in part (a) as inputs and outputs of resources per unit activity.

batch, requires 9 and 2 tons of the ingredients, respectively, and yields 1 ton of product. Orchid wants to find the least costly way to produce at least 50 tons of the new product given that there are 75 tons of ingredient 1 and 60 tons of ingredient 2 on hand. (An optimization output appears in Table 6.7.)

(a) Briefly explain how this problem can be modeled by the LP

$$\begin{aligned}
 \min \quad & 14x_1 + 30x_2 + 11x_3 \\
 \text{s.t.} \quad & 2x_1 + 5x_2 + 1x_3 \geq 50 \\
 & 3x_1 + 2x_2 + 9x_3 \leq 75 \\
 & 1x_1 + 7x_2 + 2x_3 \leq 60 \\
 & x_1, x_2, x_3 \geq 0
 \end{aligned}$$

(b) through (d) as in Exercise 6-1.

6-2 Eli Orchid can manufacture its newest pharmaceutical product in any of three processes. One costs \$14,000 per batch, requires 3 tons of one major ingredient and 1 ton of the other, and yields 2 tons of output product. The second process costs \$30,000 per batch, requires 2 and 7 tons of the ingredients, respectively, and yields 5 tons of product. The third process costs \$11,000 per

TABLE 6.7 Optimization Output for Eli Orchid Exercise 6-2

Solution value (min) = 311.111								
VARIABLE SENSITIVITY ANALYSIS:								
Name	Optimal Value	Bas Sts	Lower Bound	Upper Bound	Object Coef	Reduced Object	Lower Range	Upper Range
x1	5.556	BAS	0.000	+infin	14.000	0.000	12.000	+infin
x2	7.778	BAS	0.000	+infin	30.000	0.000	-infin	35.000
x3	0.000	NBL	0.000	+infin	11.000	5.667	5.333	+infin
CONSTRAINT SENSITIVITY ANALYSIS:								
Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range		
c1	G	7.556	50.000	-0.000	42.857	70.263		
c2	L	0.000	75.000	42.778	32.222	+infin		
c3	L	-1.111	60.000	0.000	25.000	70.000		

6-3 Professor Proof is trying to arrange for the implementation in a computer program of his latest operations research algorithm. He can contract with any mix of three sources for help: unlimited hours from undergraduates at \$4 per hour, up to 500 hours of graduate students at \$10 per hour, or unlimited hours of professional programmers at \$25 per hour. The full project would take a professional at least 1000 hours, but grad students are only 0.3 as productive, and undergraduates, 0.2. Proof has only 164 hours of his own time to devote to the effort, and he knows from experience that undergraduate programmers require more supervision than graduates, and graduates more than professionals. In particular, he estimates that he will have to invest 0.2 hour of

his own time per hour of undergraduate programming, 0.1 hour of his time per hour of graduate programming, and 0.05 hour of his time per hour of professional programming. (An optimization output appears in Table 6.8.)

(a) Briefly explain how this problem can be modeled by the LP

$$\begin{aligned} \min \quad & 4x_1 + 10x_2 + 25x_3 \\ \text{s.t.} \quad & 0.2x_1 + 0.3x_2 + x_3 \geq 1000 \\ & 0.2x_1 + 0.1x_2 + 0.05x_3 \leq 164 \\ & x_2 \leq 500 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

✓ (b) through (d) as in Exercise 6-1.

TABLE 6.8 Optimization Output for Professor Proof Exercise 6-3

Solution value (min) = 24917.647								
VARIABLE SENSITIVITY ANALYSIS:								
Name	Optimal Value	Bas Sts	Lower Bound	Upper Bound	Object Coef	Reduced Object	Lower Range	Upper Range
x1	82.353	BAS	0.000	+infin	4.000	0.000	-4.444	5.000
x2	0.000	NBL	0.000	+infin	10.000	2.235	7.765	+infin
x3	983.529	BAS	0.000	+infin	25.000	0.000	20.000	31.333
CONSTRAINT SENSITIVITY ANALYSIS:								
Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range		
c1	G	25.882	1000.000	-0.000	164.000	1093.333		
c2	L	-5.882	164.000	0.000	150.000	1000.00		
c3	L	0.000	500.000	500.000	0.000	+infin		

6-4 The NCAA is making plans for distributing tickets to the upcoming regional basketball championships. The up to 10,000 available seats will be divided between the media, the competing universities, and the general public. Media people are admitted free, but the NCAA receives \$45 per ticket from universities and \$100 per ticket from the general public. At least 500 tickets must be reserved for the media, and at least half as many tickets should go to the competing universities as to the general public. Within these restrictions, the NCAA wishes to find the allocation that raises the most money. An optimization output appears in Table 6.9.

(a) Briefly explain how this problem can be modeled by the LP

$$\begin{aligned} \max \quad & 45x_2 + 100x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \leq 10,000 \\ & x_2 - \frac{1}{2}x_3 \geq 0 \\ & x_1 \geq 500 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

(b) through (d) as in Exercise 6-1.

6-5 For each of the following constraint coefficient changes, determine whether the change would tighten or relax the feasible set, whether any implied change in the optimal value would be an increase or a decrease, and whether the rate of any such optimal value effect would become more or less steep if it varied with the magnitude of coefficient change. Assume that the model is

TABLE 6.9 Optimization Output for NCAA Ticket for Exercise 6-4

Solution value (max) = 775833.333								
VARIABLE SENSITIVITY ANALYSIS:								
Name	Optimal Value	Bas Sts	Lower Bound	Upper Bound	Object Coef	Reduced Object	Lower Range	Upper Range
x1	500.000	BAS	0.000	+infin	0.000	0.000	-infin	81.667
x2	3166.667	BAS	0.000	+infin	45.000	0.000	-200.000	100.000
x3	6333.333	BAS	0.000	+infin	100.000	0.000	45.000	+infin

CONSTRAINT SENSITIVITY ANALYSIS:							
Name	Typ	Optimal Dual	RHS Coef	Slack	Lower Range	Upper Range	
c1	L	81.667	10000.000	0.000	500.000	+infin	
c2	G	-36.667	0.000	-0.000	-4750.000	9500.000	
c3	G	-81.667	500.000	-0.000	0.000	10000.000	

a linear program, that all variables are nonnegative, and that the constraint is not the only one.

- ✓ (a) Maximize problem, $3w_1 + w_2 \geq 9$, increase 9.
- (b) Minimize problem, $5w_1 - 2w_2 \geq 11$, increase 11.
- ✓ (c) Minimize problem, $4w_1 - 3w_2 \geq 15$, decrease 15.
- (d) Maximize problem, $3w_1 + 4w_2 \leq 17$, increase 4.
- ✓ (e) Maximize problem, $3w_1 + 1w_2 \geq 9$, increase 3.
- (f) Minimize problem, $5w_1 - 2w_2 \leq 11$, increase -2.
- ✓ (g) Minimize problem, $4w_1 - 3w_2 \geq 15$, decrease -3.
- (h) Maximize problem, $3w_1 + 4w_2 \leq 17$, increase 3.

6-6 Determine whether adding each of the following constraints to a mathematical program would tighten or relax the feasible set and whether any implied change in the optimal value would be an increase or a decrease. Assume that the constraint is not the only one.

- ✓ (a) Maximize problem, $2w_1 + 4w_2 \geq 10$.
- (b) Maximize problem, $14w_1 - w_2 \geq 20$.
- ✓ (c) Minimize problem, $45w_1 + 34w_2 \leq 77$.
- (d) Minimize problem, $32w_1 + 67w_2 \leq 49$.

6-7 For each of the following objective coefficient changes, determine whether any implied change in the optimal value would be an increase or a decrease and whether the rate of any such

optimal value effect would become more or less steep if it varied with the magnitude of coefficient change. Assume that the model is a linear program and that all variables are nonnegative.

- ✓ (a) $\max 13w_1 + 4w_2$, increase 13.
- (b) $\max 5w_1 - 10w_2$, decrease 5.
- ✓ (c) $\min -5w_1 + 17w_2$, increase -5.
- (d) $\min 29w_1 + 14w_2$, decrease 14.

6-8 Return to Super Slayer Exercise 6-1.

- ✓ (a) Assign dual variables to each main constraint of the formulation in part (a), and define their meanings and units of measurement.
- ✓ (b) Show and justify the appropriate variable-type restrictions on all dual variables.
- ✓ (c) Formulate and interpret the main dual constraint corresponding to each primal variable.
- ✓ (d) Formulate and interpret an appropriate dual objective function.
- (e) Use optimal solutions in Table 6.6 to verify that optimal primal and dual objective function values are equal.
- ✓ (f) Formulate and interpret all primal complementary slackness conditions for the model.
- ✓ (g) Formulate and interpret all dual complementary slackness conditions for the model.
- (h) Verify that optimal primal and dual solutions in Table 6.6 satisfy the complementary slackness conditions of parts (f) and (g).

6-9 Do Exercise 6-8 for the problem of Exercise 6-2 using Table 6.7.

✓ **6-10** Do Exercise 6-8 for the problem of Exercise 6-3 using Table 6.8.

6-11 Do Exercise 6-8 for the problem of Exercise 6-4 using Table 6.9.

6-12 State the dual of each of the following LPs.

✓ (a) min $17x_1 + 29x_2 + x_4$
 s.t. $2x_1 + 3x_2 + 2x_3 + 3x_4 \leq 40$
 $4x_1 + 4x_2 + x_4 \geq 10$
 $3x_3 - x_4 = 0$
 $x_1, \dots, x_4 \geq 0$

(b) min $44x_1 - 3x_2 + 15x_3 + 56x_4$
 s.t. $x_1 + x_2 + x_3 + x_4 = 20$
 $x_1 - x_2 \geq 0$
 $9x_1 - 3x_2 + x_3 - x_4 \leq 25$
 $x_1, \dots, x_4 \geq 0$

✓ (c) max $30x_1 - 2x_3 + 10x_4$
 s.t. $2x_1 - 3x_2 + 9x_4 \leq 10$
 $4x_2 - x_3 \geq 19$
 $x_1 + x_2 + x_3 = 5$
 $x_1 \geq 0, x_3 \leq 0$

(d) max $5x_1 + x_2 - 4x_3$
 s.t. $x_1 + x_2 + x_3 + x_4 = 19$
 $4x_2 + 8x_4 \geq 55$
 $x_1 + 6x_2 - x_3 \leq 7$
 $x_2, x_3 \geq 0, x_4 \leq 0$

✓ (e) max $2x_1 + 9x_2$
 s.t. $3w + 2x_1 - x_2 \geq 10$
 $w - y \leq 0$
 $x_1 + 3x_2 + y = 11$
 $x_1, x_2 \geq 0$

(f) min $19y_1 + 4y_2 - 8z_2$
 s.t. $11y_1 + y_2 + z_1 = 15$
 $z_1 + 5z_2 \geq 0$
 $y_1 - y_2 + z_2 \leq 4$
 $y_1, y_2 \geq 0$

✓ (g) min $32x_2 + 50x_3 - 19x_5$
 s.t. $(15 \sum_{j=1}^3 x_j) + x_5 = 40$
 $12x_1 - 90x_2 + 14x_4 \geq 18$

$$x_4 \leq 11$$

$$x_j \geq 0, j = 1, \dots, 5$$

(h) min $10(x_3 + x_4)$
 s.t. $\sum_{j=1}^4 x_j = 400$
 $x_j - 2x_{j+1} \geq 0 \quad j = 1, \dots, 3$
 $x_1, x_2 \geq 0$

6-13 State (primal and dual) complementary slackness conditions for each LP in Exercise 6-12.

6-14 Each of the following LP has a finite optimal solution. State the corresponding dual, solve both primal and dual graphically, and verify that optimal objective function values are equal.

✓ (a) max $14x_1 + 7x_2$
 s.t. $2x_1 + 5x_2 \leq 14$
 $5x_1 + 2x_2 \leq 14$
 $x_1, x_2 \geq 0$

(b) min $4x_1 + 10x_2$
 s.t. $2x_1 + x_2 \geq 6$
 $x_1 \geq 1$
 $x_1, x_2 \geq 0$

✓ (c) min $8x_1 + 11x_2$
 s.t. $2x_1 + 9x_2 \geq 24$
 $3x_1 + x_2 \geq 11$
 $x_1, x_2 \geq 0$

(d) max $7x_1$
 s.t. $4x_1 + 2x_2 \leq 7$
 $3x_1 + 7x_2 \leq 14$
 $x_1, x_2 \geq 0$

6-15 Compute the dual solution corresponding to each of the following basic sets in the standard-form LP

$$\max \quad 6x_1 + 1x_2 + 21x_3 - 54x_4 - 8x_5$$

$$\text{s.t.} \quad 2x_1 + 5x_3 + 7x_5 = 70$$

$$+ 3x_2 + 3x_3 - 9x_4 + 1x_5 = 1$$

$$x_1, \dots, x_5 \geq 0$$

✓ (a) $\{x_1, x_2\}$

(b) $\{x_1, x_4\}$

✓ (c) $\{x_2, x_3\}$

(d) $\{x_3, x_5\}$

6-16 Each of the following is a linear program with no optimal solution. State the corresponding

dual, solve both primal and dual graphically, and verify that whenever primal or dual is unbounded, the other is infeasible.

$$\begin{aligned} \checkmark \text{ (a) max} \quad & 4x_1 + x_2 \\ \text{s.t.} \quad & 2x_1 + x_2 \geq 4 \\ & 3x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(b) max} \quad & 4x_1 + 8x_2 \\ \text{s.t.} \quad & 3x_2 \geq 6 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \checkmark \text{ (c) min} \quad & 10x_1 + 3x_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 2 \\ & -x_2 \geq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(d) min} \quad & x_1 - 5x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 4 \\ & x_1 - 5x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \checkmark \text{ (e) min} \quad & -3x_1 + 4x_2 \\ \text{s.t.} \quad & -x_1 + 2x_2 \geq 2 \\ & x_1 - 2x_2 \geq 5 \\ & x_1, x_1 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(f) max} \quad & -20x_1 + 15x_2 \\ \text{s.t.} \quad & 10x_1 - 2x_2 \leq 12 \\ & -10x_1 + 2x_2 \leq -15 \\ & x_1, x_2 \geq 0 \end{aligned}$$

6-17 For each of the following LPs and solution vectors, demonstrate that the given solution is feasible, and compute the bound it provides on the optimal objective function value of the corresponding dual.

$$\begin{aligned} \checkmark \text{ (a) min} \quad & 30x_1 + 2x_2 \quad \text{and } \mathbf{x} = (2, 5) \\ \text{s.t.} \quad & 4x_1 + x_2 \leq 15 \\ & 5x_1 - x_2 \geq 2 \\ & 15x_1 - 4x_2 = 10 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{(b) max} \quad & 10x_1 - 6x_2 \quad \text{and } \mathbf{x} = (0, 2) \\ \text{s.t.} \quad & 12x_1 + 4x_2 \leq 8 \\ & 3x_1 - x_2 \geq -5 \\ & 2x_1 + 8x_2 = 16 \\ & x_1, x_2 \geq 0 \end{aligned}$$

6-18 For each of the following, verify that the given formulation is the dual of the referenced primal in Exercise 6-17, demonstrate that the given solution is dual feasible, and compute the bound it provides on the corresponding primal optimal solution value.

$$\checkmark \text{ (a) For 6-17(a) and solution } \mathbf{v} = (0, 0, 2)$$

$$\begin{aligned} \text{max} \quad & 15v_1 + 2v_2 + 10v_3 \\ \text{s.t.} \quad & 4v_1 + 5v_2 + 15v_3 \leq 30 \\ & v_1 - v_2 - 4v_3 \leq 2 \\ & v_1 \leq 0, v_2 \geq 0, v_3 \quad \text{URS} \end{aligned}$$

$$\text{(b) For 6-17(b) and solution } \mathbf{v} = (2, 0, 2)$$

$$\begin{aligned} \text{min} \quad & 8v_1 - 5v_2 + 16v_3 \\ \text{s.t.} \quad & 12v_1 + 3v_2 + 2v_3 \geq 10 \\ & 4v_1 - v_2 + 8v_3 \geq -6 \\ & v_1 \geq 0, v_2 \leq 0, v_3 \quad \text{URS} \end{aligned}$$

6-19 Demonstrate for each linear program in Exercise 6-17 that the dual of its dual is the primal.

6-20 Razorback Tailgate (RT) makes tents for football game parking lot cookouts at its two different facilities, with two processes that may be employed in each facility. All facilities and processes produce the same ultimate product, and Razorback wants to build 22 in the next 80 business hours. Unit costs at facility 1 are \$200 and \$350 for the two processes and consume 5 and 20 production hours per unit, respectively. Similarly, at facility 2 unit costs are \$150 and \$450 with 11 and 23 hours required, respectively, for the two processes. RT wants to meet demand within the 80 available hours at each facility at minimum total cost.

(a) Briefly justify how RT's problem can be formulated as the following LP:

$$\begin{aligned} \text{min} \quad & 200x_1 + 350x_2 + 150x_3 + 450x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \geq 22 \\ & 5x_1 + 20x_2 \leq 80 \\ & 11x_3 + 23x_4 \leq 80 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

(b) State the dual of the LP in part (a) using variables v_1, v_2, v_3 on the 3 main constraints.

(c) Demonstrate that primal solution $\mathbf{x}^* = (14.73, 0.0, 7.27, 0.0)$ and $\mathbf{v}^* = (200.0, 0.0, -4.55)$ are optimal by showing

they are both feasible and have the same solution value.

- (d) Using only parts (a)–(c), what is the marginal cost of production at optimality?
- (e) Using only parts (a)–(c), what would be the marginal savings from another hour of production time at facility 1?
- (f) Using only parts (a)–(c), and assuming it had some impact, would a decrease in the constraint coefficient 11 of this LP increase or decrease the optimal solution value, and would the rate of change lessen or steepen with the magnitude of the decrease?
- (g) Using only parts (a)–(c), and assuming it had some impact, would an increase in the objective function coefficient 150 of this LP increase or decrease the optimal solution value, and would the rate of change steepen or lessen with the magnitude of the increase?
- (h) Facility 2, process 2 (i.e., x_4) is not used in the current optimum. Using only parts (a)–(c), at what lowered unit cost would it begin to be attractive to enter the solution?

6-21 As spring approaches, the campus grounds staff is preparing to buy 500 truckloads of new soil to add around buildings and in gulleys where winter has worn away the surface. Three sources are available at \$220, \$270, and \$290 per truckload, respectively, but the soils vary in nitrogen and clay composition. Soil from source 1 is 50% nitrogen and 40% clay; soil from source 2 is 65% nitrogen and 30% clay; and soil from source 3 is 80% nitrogen and 10% clay. The staff wants the total mix of soil to contain at least 350 truckloads of nitrogen, and at most 75 truckloads of clay. Within the constraints on their needs, they wish to find a minimum cost plan to purchase the soil.

- (a) Define decision variables and annotate objective and constraints to justify why this problem can be modeled as the linear program

$$\begin{aligned} \min \quad & 220t_1 + 270t_2 + 290t_3 \\ \text{s.t.} \quad & t_1 + t_2 + t_3 \geq 500 \\ & .50t_1 + .65t_2 + .80t_3 \geq 350 \\ & .40t_1 + .30t_2 + .10t_3 \leq 75 \\ & t_1, t_2, t_3 \geq 0 \end{aligned}$$

NOTE: This LP has optimal solution $\mathbf{t}^* = (83.33, 0.00, 416.67)$, and the optimal shadow price on the first constraint is $v_1^* = 313.333$

- (b) Using only material given, compute the optimal solution value of the dual corresponding to the LP of part (a), and justify your computation.
- (c) Using only material given, would a unit increase in coefficient/parameter 500 of the LP of part (a) increase or decrease the optimal solution value, and by how much? Also would the rate of change lessen or steepen with the magnitude of the increase? Explain.
- (d) Assuming it has some impact, would a decrease in coefficient/parameter 270 of the LP of part (a) increase or decrease the optimal solution value, and would the rate of change lessen or steepen with the magnitude of the decrease? Explain.
- (e) Assuming it has some impact, would an increase in coefficient/parameter 0.65 of the LP of part (a) increase or decrease the optimal solution value, and would the rate of change lessen or steepen with the magnitude of the increase? Explain.

6-22 Return to Exercise 6-1. Answer each of the following as well as possible from the results in Table 6.6.

- ✓ (a) Is the optimal solution sensitive to the exact value of the trimming hours available? At what number of hours capacity would it become relevant?
- ✓ (b) How much should Super Slayer be willing to pay for an additional hour of extrusion time? For an additional hour of assembly time?
- ✓ (c) What would be the profit effect of increasing assembly capacity to 580 hours? To 680 hours?
- ✓ (d) What would be the profit effect of increasing the profit margin on beta zappers by \$1500 per thousand? What would be the effect of a decrease in that amount?
- ✓ (e) Suppose that the model of Exercise 6-1 ignores packaging capacity because it is hard to estimate, even though each thousand beta zappers requires 2 hours of packaging, and each thousand freeze

phasers requires 3 hours. At what capacity would packaging affect the current optimal solution?

- ✓ (f) Suppose that Super Slayer also has a Ninja Nailer model it could manufacture that requires 2 hours of extrusion, 4 hours of trimming, and 3 hours of assembly per thousand units? At what profit per thousand would it be economic to produce?

6-23 Return to Eli Orchid Exercise 6-2. Answer each of the following as well as possible from the results in Table 6.7.

- (a) What is the marginal cost of production (per ton of output)?
- (b) How much would it cost to produce 70 tons of the new pharmaceutical product? To produce 100 tons?
- (c) How much should Orchid be willing to pay to obtain 20 more tons of ingredient 1? How about ingredient 2?
- (d) How cheap would the third process have to become before it might be used in an optimal solution?
- (e) How much would the cost of the 50 tons of product increase if process 2 actually cost \$32,000 per batch? If it cost \$39,000?
- (f) How much would the cost of the 50 tons of product decrease if process 1 cost \$13,000 per batch? If it cost \$10,000 per batch?
- (g) Suppose that the engineering department is thinking about a new process that produces 6 tons of product using 3 tons of each of the two original ingredients. At what cost would this new process be economic to use?
- (h) Suppose that the three processes actually use 0.1, 0.3, and 0.2 ton per batch of a third ingredient but we do not know exactly how much of it is available. Determine the minimum amount needed if the optimal primal solution in Table 6.7 is not to change.

6-24 Return to Exercise 6-3. Answer each of the following as well as possible from the results in Table 6.8.

- ✓ (a) What is the marginal cost per professional-equivalent hour of programming

associated with the optimal solution in Table 6.8?

- ✓ (b) How much would cost increase if 1050 professional-equivalent hours of programming are required? How about 1100?
- ✓ (c) Does Professor Proof's availability limit the optimal solution? How much would cost change if Professor Proof could devote only 150 hours to supervision? How about 100 hours?
- ✓ (d) How much would the hourly rate of graduate student programmers have to be reduced before Professor Proof might optimally hire some?
- ✓ (e) How much would project cost increase if professional programmers cost \$30 per hour? If they cost \$35?
- ✓ (f) Suppose that Professor Proof decides to require at least half of the total programmer hours to go to students. Could this requirement change the optimal solution?
- ✓ (g) Suppose that Professor Proof decides to allow unlimited hours of graduate student programming. Could this revision change the optimal solution?
- ✓ (h) One of Professor Proof's colleagues has expressed interest in doing some of the programming to earn outside income. At what price per hour should Proof be interested if he estimates that the colleague would be 80% as efficient as a professional and require 0.10 hour of supervision per hour of work?

6-25 Return to NCAA ticket Exercise 6-4. Answer each of the following as well as possible from the results in Table 6.9.

- (a) What is the marginal cost to the NCAA of each seat guaranteed the media?
- (b) Suppose that there is an alternative arrangement of the dome where the games will be played that can provide 15,000 seats. How much additional revenue would be gained from the expanded seating? How much would it be for 20,000 seats?
- (c) Since television revenue provides most of the income for NCAA events, another proposal would reduce the price of general public tickets to \$50. How much revenue would be lost from this change? What if the price were \$30?

- (d) Media-hating coach Sobby Day wants the NCAA to restrict media seats to 20% of those allocated for universities. Could this policy change the optimal solution? How about 10%?
- (e) To accommodate high demand from student supporters of participating universities, the NCAA is considering marketing a new “scrunch seat” that consumes only 80% of a regular bleacher seat but counts fully against the “university \geq half public” rule. Could an optimal solution allocate any such seats at a ticket price of \$35? At a price of \$25?

6-26 Paper can be made from new wood pulp, from recycled office paper, or from recycled newsprint. New pulp costs \$100 per ton, recycled office paper, \$50 per ton, and recycled newsprint, \$20 per ton. One available process uses 3 tons of pulp to make 1 ton of paper; a second uses 1 ton of pulp and 4 tons of recycled office paper; a third uses 1 ton of pulp and 12 tons of recycled newsprint; a fourth uses 8 tons of recycled office paper. At the moment only 80 tons of pulp is available. We wish to produce 100 tons of new paper at minimum total cost.

- (a) Explain why this problem can be modeled as the LP

$$\begin{aligned} \min \quad & 100x_1 + 50x_2 + 20x_3 \\ \text{s.t.} \quad & x_1 = 3y_1 + y_2 + y_3 \\ & x_2 = 4y_2 + 8y_4 \\ & x_3 = 12y_3 \\ & x_1 \leq 80 \\ & \sum_{j=1}^4 y_j \geq 100 \\ & x_1, \dots, x_3, y_1, \dots, y_4 \geq 0 \end{aligned}$$

- (b) State the dual of the given primal LP.
- (c) Enter and solve the given LP with the class optimization software.
- (d) Use your computer output to determine a corresponding optimal dual solution.
- (e) Verify that your computer dual solution is feasible in the stated dual and that it has the same optimal solution value as the primal.
- (f) Use your computer output to determine the marginal cost of paper production at optimality.

- (g) Use your computer output to determine how much we should be willing to pay to obtain an additional ton of pulp.
- (h) Use your computer output to determine or bound as well as possible how much optimal cost would change if the price of pulp increased to \$150 per ton.
- (i) Use your computer output to determine or bound as well as possible how much optimal cost would change if the price of recycled office paper decreased to \$20 per ton.
- (j) Use your computer output to determine or bound as well as possible how much optimal cost would change if the price of recycled office paper increased to \$75 per ton.
- (k) Use your computer output to determine or bound as well as possible how much optimal cost would change if the number of tons of new paper needed decreased to 60.
- (l) Use your computer output to determine or bound as well as possible how much optimal cost would change if the number of tons of new paper needed increased to 200.
- (m) Use your computer output to determine how cheap recycled newsprint would have to become before the primal solution could change.
- (n) An experimental new process will use 6 tons of newsprint and an undetermined number α tons of office paper. Use your computer output to determine how low α would have to be for the new process to be competitive with existing ones.
- (o) Use your computer output to determine whether a limit of 400 tons on recycled office paper would change the primal optimal solution.
- (p) Use your computer output to determine whether a limit of 400 tons on recycled newsprint would change the primal optimal solution.

6-27 Silva and Sons Ltd. (SSL)² is the largest coconut processor in Sri Lanka. SSL buys coconuts at 300 rupees per thousand to produce two grades (fancy and granule) of desiccated (dehydrated) coconut for candy manufacture, coconut

²R. A. Cabraal (1981), “Production Planning in a Sri Lanka Coconut Mill Using Parametric Linear Programming,” *Interfaces*, 11:3, 16–23.

shell flour used as a plastics filler, and charcoal. Nuts are first sorted into those good enough for desiccated coconut (90%) versus those good only for their shells. Those dedicated to desiccated coconut production go to hatcheting/pairing to remove the meat and then through a drying process. Their shells pass on for use in flour and charcoal. The 10% of nuts not suitable for desiccated coconut go directly to flour and charcoal. SSL has the capability to hatchet 300,000 nuts per month and dry 450 tons of desiccated coconut per month. Every 1000 nuts suitable for processing in this way yields 0.16 ton of desiccated coconut, 18% of which is fancy grade and the rest granulated. Shell flour is ground from coconut shells; 1000 shells yield 0.22 ton of flour. Charcoal also comes from shells; 1000 shells yield 0.50 ton of charcoal. SSL can sell fancy desiccated coconut at 3500 rupees per ton over hatcheting and drying cost, but the market is limited to 40 tons per month. A contract requires SSL delivery of at least 30 tons of granulated-quality desiccated coconut per month at 1350 rupees per ton over hatcheting and drying, but any larger amounts can be sold at that price. The market for shell flour is limited to 50 tons per month at 450 rupees each. Unlimited amounts of charcoal can be sold at 250 rupees per ton.

(a) Explain how this coconut production planning problem can be modeled as the LP

$$\begin{aligned}
 \max \quad & 3500s_1 + 1350s_2 + 450s_3 \\
 & + 250s_4 - 300p_1 - 300p_2 \\
 \text{s.t.} \quad & 0.10p_1 - 0.90p_2 = 0 \\
 & 0.82s_1 - 0.18s_2 = 0 \\
 & p_1 \leq 300 \\
 & s_1 + s_2 \leq 450 \\
 & s_1 \leq 40 \\
 & s_2 \geq 30 \\
 & s_3 \leq 50 \\
 & 0.16p_1 - s_1 - s_2 = 0 \\
 & 0.11p_1 + 0.11p_2 - 0.50s_3 - 0.22s_4 = 0 \\
 & p_1, p_2, s_1, s_2, s_3, s_4 \geq 0
 \end{aligned}$$

(b) State the dual of your primal linear program.

(c) Enter and solve the given primal LP with the class optimization software.

- (d) Use your computer output to determine a corresponding optimal dual solution.
- (e) Verify that your computer dual solution is feasible in the stated dual and that it has the same optimal solution value as the primal.
- (f) On the basis of your computer output, determine how much SSL should be willing to pay to increase hatcheting capacity by 1 unit (1000 nuts per month).
- (g) On the basis of your computer output, determine how much SSL should be willing to pay to increase drying capacity by 1 unit (1 ton per week).
- (h) On the basis of your computer output, determine or bound as well as possible the profit impact of a decrease in hatcheting capacity (thousands of nuts per month) to 250. Do the same for a capacity of 200.
- (i) On the basis of your computer output, determine or bound as well as possible the profit impact of an increase in hatcheting capacity (thousands of nuts per month) to 1000. Do the same for a capacity of 2000.
- (j) The company now has excess drying capacity. On the basis of your computer output, determine how low it could go before the optimal plan was affected.
- (k) The optimum now makes no shell flour. On the basis of your computer output, determine at what selling price per ton it would begin to be economical to make and sell flour.
- (l) On the basis of your computer output, determine or bound as well as possible the profit impact of a decrease in the selling price of granulated desiccated coconut to 800 rupees per ton. Do the same for a decrease to 600 rupees.
- (m) On the basis of your computer output, determine or bound as well as possible the profit impact of an increase to 400 rupees per ton in the price of charcoal. Do the same for an increase to 600 rupees.
- (n) On the basis of your computer output, determine whether the primal optimal solution would change if we dropped the constraint on drying capacity.
- (o) On the basis of your computer output, determine whether the primal optimal

solution would change if we added a new limitation that the total number of nuts available per month cannot exceed 400,000. Do the same for a total not to exceed 200,000.

6-28 Tube Steel Incorporated (TSI) is optimizing production at its 4 hot mills. TSI makes 8 types of tubular products which are either solid or hollow and come in 4 diameters. The following two tables show production costs (in dollars) per tube of each product at each mill and the extrusion times (in minutes) for each allowed combination. Missing values indicate product–mill combinations that are not feasible.

Product	Unit Cost			
	Mill 1	Mill 2	Mill 3	Mill 4
0.5 in. solid	0.10	0.10	—	0.15
1 in. solid	0.15	0.18	—	0.20
2 in. solid	0.25	0.15	—	0.30
4 in. solid	0.55	0.50	—	—
0.5 in. hollow	—	0.20	0.13	0.25
1 in. hollow	—	0.30	0.18	0.35
2 in. hollow	—	0.50	0.28	0.55
4 in. hollow	—	1.0	0.60	—

Product	Unit Time			
	Mill 1	Mill 2	Mill 3	Mill 4
0.5 in. solid	0.50	0.50	—	0.10
1 in. solid	0.60	0.60	—	0.30
2 in. solid	0.80	0.60	—	0.60
4 in. solid	0.10	1.0	—	—
0.5 in. hollow	—	1.0	0.50	0.50
1 in. hollow	—	1.2	0.60	0.60
2 in. hollow	—	1.6	0.80	0.80
4 in. hollow	—	2.0	1.0	—

Yearly minimum requirements for the solid sizes (in thousands) are 250, 150, 150, and 80, respectively. For the hollow sizes they are 190, 190, 160, and 150. The mills can operate up to three 40-hour shifts per week, 50 weeks a year. Present policy is that each mill must operate at least one shift.

- (a) Formulate a linear program to meet demand and shift requirements at minimum total cost using the decision variables

$x_{p,m} \triangleq$ thousands of units of product p produced annually at mill m

Main constraints should have a system of 4 minimum time constraints, followed by a system of 4 maximum time constraints, followed by a system of 8 demand constraints.

- (b) State the dual of your primal LP model.
- (c) Enter and solve your primal linear program with the class optimization software.
- (d) Use your computer output to determine a corresponding optimal dual solution.
- (e) Verify that your computer dual solution is feasible in the stated dual and that it has the same optimal solution value as the primal.
- (f) Use your computer results to determine the marginal cost of producing each of the eight products.
- (g) Use your computer results to explain why the policy of operating all mills at least one shift is costing the company money.
- (h) Two options being considered would open mills 3 or 4 on weekends (i.e., add up to 16 extra hours to each of 3 shifts over 50 weeks). Taking each option separately, determine or bound as well as possible from your computer results the impact these changes would have on total production cost.
- (i) Another option being considered is to hire young industrial engineers to find ways of reducing the unit costs of production at high-cost mill 4. For each of the 6 products there taken separately, use your computer results to determine to what level unit costs would have to be reduced before there could be any change in the optimal production plan.
- (j) A final pair of options being considered is to install equipment to produce 4-inch solid and 4-inch hollow tubes at mill 4. The new equipment would produce either product in 1 minute per unit. Taking each product separately, determine the unit production cost that would have to be achieved to make it economical to use the new facilities.

6-29 Consider the primal linear program

$$\begin{aligned} \max \quad & 13z_2 - 8z_3 \\ \text{s.t.} \quad & -3z_1 + z_3 \leq 19 \\ & 4z_1 + 2z_2 + 7z_3 = 10 \\ & 6z_1 + 8z_3 \geq 0 \\ & z_1, z_3 \geq 0 \end{aligned}$$

- ✓ (a) Formulate the corresponding dual in terms of variables v_1, v_2, v_3 .
- ✓ (b) Formulate and justify all Karush-Kuhn-Tucker conditions for primal solution $\bar{\mathbf{z}}$ and dual solution $\bar{\mathbf{v}}$ to be optimal in their respective problems.

6-30 Return to the primal LPs of Exercise 6-17 and corresponding duals of Exercise 6-18.

- (a) State and justify all Karush-Kuhn-Tucker conditions for each pair of models.
- (b) Compute solution values for primal \mathbf{x} in Exercise 6-17 and dual \mathbf{v} in Exercise 6-18, and show that their difference is exactly the value of the complementary slackness violation in the KKT conditions.

6-31 Consider the standard-form linear program

$$\begin{aligned} \max \quad & 5x_1 - 10x_2 \\ \text{s.t.} \quad & 1x_1 - 1x_2 + 2x_3 + 4x_5 = 2 \\ & 1x_1 + 1x_2 + 2x_4 + x_5 = 8 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

- ✓ (a) Taking x_1 and x_2 as basic, identify all elements of the corresponding partitioned model: \mathbf{B} , \mathbf{B}^{-1} , \mathbf{N} , \mathbf{c}^B , \mathbf{c}^N , and \mathbf{b} .
- ✓ (b) Then use the partitioned elements of part (a) to compute the primal solution $(\bar{\mathbf{x}}^B, \bar{\mathbf{x}}^N)$, determine its objective value, and establish that it is feasible.
- ✓ (c) Formulate the dual of the above LP in terms of partitioned elements of part (a) and dual variables \mathbf{v} .
- ✓ (d) Compute the complementary dual solution $\bar{\mathbf{v}}$ corresponding to the basis of (a), and verify that its objective value matches that of the primal.
- ✓ (e) Briefly explain how your complementary primal and dual solutions $\bar{\mathbf{x}}$ and $\bar{\mathbf{v}}$ can have the same objective function value yet not be optimal in their respective problems.

6-32 Return to the standard-form LP of Exercise 6-31, and do parts (a)–(d), this time using basis (x_3, x_4) . Then

- (e) State all KKT conditions for your primal solution of part (b) and dual solution of part (d) to be optimal in their respective problems.
- (f) Demonstrate your solutions of parts (b) and (d) are optimal by showing they satisfy KKT conditions of part (e).

6-33 Consider the linear program

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & -2x_1 + 3x_2 \geq 6 \\ & 3x_1 + 2x_2 \geq 12 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- (a) Establish that subtracting nonnegative surplus variables x_3 and x_4 leads to the equivalent standard-form:

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & -2x_1 + 3x_2 - x_3 = 6 \\ & 3x_1 + 2x_2 - x_4 = 12 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

- (b) Solve the original LP graphically in an (x_1, x_2) plot, and identify an optimal solution. Also tag each main constraint with the corresponding surplus variable.
- (c) Establish that the dual of the standard form in part (a) is

$$\begin{aligned} \max \quad & 6v_1 + 12v_2 \\ \text{s.t.} \quad & -2v_1 + 3v_2 \leq 2 \\ & 3v_1 + 2v_2 \leq 3 \\ & -v_1 \leq 0 \\ & -v_2 \leq 0 \end{aligned}$$

- (d) State all applicable complementary slackness conditions between the standard-form of part (a) and the dual of part (c).

6-34 Consider the linear program

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 + 4x_3 \\ \text{s.t.} \quad & x_1 + 2x_2 + x_3 \geq 3 \\ & 2x_1 - x_2 + 3x_3 \geq 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- ✓ (a) Use nonnegative surplus variables x_4 and x_5 to place the model in standard form.
- ✓ (b) State the dual of your standard form model in part (a) in terms of variables v_1 and v_2 .

- ✔ (c) Choosing x_4 and x_5 as basic, compute the corresponding primal basic solution and establish that it is not feasible.
- ✔ (d) Show that $v_1 = v_2 = 0$ is dual feasible in the standard form and complementary with the primal solution of part (c).
- ✔ (e) Starting from the primal basis of part (c) and dual solution of part (d), apply Dual Simplex Algorithm 6A to compute optimal primal and dual solutions for the given LP.

6-35 Consider the standard form linear program

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 + 6x_3 + 7x_4 + x_5 \\ \text{s.t.} \quad & 2x_1 - x_2 + x_3 + 6x_4 - 5x_5 - x_6 = 6 \\ & x_1 + x_2 + 2x_3 + x_4 + 2x_5 - x_7 = 3 \\ & x_1, \dots, x_7 \geq 0 \end{aligned}$$

- ✔ (a) State the dual of this model using variables v_1 and v_2 .
- ✔ (b) Establish that $v_1 = v_2 = 0$ is dual feasible in your formulation of part (a).
- ✔ (c) Starting from the dual solution of part (b), compute optimal primal and dual solutions to the given LP by Primal-Dual Simplex Algorithm 6B.

6-36 Return to the standard-form LP of Exercise 6-33(a).

- (a) Solve the model by Dual Simplex Algorithm 6A starting from the all surplus basis (x_3, x_4) . At each step, identify the basis matrix \mathbf{B} , its inverse \mathbf{B}^{-1} , the corresponding primal solution \mathbf{x} , the basic cost vector \mathbf{c}^B , the corresponding dual solution \mathbf{v} , reduced costs on all primal variables, direction of change $\Delta \mathbf{v}$, and step size λ . Also verify (i) each \mathbf{v} is dual feasible in your dual of 6-33(c), (ii) each \mathbf{v} and \mathbf{x} together satisfy complementary slackness 6-33(d), and (iii) each is improving for the dual.
- (b) Track your progress with Algorithm 6A on an (x_1, x_2) plot of the original LP, and comment.

6-37 Return to the standard-form LP of Exercise 6-34(a).

- (a) Solve the model by Primal-Dual Simplex Algorithm 6B starting from dual solution $\mathbf{v} = (0, 0)$. At each major step, state the restricted primal, the dual solution \mathbf{v} , reduced costs on all primal variables, and the direction of change $\Delta \mathbf{v}$. Also verify that each \mathbf{v} is complementary with the latest \mathbf{x} , and that each $\Delta \mathbf{v}$ is improving in the dual.
- (b) Track your progress with Algorithm 6B on an (x_1, x_2) plot of the original LP, and comment.

6-38 Consider the following linear program:

$$\begin{aligned} \max \quad & 3z_1 + z_2 \\ \text{s.t.} \quad & -2z_1 + z_2 \leq 2 \\ & z_1 + z_2 \leq 6 \\ & z_1 \leq 4 \\ & z_1, z_2 \geq 0 \end{aligned}$$

After converting to standard form, solution of the model via Rudimentary Simplex Algorithm 5A produces the following sequence of steps:

	z_1	z_2	z_3	z_4	z_5	
$\max \mathbf{c}$	3	1	0	0	0	\mathbf{b}
\mathbf{A}	-2	1	1	0	0	2
	1	1	0	1	0	6
	1	0	0	0	1	4
$t = 0$	N	N	B	B	B	
$\mathbf{z}^{(0)}$	0	0	2	6	4	0
$\Delta \mathbf{z}, z_1$	1	0	2	-1	-1	$\bar{c}_1 = 3$
$\Delta \mathbf{z}, z_2$	0	1	-1	-1	0	$\bar{c}_2 = 1$
	-	-	-	$\frac{6}{1}$	$\frac{4}{1}$	$\lambda = 4$
$t = 1$	B	N	B	B	N	
$\mathbf{z}^{(1)}$	4	0	10	2	0	12
$\Delta \mathbf{z}, z_2$	0	1	-1	-1	0	$\bar{c}_2 = 1$
$\Delta \mathbf{z}, z_5$	-1	0	-2	1	1	$\bar{c}_5 = -3$
	-	-	$\frac{10}{1}$	$\frac{2}{1}$	-	$\lambda = 2$
$t = 2$	B	B	B	N	N	
$\mathbf{z}^{(2)}$	4	2	8	0	0	14
$\Delta \mathbf{z}, z_4$	0	-1	1	1	0	$\bar{c}_4 = -1$
$\Delta \mathbf{z}, z_5$	-1	1	-3	0	1	$\bar{c}_5 = -2$

- (a) State the dual of the standard-form primal depicted at the top of this table, and enumerate all complementary slackness requirements between primal and dual.
- (b) Compute the primal and dual basic solutions at each step of the given simplex computations, and check complementary slackness, to demonstrate that Algorithm 5A is following of the KKT strategy of maintaining primal feasibility and complementary slackness while seeking dual feasibility.

REFERENCES

- Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.
- Bertsimas, Dimitris and John N. Tsitsiklis (1997), *Introduction to Linear Optimization*. Athena Scientific, Nashua, New Hampshire.
- Chvátal, Vašek (1980), *Linear Programming*, W.H. Freeman, San Francisco, California.
- Eppen, G.D., F.J. Gould, G.P. Schmidt, Jeffrey H. Moore, and Larry R. Weatherford (1993), *Introduction to Management Science*, Prentice-Hall, Upper Saddle River, New Jersey.
- Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009), *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.
- Luenberger, David G. and Yinyu Ye (2008), *Linear and Nonlinear Programming*, Springer, New York, New York.

Interior Point Methods for Linear Programming

Although Chapter 5's and 6's simplex methods remain the most widely used algorithms for solving linear programs, a very different strategy emerged in operations research practice at the end of the 1980s. **Interior point methods** still follow the improving search paradigm for linear programs, but they employ moves quite different from those in the simplex method. Instead of staying on the boundary of the feasible region and passing from extreme point to extreme point, interior point methods proceed directly across the interior.

Much more effort turns out to be required per move with interior point methods, but the number of moves decreases dramatically. In many large LPs, the result is a substantially shorter total solution time than any obtained so far with simplex.

The first commercial interior point method for linear programming was N. Karmarkar's **projective transformation** procedure and developments have continued to this day. All methods are relatively complex mathematically, with many details beyond the scope of an introductory book.

In this chapter we seek to provide an overview of popular **affine scaling**, **log-barrier**, and **primal-dual** variants, then comment briefly on the theoretical importance of interior-point approaches with respect to polynomial-time solution of linear programs (see Sections 14.2–14.3). The development assumes reader familiarity with search fundamentals of Chapter 3, LP conventions of Section 5.1, and LP optimality conditions of Section 6.7.

7.1 SEARCHING THROUGH THE INTERIOR

Any linear programming search method allowed to pass through the interior of the feasible region poses a variety of new challenges. Before developing specific algorithms, we introduce some of the key issues.

APPLICATION 7.1: FRANNIE’S FIREWOOD

To illustrate interior point computations, we need an example so small (and trivial) that it can be displayed graphically in a variety of ways. Frannie’s Firewood problem will serve.

Each year Frannie sells up to 3 cords of firewood from her small woods. One potential customer has offered her \$90 per half-cord and another \$150 per full cord. Our concern is how much Frannie should sell to each customer to maximize income, assuming that each will buy as much as he or she can.

To begin, we need a decision variable for each customer. Define

$$x_1 \triangleq \text{number of half-cords sold to customer 1}$$

$$x_2 \triangleq \text{number of cords sold to customer 2}$$

Then Frannie’s problem can be modeled as the linear program

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 \\ \text{s.t.} \quad & \frac{1}{2}x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{7.1}$$

Figure 7.1 solves model (7.1) graphically. The unique solution sells all 3 cords to the first customer, making $x_1^* = 6, x_2^* = 0$.

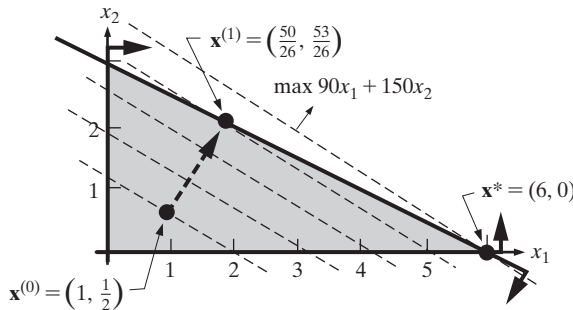


FIGURE 7.1 Graphic Solution of Frannie’s Firewood Application

Interior Points

Recall (definition [5.2](#), Section 5.1) that a feasible solution to a linear program is a **boundary point** if at least one inequality constraint of the model that can be strict for some feasible solutions is satisfied as an equality at the given point. A feasible solution is an **interior point** if no such inequalities are active. For example, solution $\mathbf{x}^{(0)} = (1, \frac{1}{2})$ of Figure 7.1 is an interior point because it satisfies all three constraints of model (7.1) as strict inequalities.

Objective as a Move Direction

There is one enormous convenience in searching from an interior point such as $\mathbf{x}^{(0)}$ of Figure 7.1 when all constraints are inequalities (we deal with equalities shortly).

With no constraints active, every direction is **feasible**; that is, a small step in any direction retains feasibility (see Section 3.2). Thus our only consideration in picking the next move is to find a direction **improving** the objective function.

What direction improves the objective most rapidly? For linear programs it is easy to see that we should move with the gradient or objective coefficient vector as in principle [3.23](#) (Section 3.3).

Principle 7.1 The move direction of most rapid improvement for linear objective $\max \mathbf{c} \cdot \mathbf{x} = \sum_j c_j x_j$ is the objective function vector $\Delta \mathbf{x} = \mathbf{c}$. For a model to $\min \mathbf{c} \cdot \mathbf{x}$, it is $\Delta \mathbf{x} = -\mathbf{c}$.

For example, in the maximize Frannie model of Figure 7.1, we prefer $\Delta \mathbf{x} = \mathbf{c} = (90, 150)$ at $\mathbf{x}^{(0)}$. This direction runs exactly perpendicular to contours of the objective function. No alternative improves the objective faster.

EXAMPLE 7.1: USING THE OBJECTIVE AS A DIRECTION

Each of the following is the objective function of a linear program with 3 decision variables. Determine for each the direction of most rapid objective improvement.

(a) $\min 4x_1 - 19x_2 + x_3$

(b) $\max -2x_1 - x_2 + 79x_3$

Solution: We apply principle [7.1](#).

(a) For a minimize objective the direction of steepest improvement is

$$\Delta \mathbf{x} = -\mathbf{c} = (-4, 19, -1)$$

(b) For a maximize objective the direction of steepest improvement is

$$\Delta \mathbf{x} = \mathbf{c} = (-2, -1, 79)$$

Boundary Strategy of Interior Point Methods

Figure 7.1 shows the effect of using cost direction $\Delta \mathbf{x} = (90, 150)$ at $\mathbf{x}^{(0)}$. A maximum feasible step of $\lambda = \frac{2}{195}$ would bring us to

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x}^{(1)} \\ &= \left(1, \frac{1}{2}\right) + \frac{2}{195} (90, 150) \\ &= \left(\frac{50}{26}, \frac{53}{26}\right) \end{aligned}$$

Very rapid progress toward an optimum is achieved in a single move.

Notice that we would have to deal with the boundary at $\mathbf{x}^{(1)}$. Constraint

$$\frac{1}{2}x_1 + x_2 \leq 3$$

is now active, and we discovered in Chapter 3 (principle [3.25](#), Section 3.3) that feasible directions $\Delta \mathbf{x}$ at $\mathbf{x}^{(1)}$ must preserve this inequality by satisfying

$$\frac{1}{2} \Delta x_1 + \Delta x_2 \leq 0$$

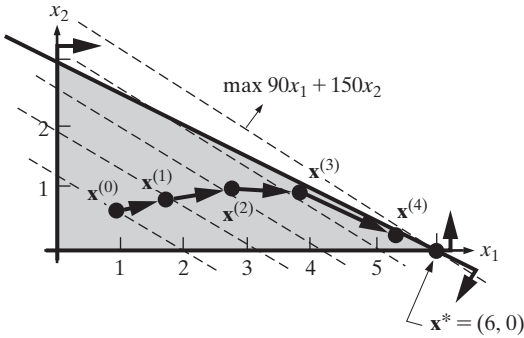


FIGURE 7.2 Typical Interior Point Search of Frannie's Firewood Application

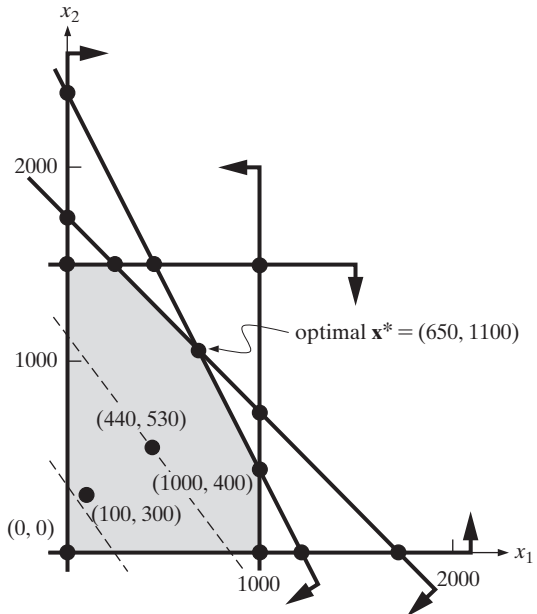
Such newly active constraints destroy the convenience of moves in the interior. That is why interior point algorithms stop short of the boundary as in the typical sequence plotted in Figure 7.2. Partial steps along suitable variations of the cost direction in principle [7.1] continue progress while avoiding the boundary.

Of course, optimal solutions to linear programs lie along the boundary of the feasible region (principle [5.4], Section 5.1). It cannot be avoided forever. The effectiveness of interior point methods depends on their keeping to the “middle” of the feasible region until an optimal solution is reached.

Principle 7.2 Interior point algorithms begin at and move through a sequence of interior feasible solutions, converging to the boundary of the feasible region only at an optimal solution.

EXAMPLE 7.2: IDENTIFYING INTERIOR POINT TRAJECTORIES

The following figure shows the Top Brass Trophy application from Section 5.1:



Determine whether each of the following sequences of solutions could have resulted from an interior point search.

- (a) $(0, 0)$, $(440, 530)$, $(650, 1100)$
- (b) $(100, 300)$, $(440, 530)$, $(650, 1100)$
- (c) $(100, 300)$, $(1000, 400)$, $(650, 1100)$
- (d) $(0, 0)$, $(1000, 0)$, $(1000, 400)$, $(650, 1100)$

Solution: The sequence of solutions visited by an interior point search should conform to principle [7.2](#).

- (a) This sequence is not appropriate because it begins at boundary point $(0, 0)$.
- (b) This could be the sequence of an interior point search. It starts at one interior solution, passes to another, and reaches the boundary only at the optimum.
- (c) This sequence starts in the interior but goes to the boundary before optimality at $(1000, 400)$. It could not result from an interior point algorithm.
- (d) This is the simplex algorithm extreme-point sequence of Figure 5.5. It never enters the interior.

Interior in LP Standard Form

In Section 5.1 we showed how every linear program can be placed in **standard form**:

$$\begin{array}{ll} \min \text{ (or max)} & \mathbf{c} \cdot \mathbf{x} \\ \text{s.t.} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \quad (7.2)$$

Any main inequalities are converted to equalities by adding slack variables, and original variables are transformed until each is subject to a nonnegativity constraint. For example, we can place the Frannie model (7.1) in standard form by adding slack variable x_3 in the main constraint to obtain

$$\begin{array}{ll} \max & 90x_1 + 150x_2 \\ \text{s.t.} & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{array} \quad (7.3)$$

Chapter 5 placed LPs in form (7.2) to make it easier to perform simplex algorithm computations, but standard form is equally convenient for interior point search. With all inequalities reduced to nonnegativity constraints, it is easy to check whether a given solution lies in the interior of the feasible region.

Principle 7.3 | A feasible solution for a linear program in standard form is an interior point if every component of the solution that can be positive in any feasible solution is strictly positive in the given point.

For example, point $\mathbf{x}^{(0)}$ of Figure 7.2 corresponds to $\mathbf{x}^{(0)} = (1, \frac{1}{2}, 2)$ in standard form (7.3). In accord with conditions [7.3](#), the vector is positive in every component.

EXAMPLE 7.3: IDENTIFYING STANDARD-FORM INTERIOR POINTS

Consider the standard-form linear program

$$\begin{aligned} \min \quad & 5x_1 - 2x_3 + 8x_4 \\ \text{s.t.} \quad & 2x_1 + 3x_2 - x_3 = 10 \\ & 6x_1 - 2x_4 = 12 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

which does have strictly positive feasible solutions. Determine whether each of the following solutions corresponds to an interior point.

$$(a) \mathbf{x}^{(1)} = (8, 0, 6, 18) \quad (b) \mathbf{x}^{(2)} = (4, 1, 1, 6) \quad (c) \mathbf{x}^{(3)} = (3, 3, 1, 6)$$

Solution:

(a) Solution $\mathbf{x}^{(1)}$ cannot be interior, because a component $x_2^{(1)} = 0$ is not strictly positive.

(b) Solution $\mathbf{x}^{(2)}$ is positive in every component. Also,

$$\begin{aligned} 2x_1^{(2)} + 3x_2^{(2)} - x_3^{(2)} &= 2(4) + 3(1) - (1) = 10 \\ 6x_1^{(2)} - 2x_4^{(2)} &= 6(4) - 2(6) = 12 \end{aligned}$$

which establishes the point is feasible. Thus $\mathbf{x}^{(2)}$ is an interior point.

(c) Solution $\mathbf{x}^{(3)}$ is also positive in every component. However,

$$\begin{aligned} 2x_1^{(3)} + 3x_2^{(3)} - x_3^{(3)} &= 2(3) + 3(3) - (1) = 14 \neq 10 \\ 6x_1^{(3)} - 2x_4^{(3)} &= 6(3) - 2(6) = 6 \neq 12 \end{aligned}$$

Thus the point is infeasible and so not interior.

Projecting to Deal with Equality Constraints

The ease of identifying interior points in standard form comes at a price. Many equality constraints are added to the system $\mathbf{Ax} = \mathbf{b}$ as main inequalities are converted, and equality constraints are active at every solution. Straightforward moves such as the objective vector directions of principle [7.1] must now be modified to preserve the equalities.

Chapter 3 (principle [3.25], Section 3.3) established that a direction $\Delta\mathbf{x}$ preserves equality constraints $\mathbf{Ax} = \mathbf{b}$ if and only if the net effect on every constraint is zero.

Principle 7.4 | A move direction $\Delta\mathbf{x}$ is feasible for equality constraints $\mathbf{Ax} = \mathbf{b}$ if it satisfies $\mathbf{A}\Delta\mathbf{x} = \mathbf{0}$.

How can we find a $\Delta \mathbf{x}$ direction satisfying conditions [7.4] that approximates as nearly as possible the direction \mathbf{d} we would really like to follow? Interior point algorithms often use some form of projection.

Principle 7.5 The **projection** of a move vector \mathbf{d} on a given system of equalities is a direction preserving those constraints and minimizing the total squared difference between its components and those of \mathbf{d} .

Readers familiar with statistics may recognize this idea as the one used in **least squares** curve fitting.

Figure 7.3 illustrates for Frannie's firewood optimization form (7.3). The shaded triangle in this 3-dimensional plot is the set of feasible (x_1, x_2, x_3) . The search begins at $\mathbf{x}^{(0)} = (1, \frac{1}{2}, 2)$.

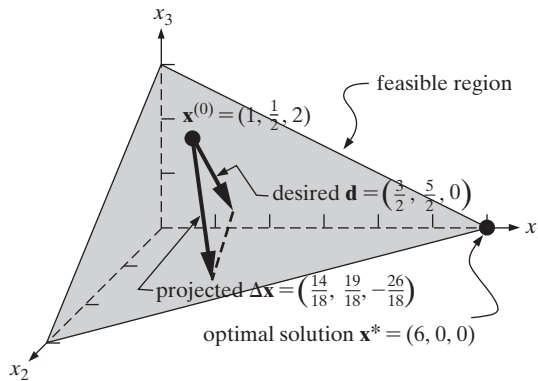


FIGURE 7.3 Projection in the Frannie's Firewood Application

To improve the objective function as quickly as possible, we would like to move parallel to standard-form cost vector $\mathbf{c} = (90, 150, 0)$. Call the desired direction $\mathbf{d} = \frac{1}{60} \mathbf{c} = (\frac{3}{2}, \frac{5}{2}, 0)$ to keep it in the picture.

A feasible direction $\Delta \mathbf{x}$ must keep the next point in the feasible plane by satisfying [7.4], or

$$\frac{1}{2} \Delta x_1 + \Delta x_2 + \Delta x_3 = 0 \quad (7.4)$$

But we would like it to be as much like \mathbf{d} as possible. Figure 7.3 shows that the best choice is \mathbf{d} 's projection $\Delta \mathbf{x} = (\frac{14}{18}, \frac{19}{18}, -\frac{26}{18})$. It is the closest to \mathbf{d} in the sense that

$$(d_1 - \Delta x_1)^2 + (d_2 - \Delta x_2)^2 + (d_3 - \Delta x_3)^2$$

is minimized. It also satisfies feasibility condition (7.4), because

$$\frac{1}{2} \left(\frac{14}{18}\right) + \left(\frac{19}{18}\right) + \left(-\frac{26}{18}\right) = 0$$

Derivation of the projection computation that produced this move direction $\Delta \mathbf{x}$ is beyond the scope of this book. However, the formula for the required projection matrix \mathbf{P} is well known:

Principle 7.6 | The projection of direction \mathbf{d} onto conditions $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$ preserving linear inequalities $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be computed as

$$\Delta \mathbf{x} = \mathbf{P}\mathbf{d}$$

where **projection matrix**

$$\mathbf{P} = (\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})$$

Here \mathbf{I} denotes an identity matrix and \mathbf{A}^T indicates the **transpose** of \mathbf{A} , which is obtained by swapping its rows for its columns. (Primer 6 reviews some properties for those who may require it.)

For our Frannie’s Firewood application,

$$\mathbf{A} = \left(\frac{1}{2}, 1, 1\right), \quad \mathbf{A}^T = \begin{pmatrix} \frac{1}{2} \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{A}\mathbf{A}^T = \frac{9}{4}$$

Thus $(\mathbf{A}\mathbf{A}^T)^{-1} = \frac{4}{9}$, and

$$\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} = \begin{pmatrix} \frac{1}{2} \\ 1 \\ 1 \end{pmatrix} \left(\frac{4}{9}\right) \left(\frac{1}{2}, 1, 1\right) = \begin{pmatrix} \frac{1}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{4}{9} & \frac{4}{9} \\ \frac{2}{9} & \frac{4}{9} & \frac{4}{9} \end{pmatrix}$$

The needed projection matrix \mathbf{P} then becomes

$$\begin{aligned} \mathbf{P} &= (\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{9} & \frac{2}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{4}{9} & \frac{4}{9} \\ \frac{2}{9} & \frac{4}{9} & \frac{4}{9} \end{pmatrix} = \begin{pmatrix} \frac{8}{9} & -\frac{2}{9} & -\frac{2}{9} \\ -\frac{2}{9} & \frac{5}{9} & -\frac{4}{9} \\ -\frac{2}{9} & -\frac{4}{9} & \frac{5}{9} \end{pmatrix} \end{aligned}$$

Applying rule 7.6 for $\mathbf{d} = \left(\frac{3}{2}, \frac{5}{2}, 0\right)$ gives

$$\begin{aligned} \Delta \mathbf{x} &= \mathbf{P}\mathbf{d} \\ &= \begin{pmatrix} \frac{8}{9} & -\frac{2}{9} & -\frac{2}{9} \\ -\frac{2}{9} & \frac{5}{9} & -\frac{4}{9} \\ -\frac{2}{9} & -\frac{4}{9} & \frac{5}{9} \end{pmatrix} \begin{pmatrix} \frac{3}{2} \\ \frac{5}{2} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{14}{18} \\ \frac{19}{18} \\ -\frac{26}{18} \end{pmatrix} \end{aligned} \tag{7.5}$$

EXAMPLE 7.4: PROJECTING TO PRESERVE EQUALITY CONSTRAINTS

Consider the standard-form linear program

$$\begin{aligned} \max \quad & 5x_1 + 7x_2 + 9x_3 \\ \text{s.t.} \quad & 1x_1 \qquad \qquad -1x_3 = -1 \\ & \qquad \qquad +2x_2 + 1x_3 = 5 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- (a) Determine the direction \mathbf{d} of most rapid objective function improvement at interior point $\mathbf{x} = (2, 1, 3)$.
- (b) Project that vector \mathbf{d} to find the nearest direction $\Delta\mathbf{x}$ that preserves the main equality constraints of the LP.
- (c) Verify that your $\Delta\mathbf{x}$ satisfies all requirements for a feasible move direction at \mathbf{x} .

Solution:

- (a) Applying principle [7.1](#), the direction of steepest improvement is

$$\mathbf{d} = \text{objective function vector} = (5, 7, 9)$$

- (b) To compute the projection, we apply formula [7.6](#). Here

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{A}^T = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ -1 & 1 \end{pmatrix}$$

Thus

$$\mathbf{A}\mathbf{A}^T = \begin{pmatrix} 2 & -1 \\ -1 & 5 \end{pmatrix} \quad \text{with inverse} \quad (\mathbf{A}\mathbf{A}^T)^{-1} = \begin{pmatrix} \frac{5}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{2}{9} \end{pmatrix}$$

Continuing yields

$$\begin{aligned} \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} &= \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \frac{5}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{2}{9} \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{5}{9} & \frac{2}{9} & -\frac{4}{9} \\ \frac{2}{9} & \frac{8}{9} & \frac{2}{9} \\ -\frac{4}{9} & \frac{2}{9} & \frac{5}{9} \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} \mathbf{P} &= [\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}] \\ &= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{5}{9} & \frac{2}{9} & -\frac{4}{9} \\ \frac{2}{9} & \frac{8}{9} & \frac{2}{9} \\ -\frac{4}{9} & \frac{2}{9} & \frac{5}{9} \end{pmatrix} \right] = \begin{pmatrix} \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \\ -\frac{2}{9} & \frac{1}{9} & -\frac{2}{9} \\ \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \end{pmatrix} \end{aligned}$$

We conclude

$$\Delta \mathbf{x} = \mathbf{P}\mathbf{d} = \begin{pmatrix} \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \\ -\frac{2}{9} & \frac{1}{9} & -\frac{2}{9} \\ \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \end{pmatrix} \begin{pmatrix} 5 \\ 7 \\ 9 \end{pmatrix} = \begin{pmatrix} \frac{14}{3} \\ -\frac{7}{3} \\ \frac{14}{3} \end{pmatrix}$$

(c) Checking net change zero conditions [7.4], we see that

$$\mathbf{A}\Delta \mathbf{x} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} \frac{14}{3} \\ -\frac{7}{3} \\ \frac{14}{3} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

With no inequalities (nonnegativities) active at \mathbf{x} , this is all that is required for $\Delta \mathbf{x}$ to be feasible.

Improvement with Projected Directions

Projection principle [7.6] yields a feasible direction $\Delta \mathbf{x}$ at any standard-form interior point. But does the direction remain improving? For example, we know that the direction $\mathbf{d} = (\frac{3}{2}, \frac{5}{2}, 0)$ used in Frannie’s Firewood computation (7.5) is improving because it parallels the maximize objective function vector $\mathbf{c} = (90, 150, 0)$. Projection would do little good if it achieved feasibility of $\Delta \mathbf{x}$ at the loss of this improving property.

Fortunately, we can show that improvement is always preserved in projecting objective function vectors.

Principle 7.7 The projection $\Delta \mathbf{x} = \mathbf{P}\mathbf{c}$ of (nonzero) objective function vector \mathbf{c} onto equality constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ is an improving direction at every \mathbf{x} for a max case LP, and its negative improves for a min case.

For example, the Frannie’s Firewood direction $\Delta \mathbf{x}$ of (7.5) satisfies gradient improvement conditions [3.21] and [3.22] of Section 3.3 because

$$\begin{aligned} \mathbf{c} \cdot \Delta \mathbf{x} &= (90, 150, 0) \cdot \left(\frac{14}{8}, \frac{19}{18}, -\frac{26}{18}\right) \\ &= \frac{685}{3} > 0 \end{aligned}$$

To see that this will always be true, we need to make a simple observation about projections. Computation [7.6] finds the nearest vector $\Delta \mathbf{x}$ to direction \mathbf{d} that satisfies $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$. Thus if \mathbf{d} already satisfies $\mathbf{A}\mathbf{d} = \mathbf{0}$, the nearest such direction must be $\Delta \mathbf{x} = \mathbf{d}$ itself. That is, re-projection of a $\Delta \mathbf{x}$ that has already been projected must leave $\Delta \mathbf{x}$ unchanged, or in symbols,

$$\begin{aligned} \text{projection}(\mathbf{d}) &= \mathbf{P}\mathbf{d} \\ &= \mathbf{P}\mathbf{P}\mathbf{d} \\ &= \text{projection}[\text{projection}(\mathbf{d})] \end{aligned} \tag{7.6}$$

Projection matrices are also known to be symmetric ($\mathbf{P} = \mathbf{P}^T$). Combining with property (7.6) applied to maximize objective function vector \mathbf{c} , we can see that the corresponding $\Delta \mathbf{x}$ of principle [7.6] has

$$\begin{aligned} \mathbf{c} \cdot \Delta \mathbf{x} &= \mathbf{c}^T \mathbf{P} \mathbf{c} \\ &= \mathbf{c}^T \mathbf{P} \mathbf{P} \mathbf{c} \\ &= \mathbf{c}^T \mathbf{P}^T \mathbf{P} \mathbf{c} \\ &= (\mathbf{P} \mathbf{c})^T (\mathbf{P} \mathbf{c}) \\ &= \Delta \mathbf{x}^T \Delta \mathbf{x} \\ &> 0 \end{aligned}$$

This is exactly what is required for improvement, and a similar analysis holds for minimize models.

EXAMPLE 7.5: VERIFYING PROJECTED OBJECTIVE IMPROVEMENT

Return to the linear program of Example 7.4 and its projection matrix \mathbf{P} derived in part (b). Assuming that the objective function was changed to each of the following, compute the $\Delta \mathbf{x}$ value obtained by projecting the corresponding steepest improvement direction of principle [7.1], and verify that the resulting move direction is improving.

(a) $\max x_1 - x_2 + x_3$

(b) $\min 2x_1 + x_3$

Solution:

(a) The corresponding steepest improvement direction is $\mathbf{c} = (1, -1, 1)$, so that the projected move direction would be

$$\Delta \mathbf{x} = \mathbf{P} \mathbf{c} = \begin{pmatrix} \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \\ -\frac{2}{9} & \frac{1}{9} & -\frac{2}{9} \\ \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{10}{9} \\ -\frac{5}{9} \\ \frac{10}{9} \end{pmatrix}$$

Checking improvement yields

$$\mathbf{c} \cdot \Delta \mathbf{x} = (1, -1, 1) \cdot \left(\frac{10}{9}, -\frac{5}{9}, \frac{10}{9}\right) = \frac{25}{9} > 0$$

(b) For this minimize objective the steepest improvement direction is $-\mathbf{c} = (-2, 0, -1)$. Projecting gives

$$\Delta \mathbf{x} = \begin{pmatrix} \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \\ -\frac{2}{9} & \frac{1}{9} & -\frac{2}{9} \\ \frac{4}{9} & -\frac{2}{9} & \frac{4}{9} \end{pmatrix} \begin{pmatrix} -2 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} -\frac{4}{3} \\ \frac{2}{3} \\ -\frac{4}{3} \end{pmatrix}$$

Checking improvement yields

$$\mathbf{c} \cdot \Delta \mathbf{x} = (2, 0, 1) \cdot \left(-\frac{4}{3}, \frac{2}{3}, -\frac{4}{3}\right) = -3 < 0$$

7.2 SCALING WITH THE CURRENT SOLUTION

We have already seen that it is important for interior point algorithms to avoid the boundary of the feasible set (until optimality). One of the tools used by virtually all such procedures to keep in the “middle” of the feasible region is **scaling**—revising the units in which decision variables are expressed to place all solution components a comfortable number of units from the boundary. In this section we introduce the most common **affine** type of rescaling used in all algorithms of this chapter.

Affine Scaling

Affine scaling adopts the simplest possible strategy for keeping away from the boundary. The model is rescaled so that the transformed version of current solution $\mathbf{x}^{(t)}$ is equidistant from all inequality constraints.

Figure 7.4 shows the idea for our Frannie’s Firewood application model (7.3). Part (a) depicts the original feasible space, with a current interior point solution $\mathbf{x}^{(t)} = (3, \frac{1}{2}, 1)$. The rescaled version in part (b) converts to new variables

$$\begin{aligned}
 y_1 &\triangleq \frac{x_1}{x_1^{(t)}} = \frac{x_1}{3} \\
 y_2 &\triangleq \frac{x_2}{x_2^{(t)}} = \frac{x_2}{\frac{1}{2}} \\
 y_3 &\triangleq \frac{x_3}{x_3^{(t)}} = \frac{x_3}{1}
 \end{aligned}$$

After dividing by current (positive because interior) x -values in this way, the corresponding current solution $\mathbf{y}^{(t)} = (1, 1, 1)$ —equidistant from all inequality (nonnegativity) constraints.

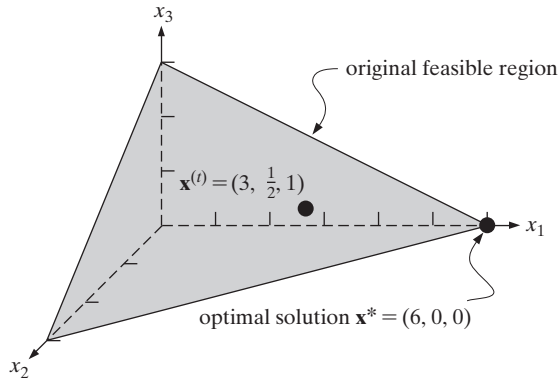
Diagonal Matrix Formalization of Affine Scaling

Although it may seem excessive at this stage, it will prove helpful to formalize affine scaling’s simple notion of dividing components by the current solution in terms of **diagonal matrices**. In particular, we convert current solution vectors

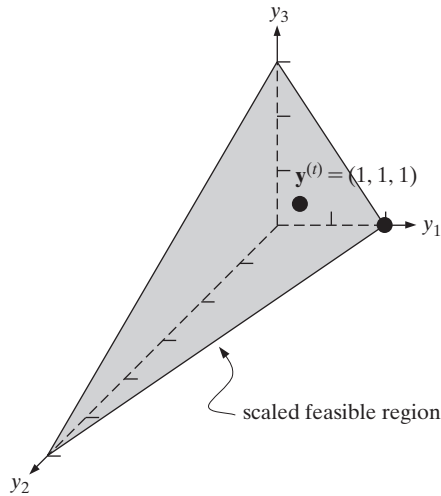
$$\mathbf{x}^{(t)} = (x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)})$$

into the square matrix

$$\mathbf{X}_t = \begin{pmatrix} x_1^{(t)} & 0 & \cdots & 0 \\ 0 & x_2^{(t)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_n^{(t)} \end{pmatrix} \quad \text{with} \quad \mathbf{X}_t^{-1} = \begin{pmatrix} \frac{1}{x_1^{(t)}} & 0 & \cdots & 0 \\ 0 & \frac{1}{x_2^{(t)}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{x_n^{(t)}} \end{pmatrix}$$



(a) Original space



(b) Affine scaled space

FIGURE 7.4 Affine Scaling of Frannie's Firewood Application

Affine scaling and unscaling can then be expressed in terms of multiplication by such matrices.

Definition 7.8 At current solution $\mathbf{x}^{(t)} > \mathbf{0}$, **affine scaling** transforms points \mathbf{x} into \mathbf{y} defined by

$$\mathbf{y} = \mathbf{X}_t^{-1}\mathbf{x} \quad \text{or} \quad y_j = \frac{x_j}{x_j^{(t)}} \quad \text{for all } j$$

where \mathbf{X}_t denotes a square matrix with the components of $\mathbf{x}^{(t)}$ on its diagonal.

With $y_j = x_j/x_j^{(t)}$ the inverse affine scaling is equally easy to express.

Principle 7.9 At current solution $\mathbf{x}^{(t)} > \mathbf{0}$, the point \mathbf{x} corresponding to affine-scaled solution \mathbf{y} is

$$\mathbf{x} = \mathbf{X}_t \mathbf{y} \quad \text{or} \quad x_j = (x_j^{(t)})y_j \quad \text{for all } j$$

where \mathbf{X}_t denotes a square matrix with the components of $\mathbf{x}^{(t)}$ on its diagonal.

For example, in the Frannie’s firewood case of Figure 7.4,

$$\mathbf{x}^{(t)} = \begin{pmatrix} 3 \\ \frac{1}{2} \\ 1 \end{pmatrix} \quad \text{gives} \quad \mathbf{X}_t = \begin{pmatrix} 3 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{with} \quad \mathbf{X}_t^{-1} = \begin{pmatrix} \frac{1}{3} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then under formula [7.8],

$$\mathbf{y} = \mathbf{X}_t^{-1} \mathbf{x}^{(t)} = \begin{pmatrix} \frac{1}{3} & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ \frac{1}{2} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Similarly, applying inverse formula [7.9] retrieves

$$\mathbf{x} = \mathbf{X}_t \mathbf{y} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ \frac{1}{2} \\ 1 \end{pmatrix}$$

EXAMPLE 7.6: AFFINE SCALING WITH DIAGONAL MATRICES

Suppose that the first two solutions visited by a linear programming search algorithm are $\mathbf{x}^{(0)} = (12, 3, 2)$ and $\mathbf{x}^{(1)} = (1, 4, 7)$. Compute the affine scalings of $\mathbf{x} = (24, 12, 14)$ relative to each of these $\mathbf{x}^{(t)}$, and verify that applying reverse scaling to the resulting \mathbf{y} ’s recovers \mathbf{x} .

Solution: For $\mathbf{x}^{(0)}$ the diagonal matrix of computation [7.8] is

$$\mathbf{X}_0 = \begin{pmatrix} 12 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad \text{with} \quad \mathbf{X}_0^{-1} = \begin{pmatrix} \frac{1}{12} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Thus

$$\mathbf{y} = \mathbf{X}_0^{-1} \mathbf{x} = \begin{pmatrix} \frac{1}{12} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 24 \\ 12 \\ 14 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 7 \end{pmatrix}$$

Reversing the scaling with formula [7.9] recovers

$$\mathbf{x} = \mathbf{X}_0 \mathbf{y} = \begin{pmatrix} 12 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \\ 7 \end{pmatrix} = \begin{pmatrix} 24 \\ 12 \\ 14 \end{pmatrix}$$

After the search advances to $\mathbf{x}^{(1)}$, the scaling changes. Now the diagonal matrix of computation [7.8] is

$$\mathbf{X}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 7 \end{pmatrix} \quad \text{with} \quad \mathbf{X}_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{7} \end{pmatrix}$$

Thus solution $\mathbf{x} = (24, 12, 14)$ scales to

$$\mathbf{y} = \mathbf{X}_1^{-1} \mathbf{x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{7} \end{pmatrix} \begin{pmatrix} 24 \\ 12 \\ 14 \end{pmatrix} = \begin{pmatrix} 24 \\ 3 \\ 2 \end{pmatrix}$$

Still, reversing the scaling with formula [7.9] recovers

$$\mathbf{x} = \mathbf{X}_1 \mathbf{y} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 7 \end{pmatrix} \begin{pmatrix} 24 \\ 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 24 \\ 12 \\ 14 \end{pmatrix}$$

Affine-Scaled Standard Form

Affine scaling in effect changes the objective function and constraint coefficients of the LP standard form:

$$\begin{array}{ll} \min \text{ or } \max & \mathbf{c} \mathbf{x} \\ \text{s.t.} & \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

Substituting $\mathbf{x} = \mathbf{X}_t \mathbf{y}$ from formula [7.9] and collecting $\mathbf{c} \cdot \mathbf{x} = \mathbf{c} \mathbf{X}_t \mathbf{y} \triangleq \mathbf{c}^{(t)} \mathbf{y}$ and $\mathbf{A} \mathbf{x} = \mathbf{A} \mathbf{X}_t \mathbf{x} \triangleq \mathbf{A}_t \mathbf{y}$ produces a new affine-scaled standard form for each solution $\mathbf{x}^{(t)}$.

Definition 7.10 At current feasible solution $\mathbf{x}^{(t)} > \mathbf{0}$, the **affine-scaled version of a standard-form linear program** is

$$\begin{array}{ll} \min \text{ or } \max & \mathbf{c}^{(t)} \cdot \mathbf{y} \\ \text{s.t.} & \mathbf{A}_t \mathbf{y} = \mathbf{b} \\ & \mathbf{y} \geq \mathbf{0} \end{array}$$

where $\mathbf{c}^{(t)} \triangleq \mathbf{c} \mathbf{X}_t$, $\mathbf{A}_t \triangleq \mathbf{A} \mathbf{X}_t$, and \mathbf{X}_t is a square matrix with the components of $\mathbf{x}^{(t)}$ on its diagonal.

Our Frannie’s Firewood application has

$$\mathbf{c} = (90, 150, 0)$$

$$\mathbf{A} = \left(\frac{1}{2}, 1, 1\right), \quad \mathbf{b} = (3)$$

Thus the affine-scaled form corresponding to the $\mathbf{x}^{(t)} = (3, \frac{1}{2}, 1)$ of Figure 7.4 is

$$\begin{aligned} \max \quad & 270y_1 + 75y_2 \\ \text{s.t.} \quad & \frac{3}{2}y_1 + \frac{1}{2}y_2 + y_3 = 3 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

with

$$\mathbf{c}^{(t)} = \mathbf{c}\mathbf{X}_t = (90, 150, 0) \begin{pmatrix} 3 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = (270, 75, 0)$$

$$\mathbf{A}_t = \mathbf{A}\mathbf{X}_t = \left(\frac{1}{2}, 1, 1\right) \begin{pmatrix} 3 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \left(\frac{3}{2}, \frac{1}{2}, 1\right)$$

EXAMPLE 7.7: AFFINE SCALING STANDARD FORM

After 7 moves in an improving search of the linear program

$$\begin{aligned} \min \quad & -3x_1 + 9x_3 \\ \text{s.t.} \quad & -x_1 + x_3 = 3 \\ & x_1 + 2x_2 = 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

$\mathbf{x}^{(7)} = (2, 1, 5)$ has been reached. Derive the corresponding affine-scaled standard-form model.

Solution: Here

$$\mathbf{c} = (-3, 0, 9), \quad \mathbf{A} = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

Corresponding elements of affine-scaled standard form [7.10](#) are

$$\mathbf{c}^{(7)} = \mathbf{c}\mathbf{X}_7 = (-3, 0, 9) \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix} = (-6, 0, 45)$$

$$\mathbf{A}_7 = \mathbf{A}\mathbf{X}_7 = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix} = \begin{pmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{pmatrix}$$

Thus the scaled model is

$$\begin{aligned} \min \quad & -6y_1 + 45y_3 \\ \text{s.t.} \quad & -2y_1 + 5y_3 = 3 \\ & 2y_1 + 2y_2 = 4 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

Projecting on Affine-Scaled Equality Constraints

We have already seen in Section 7.1 (principle [7.6](#)) that interior point algorithms often employ some form of projection to find directions that preserve the equality main constraints of standard form. We will soon see that most of those projections involve scaled problem coefficients $\mathbf{c}^{(t)}$ and \mathbf{A}_t of affine-scaled standard form [7.10](#).

Principle 7.11 The **projection** of direction \mathbf{d} onto affine-scaled conditions $\mathbf{A}_t \Delta \mathbf{y} = \mathbf{0}$ preserving linear inequalities $\mathbf{A}_t \mathbf{y} = \mathbf{b}$ can be computed:

$$\Delta \mathbf{y} = \mathbf{P}_t \mathbf{d}$$

where **projection matrix**

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{A}_t^T (\mathbf{A}_t \mathbf{A}_t^T)^{-1} \mathbf{A}_t)$$

and all other symbols are as in [7.10](#).

EXAMPLE 7.8: PROJECTING IN SCALED y -SPACE

Return to the linear program of Example 7.7 at current solution $\mathbf{x}^{(7)} = (2, 1, 5)$, and compute the projection of direction $\mathbf{d} = (1, 0, -1)$ onto corresponding the standard-form equality constraints in affine-scaled y -space.

Solution: We apply scaled projection formula [7.11](#). From Example 7.7 we have

$$\mathbf{A}_7 = \mathbf{A}\mathbf{X}_7 = \begin{pmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{pmatrix}$$

Thus

$$\mathbf{A}_7 \mathbf{A}_7^T = \begin{pmatrix} 29 & -4 \\ -4 & 8 \end{pmatrix} \quad \text{with} \quad (\mathbf{A}_7 \mathbf{A}_7^T)^{-1} = \begin{pmatrix} \frac{8}{216} & \frac{4}{216} \\ \frac{4}{216} & \frac{29}{216} \end{pmatrix}$$

so

$$\mathbf{P}_7 = (\mathbf{I} - \mathbf{A}_7^T (\mathbf{A}_7 \mathbf{A}_7^T)^{-1} \mathbf{A}_7) = \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix}$$

Thus the projected direction

$$\Delta \mathbf{y} = \mathbf{P}_7 \mathbf{d} = \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} \frac{5}{18} \\ -\frac{5}{18} \\ \frac{1}{9} \end{pmatrix}$$

Computational Effort in Interior Point Computations

The distinction between revised projection formulas of [7.11] and the unscaled formulas of [7.6] may seem rather trivial. Haven't we just changed from \mathbf{A} , \mathbf{x} , and \mathbf{P} to \mathbf{A}_t , \mathbf{y} , and \mathbf{P}_t ?

In fact, the difference is much more profound and accounts for most of the computational effort in interior point algorithms. The key insight is that the model constraint matrix \mathbf{A} does not change as the algorithm proceeds. Thus its projection matrix \mathbf{P} does not change either. If algorithms required only projection onto $\mathbf{A}\mathbf{x} = \mathbf{b}$, some representation of \mathbf{P} could be computed once and stored. Thereafter, each move would involve only choosing a desired direction and multiplying by the representation of \mathbf{P} .

Contrast now with the fact that algorithms actually have to project onto scaled constraints $\mathbf{A}_t \mathbf{y} = \mathbf{b}$ (or something similar). Since $\mathbf{A}_t \triangleq \mathbf{A}\mathbf{X}_t$, \mathbf{A}_t changes at every move with current solution $\mathbf{x}^{(t)}$, it follows that new projection computations have to be performed at every move. Clever techniques of numerical linear algebra can do much to make the computation more efficient, but the fact remains that a very considerable effort has to be expended at each move of the search.

Principle 7.12 The bulk of the computational effort in most interior point algorithms is devoted to projection operations on scaled constraint matrices such as \mathbf{A}_t , which change with each solution visited.

7.3 AFFINE SCALING SEARCH

The affine problem scaling of Section 7.2 produces a new version of the model for which scaled current solution $\mathbf{y}^{(t)}$ is = 1 in every component. Thus the transformed point is nearly equidistant from all boundary (nonnegativity) constraints. It is natural now to think of computing a search move in this simpler scaled solution space and then converting it back to the true decision variables \mathbf{x} via unscaling formula [7.9]. In this section we develop an affine scaling form of interior point search for linear programs that adopts just such a strategy.

Affine Scaling Move Directions

Principle [7.1] tells us that the preferred move direction in the scaled problem follows objective function vector $\mathbf{c}^{(t)}$. Projection formula [7.11] computes the closest $\Delta \mathbf{y}$ satisfying feasible direction requirement $\mathbf{A}_t \Delta \mathbf{y} = \mathbf{0}$. Combining gives the direction

$$\Delta \mathbf{y} = \mathbf{P}_t \mathbf{c}^{(t)} \tag{7.7}$$

for a maximize problem, and

$$\Delta \mathbf{y} = -\mathbf{P}_t \mathbf{c}^{(t)} \quad (7.8)$$

for a minimize problem.

To complete a move direction we have only to translate back into the original variables via inverse scaling formula [7.9].

Principle 7.13 An affine scaling search that has reached feasible solution $\mathbf{x}^{(t)} > \mathbf{0}$ moves next in the direction

$$\Delta \mathbf{x} = \pm \mathbf{X}_t \mathbf{P}_t \mathbf{c}^{(t)}$$

(+ to maximize and – to minimize) where $\mathbf{c}^{(t)}$, \mathbf{X}_t , and \mathbf{P}_t are as in principles [7.10] and [7.11].

We can illustrate with the initial solution $\mathbf{x}^{(0)} = (1, \frac{1}{2}, 2)$ to our Frannie's Firewood application of Figure 7.4. Scaled standard-form principle [7.10] makes

$$\mathbf{c}^{(0)} = \mathbf{cX}_0 = (90, 150, 0) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} = (90, 75, 0)$$

and

$$\mathbf{A}_0 = \mathbf{AX}_0 = (\frac{1}{2}, 1, 1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} = (\frac{1}{2}, \frac{1}{2}, 2)$$

Thus

$$\mathbf{P}_0 = [1 - \mathbf{A}_0^T (\mathbf{A}_0 \mathbf{A}_0^T)^{-1} \mathbf{A}_0] = \begin{pmatrix} \frac{17}{18} & -\frac{1}{18} & -\frac{2}{9} \\ -\frac{1}{18} & \frac{17}{18} & -\frac{2}{9} \\ -\frac{2}{9} & -\frac{2}{9} & \frac{1}{9} \end{pmatrix}$$

We can now compute our maximizing [7.13] move direction as

$$\begin{aligned} \Delta \mathbf{x} &= \mathbf{X}_0 \mathbf{P}_0 \mathbf{c}^{(0)} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} \frac{17}{18} & -\frac{1}{18} & -\frac{2}{9} \\ -\frac{1}{18} & \frac{17}{18} & -\frac{2}{9} \\ -\frac{2}{9} & -\frac{2}{9} & \frac{1}{9} \end{pmatrix} \begin{pmatrix} 90 \\ 75 \\ 0 \end{pmatrix} = \begin{pmatrix} 80\frac{5}{6} \\ 32\frac{11}{12} \\ -73\frac{1}{3} \end{pmatrix} \quad (7.9) \end{aligned}$$

EXAMPLE 7.9: COMPUTING AN AFFINE-SCALED MOVE DIRECTION

Return to the linear program of Examples 7.7 and 7.8 at current solution $\mathbf{x}^{(7)} = (2, 1, 5)$. Compute the next affine scaling move direction.

Solution: From Example 7.7,

$$\mathbf{c}^{(7)} = (-6, 0, 45) \quad \text{and} \quad \mathbf{A}_7 = \begin{pmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{pmatrix}$$

and from Example 7.8,

$$\mathbf{P}_7 = \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix}$$

Continuing with formula [7.13](#) for a minimizing problem gives

$$\Delta \mathbf{x}^{(8)} = - \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix} \begin{pmatrix} -6 \\ 0 \\ 45 \end{pmatrix} = \begin{pmatrix} -11\frac{1}{9} \\ 5\frac{5}{9} \\ -11\frac{1}{9} \end{pmatrix}$$

Feasibility and Improvement of Affine Scaling Directions

Alert readers will notice that our derivation of direction [7.13](#) is guided by rules to produce an improving and feasible move in scaled y -space. But we are using the direction in the original x -space. Will the familiar improving and feasible direction properties of improving search be preserved in that original problem setting?

Fortunately, the answer is yes.

Principle 7.14 (Nonzero) affine scaling search directions derived by formula [7.13](#) are improving and feasible for the original model over \mathbf{x} variables.

For feasibility, the $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$ required by condition [7.4](#) is the same as $\mathbf{A}\mathbf{X}_t\Delta \mathbf{y} = \mathbf{0}$, which must be true when projected on $\mathbf{A}_t\Delta \mathbf{y} = \mathbf{A}\mathbf{X}_t\Delta \mathbf{y}$. The argument for improvement is much the same as the one given for unscaled projection [7.7](#) in Section 7.1.

We may illustrate [7.14](#) with Frannie's Firewood direction

$$\Delta \mathbf{x}^{(1)} = (80\frac{5}{6}, 32\frac{11}{12}, -73\frac{1}{3})$$

of expression (7.9). Since we are in the interior, feasibility requires only $\frac{1}{2} \Delta x_1 + \Delta x_2 + \Delta x_3 = 0$ (condition [7.4](#)). Checking, we find that

$$\frac{1}{2} (80\frac{5}{6}) + (32\frac{11}{12}) + (-73\frac{1}{3}) = 0$$

Similarly, improvement for this maximize problem requires that $\mathbf{c} \cdot \Delta \mathbf{x} > 0$. Here

$$(90, 150, 0) \cdot (80\frac{5}{6}, 32\frac{11}{12}, -73\frac{1}{3}) \approx 12212 > 0$$

Affine Scaling Step Size

With directions of construction [7.13](#) in hand, the next question involves how far to move. That is, we need to choose an appropriate improving search step size λ .

As usual, there may be no limit on progress in direction $\Delta \mathbf{x}$. For linear programs in standard form, the only inactive constraints at interior point solution $\mathbf{x}^{(t)} > \mathbf{0}$ are the nonnegativities. If $\Delta \mathbf{x}$ decreases no component of the solution, we can improve forever without encountering a nonnegativity constraint.

Principle 7.15 A linear program in standard form is unbounded if affine scaling construction [7.13] ever produces a direction $\Delta \mathbf{x} \geq \mathbf{0}$.

When affine scaling produces a move direction with some negative components, we must limit the step so that

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x} \geq \mathbf{0}$$

There is also a second consideration. Definition [7.2] observed that interior point algorithms avoid the boundary until an optimal solution is obtained. That is, we should move all the way until a nonnegativity constraint is active only if such a move will produce an optimal solution.

There are many step size rules that fulfill both of these requirements. The one we will adopt is best understood in affine-scaled space over y -variables. Scaling makes the current solution there a vector of 1s. For example, Figure 7.5 displays current scaled solution $\mathbf{y}^{(0)} = (1, 1, 1)$. The highlighted unit sphere around $\mathbf{y}^{(0)}$ shows how a move of length 1 in any direction maintains all nonnegativity constraints. We can implement affine scaling by stepping to the limits of such a sphere.

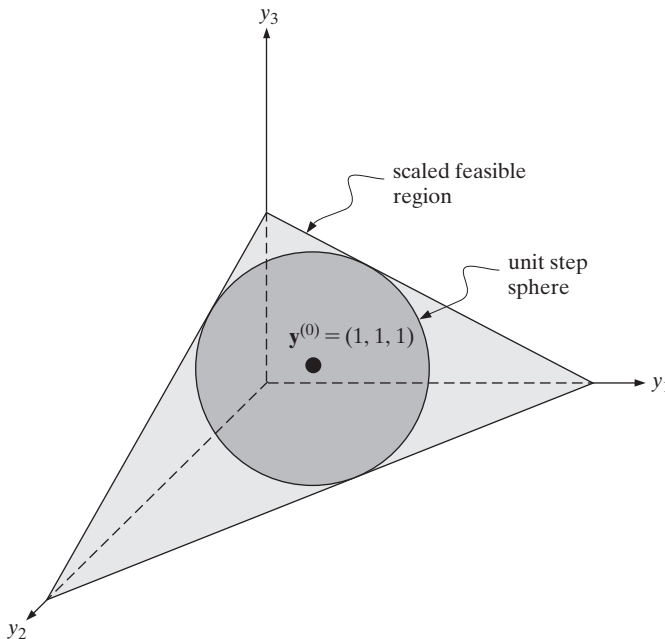


FIGURE 7.5 Feasible Unit Sphere in the Frannie Firewood Application

Principle 7.16 At the current feasible solution $\mathbf{x}^{(t)} > \mathbf{0}$ of a linear program in standard form, if the $\Delta \mathbf{x}$ computed from [7.13] is negative in some components, affine scaling search applies step size

$$\lambda = \frac{1}{\|\Delta \mathbf{x} \mathbf{X}_t^{-1}\|} = \frac{1}{\|\Delta \mathbf{y}\|}$$

where $\|d\|$ denotes $\sqrt{\sum_j (d_j)^2}$, the length or **norm** of vector \mathbf{d} .

Figure 7.5 shows this move to the limit of the sphere along the direction computed at equation (7.9). The scaled direction is

$$\Delta \mathbf{y} = \mathbf{X}_0^{-1} \Delta \mathbf{x} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 80\frac{5}{6} \\ 32\frac{11}{12} \\ -73\frac{1}{3} \end{pmatrix} = \begin{pmatrix} 80\frac{5}{6} \\ 65\frac{5}{6} \\ -36\frac{2}{3} \end{pmatrix}$$

with length (norm)

$$\|\Delta \mathbf{y}\| = \sqrt{(80\frac{5}{6})^2 + (65\frac{5}{6})^2 + (-36\frac{2}{3})^2} \approx 110.5$$

Thus the computation [7.16] step of

$$\lambda = \frac{1}{\|\Delta \mathbf{y}\|} = \frac{1}{110.5} \approx 0.00905 \tag{7.10}$$

brings us exactly to the boundary of the unit sphere.

EXAMPLE 7.10: COMPUTING AFFINE SCALING STEP SIZE

Determine the appropriate step size to apply to the move direction $\Delta \mathbf{x} = (-11\frac{1}{9}, 5\frac{5}{9}, -11\frac{1}{9})$ derived in Example 7.9 at point $\mathbf{x}^{(7)} = (2, 1, 5)$.

Solution: Applying principle [7.16] yields

$$\Delta \mathbf{y} = \mathbf{X}_7^{-1} \Delta \mathbf{x} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{5} \end{pmatrix} \begin{pmatrix} -11\frac{1}{9} \\ 5\frac{5}{9} \\ -11\frac{1}{9} \end{pmatrix} = \begin{pmatrix} -5\frac{5}{9} \\ 5\frac{5}{9} \\ -2\frac{2}{9} \end{pmatrix}$$

This $\Delta \mathbf{y}$ has

$$\|\Delta \mathbf{y}\| = \sqrt{(-5\frac{5}{9})^2 + (5\frac{5}{9})^2 + (2\frac{2}{9})^2} \approx 8.165$$

Thus the appropriate step is

$$\lambda = \frac{1}{8.165} = 0.1225$$

Termination in Affine Scaling Search

Applying the step computed in expression (6.10) advances our search of the Frannie's firewood application to point

$$\begin{aligned}\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x} \\ &= (0.5, 1, 1) + 0.00905(80.83, 32.92, -73.33) \\ &\approx (1.73, 0.80, 1.34)\end{aligned}\quad (7.11)$$

Notice that the new point remains positive in all components and thus in the interior of the feasible region.

With step rule [7.16](#) constructed to keep updated (scaled) points within the unit sphere of feasible \mathbf{y} , the new solution will always be interior unless the search moves to one of the points where that sphere touches a nonnegativity constraint. Such moves are rare, but if they occur, the new solution can be shown to be *optimal*.

Much more typically, an affine scaling search will remain in the interior, stepping ever closer to an optimal solution on the boundary. Then a stopping rule is required to terminate computation when a solution becomes sufficiently close to an optimum.

But how do we know that we are near an optimal solution? Crude schemes simply stop when the solution value is no longer changing very much (as with Algorithm 7A). More precise rules derive bounds on the optimal solution value at

ALGORITHM 7A: AFFINE SCALING SEARCH FOR LINEAR PROGRAMS

Step 0: Initialization. Choose any starting feasible interior point solution, $\mathbf{x}^{(0)} > \mathbf{0}$, and set solution index $t \leftarrow 0$.

Step 1: Optimality. If any component of $\mathbf{x}^{(t)}$ is 0, or if recent algorithm steps have made no significant change in the solution value, stop. Current point $\mathbf{x}^{(t)}$ is either optimal in the given LP or very nearly so.

Step 2: Move Direction. Construct the next move direction by projecting in affine-scaled space as

$$\Delta \mathbf{x}^{(t+1)} \leftarrow \pm \mathbf{X}_t \mathbf{P}_t \mathbf{c}^{(t)}$$

(+ to maximize, – to minimize) where \mathbf{X}_t , \mathbf{P}_t , and $\mathbf{c}^{(t)}$ are the scaled values of Section 7.2.

Step 3: Step Size. If there is no limit on feasible moves in direction $\Delta \mathbf{x}^{(t+1)}$ (all components are nonnegative), stop; the given model is unbounded. Otherwise, construct step size

$$\lambda \leftarrow \frac{1}{\|\Delta \mathbf{x}^{(t+1)} \mathbf{X}_t^{-1}\|}$$

Step 4: Advance. Compute the new solution

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)}$$

Then advance $t \leftarrow t + 1$, and return to Step 1.

each step so that we can know how much room remains for improvement. We look at the latter briefly in Section 7.5.

Affine Scaling Search of the Frannie’s Firewood Application

Algorithm 7A collects all the insights of this section in an affine scaling algorithm for linear programs in standard form. Table 7.1 and Figure 7.6 detail its application to our Frannie’s firewood application.

TABLE 7.1 Affine Scaling Search of the Frannie’s Firewood Application

	x_1	x_2	x_3	
max c	90	150	0	b
A	0.5	1	1	3
$\mathbf{x}^{(0)}$	1.00	0.50	2.00	$\mathbf{c} \cdot \mathbf{x}^{(0)} = 165.00$
$\Delta \mathbf{x}^{(1)}$	80.83	32.92	-73.33	$\lambda = 0.00905$
$\mathbf{x}^{(1)}$	1.73	0.80	1.34	$\mathbf{c} \cdot \mathbf{x}^{(1)} = 275.51$
$\Delta \mathbf{x}^{(2)}$	160.94	49.25	-129.72	$\lambda = 0.00676$
$\mathbf{x}^{(2)}$	2.82	1.13	0.46	$\mathbf{c} \cdot \mathbf{x}^{(2)} = 423.40$
$\Delta \mathbf{x}^{(3)}$	87.26	-10.29	-33.34	$\lambda = 0.0126$
$\mathbf{x}^{(3)}$	3.92	1.00	0.04	$\mathbf{c} \cdot \mathbf{x}^{(3)} = 502.84$
$\Delta \mathbf{x}^{(4)}$	48.13	-23.79	-0.27	$\lambda = 0.0362$
$\mathbf{x}^{(4)}$	5.66	0.14	0.03	$\mathbf{c} \cdot \mathbf{x}^{(4)} = 530.45$
$\Delta \mathbf{x}^{(5)}$	1.49	-0.58	-0.16	$\lambda = 0.1472$
$\mathbf{x}^{(5)}$	5.88	0.05	0.01	$\mathbf{c} \cdot \mathbf{x}^{(5)} = 537.25$
$\Delta \mathbf{x}^{(6)}$	0.19	-0.09	-0.01	$\lambda = 0.5066$
$\mathbf{x}^{(6)}$	5.98	0.01	0.003	$\mathbf{c} \cdot \mathbf{x}^{(6)} = 539.22$
$\Delta \mathbf{x}^{(7)}$	0.008	-0.003	-0.001	$\lambda = 1.7689$
$\mathbf{x}^{(7)}$	5.99	0.005	+0.000	$\mathbf{c} \cdot \mathbf{x}^{(7)} = 539.79$
$\Delta \mathbf{x}^{(8)}$	0.001	-0.001	-0.000	$\lambda = 6.3305$
$\mathbf{x}^{(8)}$	6.00	0.001	+0.000	$\mathbf{c} \cdot \mathbf{x}^{(8)} = 539.95$
$\Delta \mathbf{x}^{(9)}$	+0.000	-0.000	-0.000	$\lambda = 23.854$
$\mathbf{x}^{(9)}$	6.00	+0.000	+0.000	$\mathbf{c} \cdot \mathbf{x}^{(9)} = 539.99$

Notice that the search makes very rapid progress while it is near the “middle” of the feasible region. Later iterations approach the boundary but never quite reach it. We terminate after the ninth iteration changes the objective function by less than 0.1.

7.4 LOG BARRIER METHODS FOR INTERIOR POINT SEARCH

Affine scaling is only one way to keep algorithms away from the boundary. In this section we develop a **log barrier** alternative.

Barrier Objective Functions

Remembering that the boundary of a standard-form linear program is defined by nonnegativity constraints, log barrier methods exploit the fact that $\ln(x_j) \rightarrow -\infty$ as

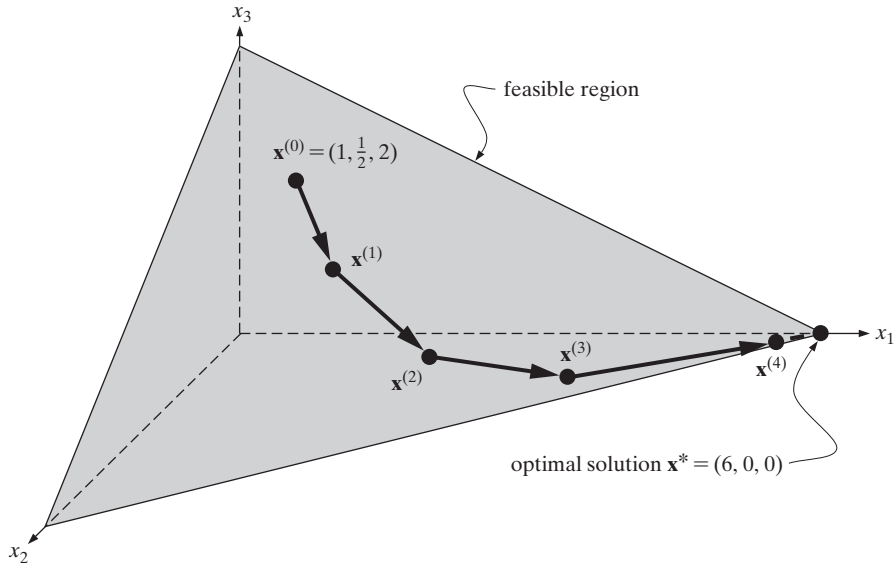


FIGURE 7.6 Affine Scaling Search of the Frannie's Firewood Application

$x_j \rightarrow 0$. A modified barrier objective function includes $\ln(x_j)$ terms to keep the x_j away from the boundary.

Principle 7.17 A maximize objective $\sum_j c_j x_j$ of a standard-form linear program is modified with **in log barrier methods** as

$$\max \sum_j c_j x_j + \mu \sum_j \ln(x_j)$$

where $\mu > 0$ is a specified weighting constant. The corresponding form for a minimize objective is

$$\min \sum_j c_j x_j - \mu \sum_j \ln(x_j)$$

We can illustrate with our familiar Frannie's firewood standard form

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 \\ \text{s.t.} \quad & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Adding a barrier term produces the modified model

$$\begin{aligned} \max \quad & 90x_1 + 150x_2 + \mu [\ln(x_1) + \ln(x_2) + \ln(x_3)] \\ \text{s.t.} \quad & \frac{1}{2}x_1 + x_2 + x_3 = 3 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{7.12}$$

Suppose that we (arbitrarily) choose multiplier $\mu = 64$. Then the objective at feasible point $\mathbf{x} = (2, 1, 1)$, which is far from the boundary, evaluates to

$$90(2) + 150(1) + 0(1) + 64 [\ln(2) + \ln(1) + \ln(1)] \approx 374.36$$

The log-barrier terms do have an influence, because the true objective value at this \mathbf{x} is $90(2) + 150(1) = 330$. Still, the difference is modest.

Compare with near-boundary feasible point $\mathbf{x} = (0.010, 2.99, 0.005)$. There the barrier objective function evaluates as

$$\begin{aligned} &90(0.010) + 150(2.99) + 0(0.005) + 64 [\ln(0.010) + \ln(2.99) + \ln(0.005)] \\ &= 449.4 + 64(-4.605 + 1.095 - 5.298) \\ &\approx -114.31 \end{aligned}$$

The corresponding true objective value is $90(0.010) + 150(2.99) = 449.4$. We see that the negative logarithms of x_j near 0.0 have severely penalized this solution in the modified maximize objective.

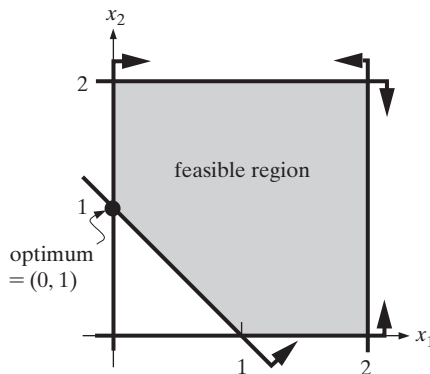
It is in this penalization sense that modified objective functions of principle 7.17 erect a “barrier” to near-boundary solutions. As components x_j approach their boundary value of 0.0, penalties become larger and larger, thus guaranteeing that no improving search will choose to approach the boundary too closely.

EXAMPLE 7.11: FORMING LOG BARRIER OBJECTIVES

Linear program

$$\begin{aligned} \min \quad &5x_1 + 3x_2 \\ \text{s.t.} \quad &x_1 + x_2 \geq 1 \\ &0 \leq x_1 \leq 2 \\ &0 \leq x_2 \leq 2 \end{aligned}$$

has feasible region as displayed in the following figure:



- (a) Place the model in standard form.
- (b) Show the modified objective function obtained when a logarithmic barrier is introduced to discourage approaching the boundary.

(c) Choose a solution near the middle of the feasible region, and compare the true and modified objective functions at that point using $\mu = 10$.

(d) Choose an interior point solution near the boundary of the feasible region, and compare the true and modified objective functions at that point using the same $\mu = 10$.

Solution:

(a) Introducing slack variables x_3 , x_4 , and x_5 , the standard-form model is

$$\begin{aligned} \min \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & x_1 + x_2 - x_3 = 1 \\ & x_1 + x_4 = 2 \\ & x_2 + x_5 = 2 \\ & x_1, \dots, x_5 \geq 0 \end{aligned}$$

(b) Following principle [7.17](#), the barrier objective with multiplier μ is

$$\min 5x_1 + 3x_2 - \mu [\ln(x_1) + \ln(x_2) + \ln(x_3) + \ln(x_4) + \ln(x_5)]$$

(c) From the figure it is clear that $x_1 = x_2 = \frac{5}{4}$ is near the middle of the feasible region. There the true objective function value is $5(\frac{5}{4}) + 3(\frac{5}{4}) = 10$. Corresponding values for slack variables are $x_3 = \frac{5}{4} + \frac{5}{4} - 1 = \frac{3}{2}$ and $x_4 = x_5 = 2 - \frac{5}{4} = \frac{3}{4}$. Thus the log barrier objective with $\mu = 10$ evaluates

$$5(\frac{5}{4}) + 3(\frac{5}{4}) - 10 [\ln(\frac{5}{4}) + \ln(\frac{5}{4}) + \ln(\frac{3}{2}) + \ln(\frac{3}{4}) + \ln(\frac{3}{4})] \approx 7.237$$

Modified cost is a bit lower than the true objective value.

(d) One point near the boundary is $x_1 = 1.999$, $x_2 = 0.001$. There the true objective function value is $5(1.999) + 3(0.001) = 9.998$. Corresponding values for slack variables are $x_3 = 1.999 + 0.001 - 1 = 1.000$, $x_4 = 2 - 1.999 = 0.001$, and $x_5 = 2 - 0.001 = 1.999$. Thus the log barrier objective with $\mu = 10$ evaluates

$$\begin{aligned} & 5(1.999) + 3(0.001) - 10 [\ln(1.999) + \ln(0.001) + \ln(1.000) \\ & + \ln(1.999) + \ln(0.001)] \approx 134.3 \end{aligned}$$

This near-boundary point is penalized severely.

Problems with Gradient Directions

To make any progress on our underlying LP with barrier methods, we must find improving feasible directions for the now nonlinear standard form

$$\begin{aligned} \max \text{ or } \min \quad & f(\mathbf{x}) \triangleq \sum_j [c_j x_j \pm \mu \ln(x_j)] \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \end{aligned} \tag{7.13}$$

Nonnegativity constraints can be ignored because the barrier objective will keep us away from the boundary.

Although the gradient-based directions introduced in Section 3.3 have served us well so far, they usually perform poorly with nonlinear objective functions

(see Section 16.5 for details). In essence they treat the objective as if it were approximated as

$$f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) \approx f(\mathbf{x}^{(t)}) + \lambda \sum_j \frac{\partial f}{\partial x_j} \Delta x_j \tag{7.14}$$

where $\mathbf{x}^{(t)}$ is the current solution and $\Delta \mathbf{x}$ a proposed move direction. This approximation, known as the first-order **Taylor series** (see also Section 16.3), takes the function to be roughly its current value plus the net effect of partial derivative slopes times move components. Choosing gradient-based direction $\Delta \mathbf{x} = \pm \nabla f(\mathbf{x}^{(t)})$ produces the most rapid change in the approximation per unit λ .

With a linear objective function, partial derivatives are constant and approximation (7.14) is exact. But with nonlinear cases, the slope information in partial derivatives can decay rapidly as we move away from point $\mathbf{x}^{(t)}$. The result is that a direction based on rapid progress for the simple (7.14) approximation may prove very ineffective for the real nonlinear objective.

Newton Steps for Barrier Search

For those who remember their calculus (refer to Primers 2 and 7 if needed), it is natural to think of enhancing approximation (7.14) with second partial derivatives. The corresponding second-order Taylor form is

$$f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) \approx f(\mathbf{x}^{(t)}) + \lambda \sum_j \frac{\partial f}{\partial x_j} \Delta x_j + \frac{\lambda^2}{2} \sum_j \sum_k \frac{\partial^2 f}{\partial x_j \partial x_k} \Delta x_j \Delta x_k \tag{7.15}$$

New terms account for changes in the rates of change $\partial f / \partial x_j$ as we move away from $\mathbf{x}^{(t)}$.

The partial derivatives needed in this new approximation have a particularly easy form for our log barrier objective [7.17](#) (+ for maximize, – for minimize):

$$\begin{aligned} \frac{\partial f}{\partial x_j} &= c_j \pm \frac{\mu}{x_j} \\ \frac{\partial^2 f}{\partial x_j \partial x_k} &= \begin{cases} \mp \frac{\mu}{(x_j)^2} & \text{if } j=k \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7.16}$$

Thus fixing $\lambda = 1$ for the moment and taking second-order approximation as exact, it makes sense to choose a move direction $\Delta \mathbf{x}$ at $\mathbf{x}^{(t)}$ that solves

$$\max \text{ or } \min \quad f(\mathbf{x}^{(t)} + \Delta \mathbf{x}) \approx \sum_j \left[c_j \pm \frac{\mu}{x_j^{(t)}} \mp \frac{1}{2} \frac{\mu}{(x_j^{(t)})^2} (\Delta x_j)^2 \right] \tag{7.17}$$

s.t. $\mathbf{A} \Delta \mathbf{x} = \mathbf{0}$

That is, we choose the $\Delta \mathbf{x}$ that most improves our barrier objective function—as approximated by Taylor expression (7.15)—subject to the familiar requirements

(principle [7.4](#)) that the direction preserve all equality constraints of standard form (7.13).

Using Lagrange multiplier methods is beyond the scope of this section (but developed in Section 17.3); it can be shown that the $\Delta \mathbf{x}$ which solves the direction-finding problem (7.17) is remarkably like the affine-scaled steps of Section 7.3. It takes the form

$$\Delta \mathbf{x} = \pm \frac{1}{\mu} \mathbf{X}_t \mathbf{P}_t \begin{pmatrix} c_1^{(t)} \pm \mu \\ \vdots \\ c_n^{(t)} \pm \mu \end{pmatrix} \quad (7.18)$$

where \mathbf{X}_t , \mathbf{P}_t , and $\mathbf{c}^{(t)}$ are the scaled problem data of [7.10](#) and [7.11](#) in Section 7.2.

The only difference from affine scaling direction [7.13](#) is the inclusion of barrier multiplier μ in forming the scaled direction to project.

This move is often called a **Newton step** because it is based on the same second-order Taylor approximation (7.15) that gives rise to the famous Newton method for equation solving and unconstrained optimization (see Section 16.6 for a full development). Allowing a step size λ to be applied, which means that we may drop the leading constant $1/\mu$, yields the directions to be employed in our barrier form of interior point LP search.

Principle 7.18 | A Newton step barrier algorithm that has reached feasible solution $\mathbf{x}^{(t)} > \mathbf{0}$ with barrier multiplier $\mu > 0$ moves next in direction

$$\Delta \mathbf{x} = \pm \mathbf{X}_t \mathbf{P}_t \begin{pmatrix} c_1^{(t)} \pm \mu \\ \vdots \\ c_n^{(t)} \pm \mu \end{pmatrix}$$

(+ to maximize and – to minimize) where \mathbf{X}_t , \mathbf{P}_t , and $\mathbf{c}^{(t)}$ are the affine-scaled problem data of definitions [7.10](#) and [7.11](#).

To illustrate principle [7.18](#), we return to our Frannie’s Firewood application at $\mathbf{x}^{(0)} = (1, \frac{1}{2}, 2)$ with $\mu = 16$. Scaling of Section 7.2 gives

$$\mathbf{c}^{(0)} = \mathbf{cX}_0 = (90, 150, 0) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} = (90, 75, 0)$$

and

$$\mathbf{A}_0 = \mathbf{AX}_0 = \left(\frac{1}{2}, 1, 1\right)$$

Thus

$$\mathbf{P}_0 = (\mathbf{I} - \mathbf{A}_0^T(\mathbf{A}_0\mathbf{A}_0^T)^{-1}\mathbf{A}_0) = \begin{pmatrix} \frac{17}{18} & -\frac{1}{18} & -\frac{2}{9} \\ -\frac{1}{18} & \frac{17}{18} & -\frac{2}{9} \\ -\frac{2}{9} & -\frac{2}{9} & \frac{1}{9} \end{pmatrix}$$

Now projecting to find the next move direction,

$$\begin{aligned} \Delta \mathbf{x}^{(1)} &= \mathbf{X}_0 \mathbf{P}_0 \begin{pmatrix} c_1^{(0)} + \mu \\ \vdots \\ c_n^{(0)} + \mu \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} \frac{17}{18} & -\frac{1}{18} & -\frac{2}{9} \\ -\frac{1}{18} & \frac{17}{18} & -\frac{2}{9} \\ -\frac{2}{9} & -\frac{2}{9} & \frac{1}{9} \end{pmatrix} \begin{pmatrix} 90 + 16 \\ 75 + 16 \\ 0 + 16 \end{pmatrix} \\ &= (91\frac{1}{2}, 38\frac{1}{4}, -84) \end{aligned}$$

EXAMPLE 7.12: COMPUTING NEWTON STEP BARRIER SEARCH DIRECTIONS

Previous Example 7.7 and 7.8 considered the model

$$\begin{aligned} \min \quad & -3x_1 + 9x_3 \\ \text{s.t.} \quad & -x_1 + x_3 = 3 \\ & x_1 + 2x_2 = 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

at current solution $\mathbf{x}^{(7)} = (2, 1, 5)$. Compute the next Newton step barrier search direction assuming a barrier multiplier $\mu = 120$.

Solution: From the earlier sample exercises we know that

$$\mathbf{c}^{(7)} = \mathbf{cX}_7 = (-6, 0, 45)$$

at the specified $\mathbf{x}^{(7)}$,

$$\mathbf{A}_7 = \mathbf{AX}_7 = \begin{pmatrix} -2 & 0 & 5 \\ 2 & 2 & 0 \end{pmatrix}$$

and

$$\mathbf{P}_7 = (\mathbf{I} - \mathbf{A}_7^T(\mathbf{A}_7\mathbf{A}_7^T)^{-1}\mathbf{A}_7) = \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix}$$

Thus principle [7.18](#) for this minimizing problem gives

$$\begin{aligned}\Delta \mathbf{x}^{(8)} &= -\mathbf{X}_7 \mathbf{P}_7 \begin{pmatrix} c_1^{(7)} - \mu \\ \vdots \\ c_n^{(7)} - \mu \end{pmatrix} \\ &= -\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} \frac{25}{54} & -\frac{25}{54} & \frac{10}{54} \\ -\frac{25}{54} & \frac{25}{54} & -\frac{10}{54} \\ \frac{10}{54} & -\frac{10}{54} & \frac{4}{54} \end{pmatrix} \begin{pmatrix} -6 - 120 \\ 0 - 120 \\ 45 - 120 \end{pmatrix} = \begin{pmatrix} 33\frac{1}{3} \\ -16\frac{2}{3} \\ 33\frac{1}{3} \end{pmatrix}\end{aligned}$$

Newton Step Barrier Search Step Sizes

As usual, our next concern is how big a step to take in barrier search direction [7.18](#). Any such rule must first assure that

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)} > \mathbf{0}$$

so that the new point will also be in the interior.

This is easily accomplished by familiar minimum ratio computations. To remain interior, we keep λ less than or equal to say 90% of the maximum feasible step size, that is,

$$\lambda \leq 0.9\lambda_{\max} \tag{7.19}$$

where

$$\lambda_{\max} = \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\}$$

With barrier objective [7.17](#) nonlinear, there is also the possibility that improvement stops before we approach λ_{\max} . A direction that improves near current $\mathbf{x}^{(t)}$ may begin to degrade the barrier objective function value after a larger step.

Figure 7.7 illustrates both cases. At Fannie's firewood, $\mathbf{x}^{(0)} = (1, \frac{1}{2}, 2)$, direction $\Delta \mathbf{x}^{(1)} = (91.5, 38.25, -84)$ decreases only x_3 . Thus expression (6.19) yields

$$0.9\lambda_{\max} = 0.9 \left(\frac{2}{84} \right) = 0.0214$$

Part (a) of the figure shows that the barrier function improves throughout the range $0 \leq \lambda \leq 0.0214$.

Contrast with the more typical step at $\mathbf{x}^{(t)} = (5.205, 0.336, 0.062)$. The corresponding $\Delta \mathbf{x}^{(t+1)} = (-3.100, 1.273, 0.278)$. Only x_1 decreases and the minimum ratio expression (7.19) gives

$$0.9 \left(\frac{x_1^{(t)}}{-\Delta x_1^{(t+1)}} \right) = 0.9 \left(\frac{5.205}{3.100} \right) = 1.511$$

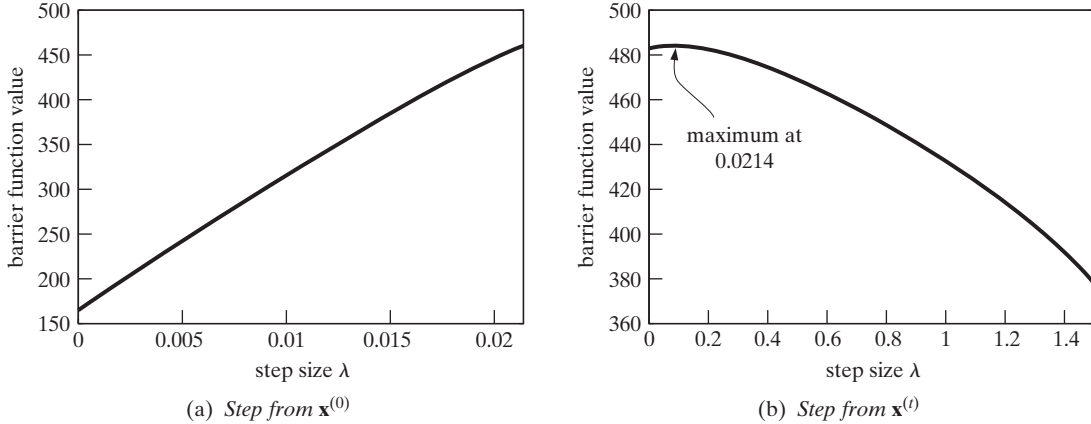


FIGURE 7.7 Frannie’s Firewood Barrier Function Changes with Step Size

However, Figure 7.7(b) shows that the barrier function reaches a maximum long before this limit is encountered.

We need an enhancement of the basic minimum ratio step size rule to account for this possibility of initial improvement followed by worsening of the barrier objective. Fortunately, one is readily available in the Newton step computations of expression (7.18). That best move for the second-order Taylor approximation consisted of $1/\mu$ times the barrier search direction of principle [7.18].

Combining this idea with a minimum ratio gives a step size that both keeps to the interior and approximately optimizes the barrier objective:

Principle 7.19 At current feasible solution $\mathbf{x}^{(t)} > \mathbf{0}$ and barrier multiplier $\mu > 0$, a Newton step barrier search algorithm should apply step size

$$\lambda = \min \left\{ \frac{1}{\mu}, 0.9\lambda_{\max} \right\}$$

to the move direction of [7.18], where

$$\lambda_{\max} = \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\}$$

EXAMPLE 7.13: COMPUTING BARRIER SEARCH STEP SIZES

- Return to the minimize problem of Example 7.12.
- (a) Determine the maximum step that can be applied to the direction $\Delta \mathbf{x}^{(8)}$ computed there before the boundary is reached.
 - (b) Sketch the barrier objective function for that model as a function of the step size λ applied to direction $\Delta \mathbf{x}^{(8)}$.

(c) Compute the step size that would be applied by a Newton step barrier search algorithm.

Solution:

(a) In Sample Exercise 7.12 we computed the next move direction,

$$\Delta \mathbf{x}^{(8)} = \left(33\frac{1}{3}, -16\frac{2}{3}, 33\frac{1}{3} \right)$$

Applying the minimum ratio computation (7.19), the boundary would be reached at

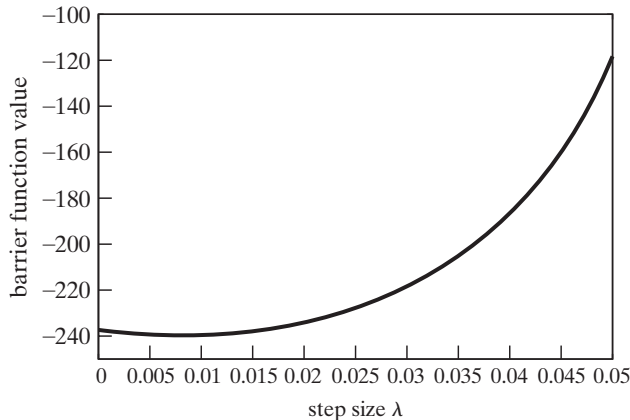
$$\begin{aligned} \lambda_{\max} &= \min \left\{ \frac{x_j^{(7)}}{-\Delta x_j^{(8)}} : \Delta x_j^{(t+1)} < 0 \right\} \\ &= \min \left\{ \frac{1}{16\frac{2}{3}} \right\} \\ &= 0.060 \end{aligned}$$

We should stop before, say, 90% or $\lambda = 0.9(0.060) = 0.054$ to stay interior.

(b) The barrier objective function for this exercise is

$$\min -3x_1 + 9x_3 - \mu[\ln(x_1) + \ln(x_2) + \ln(x_3)]$$

Plotting as a function of step size gives the following:



We seek the λ that yields a minimum value without leaving the interior.

(c) Applying rule [7.19], Newton step barrier search would use step size

$$\lambda = \min \left\{ \frac{1}{\mu}, 0.9\lambda_{\max} \right\} = \min \left\{ \frac{1}{120}, 0.054 \right\} = \frac{1}{120} = 0.00833$$

Impact of the Barrier Multiplier μ

Having worked out all the details of a barrier search for any fixed μ in barrier forms [7.17], it is time to consider how to manage that barrier multiplier. Parameter μ controls how much weight is assigned to keeping a search away from the boundary. For example, Table 7.2 details the effect for our Frannie's Firewood barrier

form (7.12). The optimal barrier problem values of x_1 , x_2 , and x_3 are shown for a range of barrier multiplier values. For example, high weight $\mu = 2^{16} = 65,536$ on logarithmic barriers makes the optimal \mathbf{x} in model (7.12) approximately (2, 1, 1). As the multiplier decreases to $\mu = 2^{-5} = \frac{1}{32}$, the optimal \mathbf{x} in (7.12) approaches the true optimum (without barriers) of $\mathbf{x}^* = (6, 0, 0)$.

TABLE 7.2 Impact of μ on the Frannie’s Firewood Application

μ	x_1	x_2	x_3	μ	x_1	x_2	x_3
$2^{16} = 65,536$	2.002	1.001	0.998	$2^5 = 32$	4.250	0.711	0.164
$2^{15} = 32,768$	2.004	1.001	0.997	$2^4 = 16$	4.951	0.439	0.086
$2^{14} = 16,384$	2.009	1.002	0.993	$2^3 = 8$	5.426	0.243	0.044
$2^{13} = 8,192$	2.017	1.005	0.987	$2^2 = 4$	5.702	0.127	0.022
$2^{12} = 4,096$	2.034	1.009	0.974	$2^1 = 2$	5.847	0.065	0.011
$2^{11} = 2,048$	2.068	1.018	0.948	$2^0 = 1$	5.923	0.033	0.006
$2^{10} = 1,024$	2.134	1.035	0.898	$2^{-1} = \frac{1}{2}$	5.961	0.017	0.003
$2^9 = 512$	2.261	1.060	0.809	$2^{-2} = \frac{1}{4}$	5.980	0.009	0.001
$2^8 = 256$	2.494	1.088	0.665	$2^{-3} = \frac{1}{8}$	5.990	0.004	0.001
$2^7 = 128$	2.889	1.079	0.477	$2^{-4} = \frac{1}{16}$	5.995	0.002	0.000
$2^6 = 64$	3.489	0.960	0.295	$2^{-5} = \frac{1}{32}$	5.997	0.001	0.000

Principle 7.20 High values of barrier weight $\mu > 0$ severely penalize interior points near the boundary. Low values encourage the search to approach the boundary.

Barrier Algorithm Multiplier Strategy

The power of multiplier μ to control how close barrier optimal solutions come to the boundary suggests a strategy for using barrier methods to solve the underlying LP.

Principle 7.21 Barrier algorithms begin with multiplier $\mu > 0$ relatively high and slowly reduce it toward zero as the search proceeds.

Initial moves seek good solutions far from the boundary, with later ones coming ever closer to an optimum in the LP as barrier multiplier $\mu \rightarrow 0$.

Newton Step Barrier Algorithm

Algorithm 7B collects all the ideas developed so far in a Newton step barrier algorithm for linear programming. The search begins with an interior (feasible) point $\mathbf{x}^{(0)}$ and a relatively large barrier multiplier μ . For each μ , an inner loop seeks to approximately optimize the corresponding log barrier problem. That is, we take one or more steps to approach the barrier model optimum for that μ .

Once slow progress indicates that we are near enough to the optimal value in the barrier model, an outer loop reduces μ and repeats the inner loop. The process

ALGORITHM 7B: NEWTON STEP BARRIER SEARCH FOR LINER PROGRAM

Step 0: Initialization. Choose any starting feasible interior point solution, $\mathbf{x}^{(0)} > \mathbf{0}$, and a relatively large initial barrier multiplier μ . Also set solution index $t \leftarrow 0$.

Step 1: Move Direction. Construct the next move direction by projecting in affine-scaled space as

$$\Delta \mathbf{x}^{(t+1)} \leftarrow \pm \mathbf{X}_t \mathbf{P}_t \begin{pmatrix} c_1^{(t)} \pm \mu \\ \vdots \\ c_n^{(t)} \pm \mu \end{pmatrix}$$

(+ to maximize and – to minimize) where \mathbf{X}_t , \mathbf{P}_t , and $\mathbf{c}^{(t)}$ are the scaled problem data of Section 7.2.

Step 2: Step Size. Choose a step size

$$\lambda \leftarrow \min \left\{ \frac{1}{\mu}, 0.9\lambda_{\max} \right\}$$

where

$$\lambda_{\max} \leftarrow \min \left\{ \frac{x_j^{(t)}}{-\Delta x_j^{(t+1)}} : \Delta x_j^{(t+1)} < 0 \right\}$$

Step 3: Advance. Compute the new solution

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)}$$

Step 4: Inner Loop. If progress indicates that $\mathbf{x}^{(t+1)}$ is far from optimal for the current μ , increment $t \leftarrow t + 1$, and return to Step 1.

Step 5: Outer Loop. If barrier multiplier μ is near zero, stop. The current point $\mathbf{x}^{(t+1)}$ is either optimal in the given LP or very nearly so. Otherwise, reduce the barrier multiplier μ , increment $t \leftarrow t + 1$, and return to Step 1.

terminates when μ nears zero, indicating that the current solution is very close to an optimum in the LP.

As with affine scaling Algorithm 7A, many details of stopping and convergence are left vague in the statement of Algorithm 7B because they involve mathematical issues beyond the scope of this chapter. However, in Section 7.5 we provide further insight regarding how such issues are handled in commercial-quality algorithms.

Newton Barrier Solution of Frannie's Firewood Application

Table 7.3 illustrates Algorithm 7B on our Frannie's firewood application. The table shows both the true objective function value at every step and the modified

TABLE 7.3 Newton Barrier Search of Frannie’s Firewood Application

	x_1	x_2	x_3	
max c	90	150	0	b
A	0.5	1	1	3
$\mathbf{x}^{(0)}$	1.000	0.500	2.000	obj = 165.00 (165.00)
$\Delta \mathbf{x}^{(1)}$	91.500	38.250	-84.000	$\mu = 16.0, \lambda = 0.0214$
$\mathbf{x}^{(1)}$	2.961	1.320	0.200	obj = 464.41 (460.46)
$\Delta \mathbf{x}^{(2)}$	59.996	-26.113	-3.885	$\mu = 16.0, \lambda = 0.0455$
$\mathbf{x}^{(2)}$	5.689	0.132	0.023	obj = 531.84 (467.12)
$\Delta \mathbf{x}^{(3)}$	-3.519	1.487	0.272	$\mu = 16.0, \lambda = 0.0625$
$\mathbf{x}^{(3)}$	5.470	0.225	0.040	obj = 526.00 (477.93)
$\Delta \mathbf{x}^{(4)}$	-4.227	1.771	0.343	$\mu = 16.0, \lambda = 0.0625$
$\mathbf{x}^{(4)}$	5.205	0.336	0.062	obj = 518.82 (483.19)
$\Delta \mathbf{x}^{(5)}$	-3.100	1.273	0.278	$\mu = 16.0, \lambda = 0.0625$
$\mathbf{x}^{(5)}$	5.012	0.415	0.079	obj = 513.31 (484.44)
$\Delta \mathbf{x}^{(6)}$	-0.917	0.359	0.099	$\mu = 16.0, \lambda = 0.0625$
$\mathbf{x}^{(6)}$	4.954	0.438	0.085	obj = 511.52 (484.51)
$\Delta \mathbf{x}^{(7)}$	6.804	-2.756	-0.646	$\mu = 8.0, \lambda = 0.1188$
$\mathbf{x}^{(7)}$	5.762	0.110	0.009	obj = 535.16 (493.42)
$\Delta \mathbf{x}^{(8)}$	-1.076	0.483	0.055	$\mu = 8.0, \lambda = 0.1250$
$\mathbf{x}^{(8)}$	5.628	0.171	0.015	obj = 532.11 (498.40)
$\Delta \mathbf{x}^{(9)}$	0.426	-0.232	0.019	$\mu = 4.0, \lambda = 0.2500$
$\mathbf{x}^{(9)}$	5.734	0.113	0.020	obj = 533.01 (515.63)
$\Delta \mathbf{x}^{(10)}$	-0.118	0.052	0.007	$\mu = 4.0, \lambda = 0.2500$
$\mathbf{x}^{(10)}$	5.705	0.126	0.022	obj = 532.29 (515.67)
$\Delta \mathbf{x}^{(11)}$	0.552	-0.233	-0.043	$\mu = 2.0, \lambda = 0.4614$
$\mathbf{x}^{(11)}$	5.959	0.018	0.002	obj = 539.06 (522.36)
$\Delta \mathbf{x}^{(12)}$	-0.059	0.026	0.004	$\mu = 2.0, \lambda = 0.5000$
$\mathbf{x}^{(12)}$	5.930	0.031	0.004	obj = 538.35 (523.91)
$\Delta \mathbf{x}^{(13)}$	-0.006	0.002	0.001	$\mu = 1.0, \lambda = 1.0000$
$\mathbf{x}^{(13)}$	5.924	0.033	0.005	obj = 538.10 (531.18)
$\Delta \mathbf{x}^{(14)}$	-0.001	0.000	0.000	$\mu = 1.0, \lambda = 1.0000$
$\mathbf{x}^{(14)}$	5.923	0.033	0.006	obj = 538.02 (531.18)
$\Delta \mathbf{x}^{(15)}$	0.038	-0.016	-0.003	$\mu = 0.50, \lambda = 1.8$
$\mathbf{x}^{(15)}$	5.992	0.003	0.001	obj = 539.80 (534.10)
$\Delta \mathbf{x}^{(16)}$	-0.003	0.001	0.000	$\mu = 0.50, \lambda = 2.0$
$\mathbf{x}^{(16)}$	5.986	0.006	0.001	obj = 539.64 (534.53)

objective value, including log-barrier terms (in parentheses). Figure 7.8 tracks progress graphically.

Initial point $\mathbf{x}^{(0)}$ is interior and satisfies the only equality constraint $\frac{1}{2}x_1 + x_2 + x_3 = 3$. We begin with barrier multiplier $\mu = 16$. After six moves, the search arrives at point $\mathbf{x}^{(6)}$, which very nearly optimizes barrier objective function

$$\max 90x_1 + 150x_2 + 0x_3 + 16 [\ln(x_1) + \ln(x_2) + \ln(x_3)]$$

Now we begin to reduce μ . In Table 7.3 each reduction updates $\mu \leftarrow \frac{1}{2}\mu$, although μ would normally not be reduced so rapidly. Two steps follow each change to bring the search near the best point for each new μ .

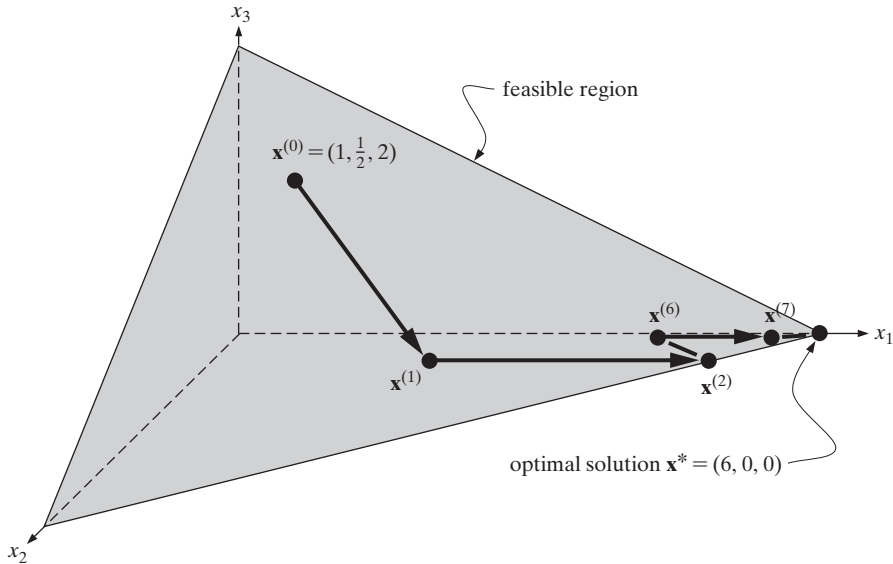


FIGURE 7.8 Barrier Search of Frannie's Firewood Application

Algorithm 7B terminates when μ approaches zero and no further progress is being made. Here we stop after $\mu = \frac{1}{2}$ with $\mathbf{x}^{(16)} = (5.986, 0.006, 0.001)$, which is very near the true LP optimum $\mathbf{x}^* = (6, 0, 0)$.

7.5 PRIMAL-DUAL INTERIOR-POINT SEARCH

This section treats a final family of interior-point search algorithms that take a primal-dual approach building heavily on the optimality theory results of Section 6.7

KKT Optimality Conditions

Specifically we assume instances of primal linear programs in (minimizing) standard form.

$$\begin{aligned} \min \quad & \mathbf{c}\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{7.20}$$

and corresponding duals

$$\begin{aligned} \max \quad & \mathbf{v}\mathbf{b} \\ \text{s.t.} \quad & \mathbf{A}^T\mathbf{v} + \mathbf{w} = \mathbf{c} \\ & \mathbf{v} \text{ URS, } \mathbf{w} \geq \mathbf{0} \end{aligned} \tag{7.21}$$

Here \mathbf{v} is the vector of dual multipliers on the main primal constraints, and \mathbf{w} is a vector of nonnegative slack variables added to convert main dual constraints into equalities.

Combining these model forms with complementary slackness requirements produces the full **Karush-Kuhn-Tucker conditions** for optimality of given primal and dual solutions layed out in principle [6.61](#).

Principle 7.22 Primal and dual solutions $\bar{\mathbf{x}}$ and $(\bar{\mathbf{v}}, \bar{\mathbf{w}})$ are optimal in their respective problems if and only if

$$\mathbf{A}\bar{\mathbf{x}} = \mathbf{b} \quad (\text{Primal Feasibility})$$

$$\bar{\mathbf{x}} \geq \mathbf{0}$$

$$\mathbf{A}^T \bar{\mathbf{v}} + \bar{\mathbf{w}} = \mathbf{c} \quad (\text{Dual Feasibility})$$

$$\bar{\mathbf{w}} \geq \mathbf{0}$$

$$\bar{x}_j \bar{w}_j = 0 \text{ for all } j \quad (\text{Complementary Slackness})$$

Strategy of Primal-Dual Interior-Point Search

Section 6.9 presents a “primal-dual” version of the Simplex method (Algorithm 7B) which pursues a strategy of maintaining feasibility of all dual solutions encountered and seeking a corresponding primal-feasible solution among those satisfying complementary slackness with the current dual. Although it shares the same name, **Primal-Dual Interior-Point Search** takes a significantly different approach.

Principle 7.23 Primal-Dual Interior-Point search begins with and maintains primal and dual solutions that at each iteration are both strictly feasible, then seeks to systematically reduce the violation of complementary slackness between them.

Feasible Move Directions

Like other methods of this chapter, Primal-Dual interior search does visit only strictly feasible primal and dual solutions with $\bar{\mathbf{x}} > \mathbf{0}$ and $\bar{\mathbf{w}} > \mathbf{0}$ so that all are in the interior of their respective feasible sets. Unlike other methods, however, Primal-Dual interior search addresses main primal and dual constraints directly. That is, there is no selecting preferred directions and then projecting them on the main equalities to preserve feasibility. Instead, the requirements enforced on move directions to preserve feasibility in constraints apply the familiar pattern of earlier chapters.

Principle 7.24 For current, strictly feasible solutions to primal (7.20) and dual (7.21), feasible move directions $(\Delta \mathbf{x}, \Delta \mathbf{v}, \Delta \mathbf{w})$ must satisfy

$$\mathbf{A} \Delta \mathbf{x} = \mathbf{0} \text{ and}$$

$$\mathbf{A}^T \Delta \mathbf{v} + \Delta \mathbf{w} = \mathbf{0}$$

Notice that nonnegativity constraints $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{w} \geq \mathbf{0}$ can be disregarded in choosing feasible move directions. Strictly feasible current solutions $\bar{\mathbf{x}} > \mathbf{0}$ and $\bar{\mathbf{w}} > \mathbf{0}$ leave all such constraints inactive.

Management of Complementary Slackness

The challenge with Primal-Dual interior search is handling of the complementary slackness constraints of optimality conditions [7.22]. Unless the current solutions are optimal, there will be a **duality gap**

$$c\bar{\mathbf{x}} - b\bar{\mathbf{v}} = \text{Total Complementary Slackness Violation} = \sum_j \bar{x}_j \bar{w}_j \quad (7.22)$$

Furthermore, with $\bar{\mathbf{x}}$ and $\bar{\mathbf{w}}$ strictly feasible, every term of the violation sum will have $\bar{x}_j \bar{w}_j > 0$.

The approach of Primal-Dual interior search is to set a target $\mu > 0$ for each of these violations, solve for feasible move directions that reduce the disparity between current complementary slackness products and the target, then slowly diminish μ toward 0 where complementary slackness would be achieved. This requires adding conditions

$$\begin{aligned} (\bar{x}_j + \Delta x_j)(\bar{w}_j + \Delta w_j) &= \mu \text{ or} \\ \bar{x}_j \bar{w}_j + \bar{x}_j \Delta w_j + \Delta x_j \bar{w}_j + \Delta x_j \Delta w_j &= \mu \text{ for all } j \end{aligned} \quad (7.23)$$

to requirements [7.24].

Now the challenge is quadratic terms $\Delta x_j \Delta w_j$ in system (7.23). Primal-Dual interior search disregards them in the interest of maintaining a linear system of approximate conditions to solve in determining the next set of move directions.

Principle 7.25 At current strictly feasible solutions $(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \bar{\mathbf{w}})$, Primal-Dual Interior-Point search follows move directions $(\Delta \mathbf{x}, \Delta \mathbf{v}, \Delta \mathbf{w})$ satisfying feasibility requirements of [7.24] together with $\bar{x}_j \bar{w}_j + \bar{x}_j \Delta w_j + \Delta x_j \bar{w}_j = \mu$ on all j , where μ is a target value for complementarity violation decreased slowly toward 0.

Step Size

All that remains in detailing an algorithm is the stepsize λ to be followed at any step. From the nonnegativity constraints on \mathbf{x} and \mathbf{w} set outer limits, and a constant positive boundary prevention $\delta < 1$ adjusts for the need to stay in the interior.

Principle 7.26 At current strictly feasible solutions $(\bar{\mathbf{x}}, \bar{\mathbf{v}}, \bar{\mathbf{w}})$, and move directions $(\Delta \mathbf{x}, \Delta \mathbf{v}, \Delta \mathbf{w})$ satisfying [7.25], Primal-Dual Interior-Point search employs stepsize $\lambda \leftarrow \delta \min\{\lambda_P, \lambda_D\}$, where $\lambda_P = \min\{-\bar{x}_j/\Delta x_j : \bar{x}_j < \mathbf{0}\}$ and $\lambda_D = \min\{-\bar{w}_j/\Delta w_j : \bar{w}_j < \mathbf{0}\}$.

Algorithm 7C collects all these ideas in a formal Primal-Dual interior-point method for solving linear programs.

Solving the Conditions for Move Directions

Given current solutions $\mathbf{x}^{(t)}$, $\mathbf{v}^{(t)}$, and $\mathbf{w}^{(t)}$ at any iteration t , let \mathbf{X}_t , \mathbf{V}_t , and \mathbf{W}_t denote square matrices with their respective solution vectors arrayed along the diagonals,

ALGORITHM 7C: PRIMAL-DUAL INTERIOR-POINT LP SEARCH

Step 0: Initialization. Choose a strictly feasible primal solution $\mathbf{x}^{(0)}$ and a strictly feasible dual solution $(\mathbf{v}^{(0)}, \mathbf{w}^{(0)})$. Then select a value for where the standard target reduction factor is $0 < \rho < 1$; initialize the average complementarity target by computing duality gap $g_0 \leftarrow \mathbf{c}\mathbf{x}^{(0)} - \mathbf{b}\mathbf{v}^{(0)}$ and making $\mu_0 \leftarrow g_0/n$ where n is the dimension of \mathbf{x} . Finally, initialize solution index $t \leftarrow 0$.

Step 1: Optimality. If gap $g_t \leftarrow \mathbf{c}\mathbf{x}^{(t)} - \mathbf{b}\mathbf{v}^{(t)}$ is sufficiently close to zero, stop. Current solutions $\mathbf{x}^{(t)}$, $\mathbf{v}^{(t)}$, and $\mathbf{w}^{(t)}$ are optimal or close to optimal in their respective models. Otherwise, reduce $\mu_{t+1} \leftarrow \rho \cdot \mu_t$, and proceed to Step 2.

Step 2: Move Direction. Compute move directions $\Delta\mathbf{x}^{(t+1)}$, $\Delta\mathbf{v}^{(t+1)}$, and $\Delta\mathbf{w}^{(t+1)}$ by solving the equation system

$$\mathbf{A}\Delta\mathbf{x}^{(t+1)} = \mathbf{0}$$

$$\mathbf{A}^T\Delta\mathbf{v}^{(t+1)} + \Delta\mathbf{w}^{(t+1)} = \mathbf{0}$$

$$x_j^{(t)}\Delta w_j^{(t+1)} + w_j^{(t)}\Delta x_j^{(t+1)} = \mu_{t+1} - x_j^{(t)}w_j^{(t)} \text{ for all } j$$

Step 3: Step Size. Compute stepsize $\lambda \leftarrow \delta \min\{\lambda_p, \lambda_D\}$, where

$$\lambda_p = \min\{-x_j^{(t)}/\Delta x_j^{(t+1)} : \Delta x_j^{(t+1)} < 0\}$$

$$\lambda_D = \min\{-w_j^{(t)}/\Delta w_j^{(t+1)} : \Delta w_j^{(t+1)} < 0\}$$

and $0 < \delta < 1$ is the standard positive boundary prevention factor.

Step 4: Advance. Update solutions

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda\Delta\mathbf{x}^{(t+1)}$$

$$\mathbf{v}^{(t+1)} \leftarrow \mathbf{v}^{(t)} + \lambda\Delta\mathbf{v}^{(t+1)}$$

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \lambda\Delta\mathbf{w}^{(t+1)}$$

Then advance $t \leftarrow t + 1$ and return to Step 1.

and $\mathbf{1}$ be a vector of 1's. Then the conditions of Algorithm 7C Step 2 can be rewritten in full matrix form as follows:

$$(a) \quad \mathbf{A}\Delta\mathbf{x}^{(t+1)} = \mathbf{0}$$

$$(b) \quad \mathbf{A}^T\Delta\mathbf{v}^{(t+1)} + \Delta\mathbf{w}^{(t+1)} = \mathbf{0} \tag{7.24}$$

$$(c) \quad \mathbf{X}_t\Delta\mathbf{w}^{(t+1)} + \mathbf{W}_t\Delta\mathbf{x}^{(t+1)} = \mu_{t+1}\mathbf{1} - \mathbf{X}_t\mathbf{W}_t\mathbf{1}$$

To begin computing a solution, solve (7.24)(c) for $\Delta\mathbf{x}^{(t+1)}$ in terms of $\Delta\mathbf{w}^{(t+1)}$

$$\Delta\mathbf{x}^{(t+1)} \leftarrow -\mathbf{W}_t^{-1}\mathbf{X}_t\Delta\mathbf{w}^{(t+1)} + \mathbf{W}_t^{-1}(\mu_{t+1}\mathbf{1} - \mathbf{X}_t\mathbf{W}_t\mathbf{1}) \tag{7.25}$$

Then, in turn, solve (7.24)(b) for $\Delta\mathbf{w}^{(t+1)}$ in terms of $\Delta\mathbf{v}^{(t+1)}$

$$\Delta\mathbf{w}^{(t+1)} \leftarrow -\mathbf{A}^T\Delta\mathbf{v}^{(t+1)} \tag{7.26}$$

Now substituting (7.26) in conditions (7.24)(c) and multiplying through by $\mathbf{A}\mathbf{W}_t^{-1}$ gives

$$-\mathbf{A}\mathbf{W}_t^{-1}\mathbf{X}_t\mathbf{A}^T\Delta\mathbf{v}^{(t+1)} + \mathbf{A}\mathbf{W}_t^{-1}\mathbf{W}_t\Delta\mathbf{x}^{(t+1)} = \mathbf{A}\mathbf{W}_t^{-1}(\mu_{t+1}\mathbf{1} - \mathbf{X}_t\mathbf{W}_t\mathbf{1}) \quad (7.27)$$

Noting that (7.24)(a) implies

$$\mathbf{A}\mathbf{W}_t^{-1}\mathbf{W}_t\Delta\mathbf{x}^{(t+1)} = \mathbf{A}\Delta\mathbf{x}^{(t+1)} = 0$$

expression (7.27) simplifies to produce

$$-\mathbf{A}\mathbf{W}_t^{-1}\mathbf{X}_t\mathbf{A}^T\Delta\mathbf{v}^{(t+1)} = \mathbf{A}\mathbf{W}_t^{-1}(\mu_{t+1}\mathbf{1} - \mathbf{X}_t\mathbf{W}_t\mathbf{1})$$

Now solving for $\Delta\mathbf{v}^{(t+1)}$ produces the last required expression

$$\Delta\mathbf{v}^{(t+1)} \leftarrow -[\mathbf{A}\mathbf{W}_t^{-1}\mathbf{X}_t\mathbf{A}^T]^{-1}\mathbf{A}\mathbf{W}_t^{-1}(\mu_{t+1}\mathbf{1} - \mathbf{X}_t\mathbf{W}_t\mathbf{1}) \quad (7.28)$$

Principle 7.27 Given a complementarity target $\mu > \mathbf{0}$, along with current solutions $\mathbf{x}^{(t)}$, $\mathbf{v}^{(t)}$, and $w^{(t)}$ at iteration \mathbf{t} of Algorithm 7C (respectively \mathbf{X}_t , \mathbf{V}_t , and \mathbf{W}_t in diagonal matrix form), the next set of move directions of the algorithm can be computed by using (7.28) to compute $\Delta\mathbf{v}^{(t+1)}$, then employing the result in (7.26) to compute a corresponding $\Delta\mathbf{w}^{(t+1)}$, and finally substituting in (7.25) to obtain $\Delta\mathbf{x}^{(t+1)}$.

EXAMPLE 7.14: PRIMAL-DUAL INTERIOR-POINT SEARCH

To illustrate Algorithm 7C, consider the standard-form linear program

$$\begin{aligned} \min \quad & 11x_1 + 5x_2 + 8x_3 + 16x_4 \\ \text{s.t.} \quad & -1x_2 + 2x_3 + 1x_4 = 1 \\ & 2x_1 + 1x_2 + 3x_4 = 7 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

It is easy to check that strictly positive solution $\mathbf{x}^{(0)} = (1, 2, 1, 1)$ is feasible in this primal model.

The corresponding dual with variables v_1 and v_2 on the two main constraints, and slacks w_1, \dots, w_4 is

$$\begin{aligned} \max \quad & 1v_1 + 7v_2 \\ \text{s.t.} \quad & 2v_2 + w_1 = 1 \\ & -1v_1 + 2v_2 + w_2 = 5 \\ & 2v_1 + w_3 = 8 \\ & 1v_1 + 3v_2 + w_4 = 16 \\ & v_1, v_2 \text{ URS, } w_1, w_2, w_3, w_4 \geq 0 \end{aligned}$$

A feasible dual solution is $\mathbf{v}^{(0)} = (3, 4)$ and $\mathbf{w}^{(0)} = (3, 4, 2, 1)$, strictly positive as required on the sign-restricted variables.

The primal has solution value $\mathbf{c}\mathbf{x}^{(0)} = 45$, where \mathbf{c} is the vector of objective function coefficients, and the dual $\mathbf{b}\mathbf{v}^{(0)} = 31$, where \mathbf{b} is the right-hand side. This leaves a

complementarity gap of $g_0 \leftarrow 45 - 31 = 14$, and the algorithm will begin with average complementarity target for the four primal variables fixed $\mu_0 \leftarrow 14/4 = 3.5$, reducing it by factor $\rho = 0.6$ at each iteration.

Construction of the first set of move directions begins with

$$\mathbf{X}_0 = \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \mathbf{V}_0 = \begin{bmatrix} 3 & \\ & 4 \end{bmatrix} \quad \mathbf{W}_0 = \begin{bmatrix} 3 & & & \\ & 4 & & \\ & & 2 & \\ & & & 1 \end{bmatrix}$$

so that

$$\begin{aligned} \mathbf{A}\mathbf{W}_0^{-1}\mathbf{X}_0\mathbf{A}^T &= \begin{bmatrix} 0 & -1 & 2 & 1 \\ 2 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1/3 & & & \\ & 1/4 & & \\ & & 1/2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ -1 & 1 \\ 2 & 0 \\ 1 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 3.5 & 2.5 \\ 2.5 & 1.8333 \end{bmatrix} \end{aligned}$$

$$(\mathbf{A}\mathbf{W}_0^{-1}\mathbf{X}_0\mathbf{A}^T)^{-1} = \begin{bmatrix} 3.5 & 2.5 \\ 2.5 & 1.8333 \end{bmatrix}^{-1} = \begin{bmatrix} 0.3421 & -0.0789 \\ -0.0789 & 0.1105 \end{bmatrix}$$

$$\mu_1\mathbf{1} - \mathbf{X}_0\mathbf{W}_0\mathbf{1} = \begin{bmatrix} 3.5 \cdot 0.6 \\ 3.5 \cdot 0.6 \\ 3.5 \cdot 0.6 \\ 3.5 \cdot 0.6 \end{bmatrix} - \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 3 & & & \\ & 4 & & \\ & & 2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.9 \\ -5.9 \\ 0.1 \\ 1.1 \end{bmatrix}$$

Then

$$\begin{aligned} \Delta\mathbf{v}^{(1)} &= -[\mathbf{A}\mathbf{W}_0^{-1}\mathbf{X}_0\mathbf{A}^T]^{-1}\mathbf{A}\mathbf{W}_0^{-1}(\mu_1\mathbf{1} - \mathbf{X}_0\mathbf{W}_0\mathbf{1}) \\ &= \begin{bmatrix} 0.3421 & -0.0789 \\ -0.0789 & 0.1105 \end{bmatrix} \begin{bmatrix} 0 & -1 & 2 & 1 \\ 2 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1/3 & & & \\ & 1/4 & & \\ & & 1/2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} -0.9 \\ -5.9 \\ 0.1 \\ 1.1 \end{bmatrix} \\ &= \begin{bmatrix} -0.8184 \\ 0.0758 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \Delta\mathbf{w}^{(1)} &= -\mathbf{A}^T\Delta\mathbf{v}^{(1)} \\ &= \begin{bmatrix} 0 & 2 \\ -1 & 1 \\ 2 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -0.8184 \\ 0.0758 \end{bmatrix} = \begin{bmatrix} -0.1516 \\ -0.8942 \\ 1.6368 \\ 0.5911 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \Delta \mathbf{x}^{(1)} &= -\mathbf{W}_0^{-1} \mathbf{X}_0 \Delta \mathbf{w}^{(1)} + \mathbf{W}_0^{-1} (\mu_1 \mathbf{1} - \mathbf{X}_0 \mathbf{W}_0 \mathbf{1}) \\ &= - \begin{bmatrix} 1/3 & & & \\ & 1/4 & & \\ & & 1/2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} -0.1516 \\ -0.8942 \\ 1.6368 \\ 0.5911 \end{bmatrix} \\ &\quad + \begin{bmatrix} 1/3 & & & \\ & 1/4 & & \\ & & 1/2 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} -0.9 \\ -5.9 \\ 0.1 \\ 1.1 \end{bmatrix} = \begin{bmatrix} -0.2495 \\ -1.0279 \\ -0.7684 \\ 0.5089 \end{bmatrix} \end{aligned}$$

Next step size is determined by

$$\lambda_P = \min \{ -x_j^{(t)} / \Delta x_j^{(t+1)} : \Delta x_j^{(t+1)} < 0 \} = \min \left\{ \frac{1}{0.2495}, \frac{2}{1.0279}, \frac{1}{0.7684} \right\} = 1.3014$$

$$\lambda_D = \min \{ -w_j^{(t)} / \Delta w_j^{(t+1)} : \Delta w_j^{(t+1)} < 0 \} = \min \left\{ \frac{3}{0.1516}, \frac{4}{0.8942} \right\} = 4.4732$$

$$\lambda = \delta \min \{ \lambda_P, \lambda_D \} = 0.999 * 1.3014 = 1.3001$$

where $\delta = 0.999$ is the boundary prevention factor

Updated solutions are given by

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x}^{(1)} = (0.6757, 0.6637, 0.0010, 1.6617)$$

$$\mathbf{v}^{(1)} = \mathbf{v}^{(0)} + \lambda \Delta \mathbf{v}^{(1)} = (1.9360, 4.0985)$$

$$\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + \lambda \Delta \mathbf{w}^{(1)} = (2.8029, 2.8375, 4.1280, 1.7684)$$

Note that the updated solutions remain feasible. The primal has solution value $\mathbf{cx}^{(1)} = 37.3453$ and the dual $\mathbf{bv}^{(1)} = 30.6257$, which leaves a new complementary gap of $g_1 \leftarrow 37.3453 - 30.6257 = 6.7196 < 14$.

With the duality gap not yet close to $= 0$, the algorithm proceeds to a new round of moves. Table 7.4 shows details of the next several iterations. Computation is terminated when primal and dual solutions are discovered with gap between them < 1.0 .

TABLE 7.4 Progress of Primal-Dual Interior Algorithm 7C on Example 7.14

$t = 0$	$\mathbf{x}^{(0)} \leftarrow (1.0000, 2.0000, 1.0000, 1.0000)$ $\mathbf{v}^{(0)} \leftarrow (3.0000, 4.0000), \mathbf{w}^{(0)} \leftarrow (3.0000, 4.0000, 2.0000, 1.0000)$ $\Delta \mathbf{x}^{(1)} \leftarrow (-0.2495, -1.0279, -0.7684, 0.5089)$ $\Delta \mathbf{v}^{(1)} \leftarrow (-0.8184, 0.0758), \Delta \mathbf{w}^{(1)} \leftarrow (-0.2495, -0.8942, 1.6368, 0.5911)$	primal value = 45.00 dual value = 31.00 $g_0 = 14.00, \mu_0 = 3.5$ $\lambda \leftarrow 1.3001$
$t = 1$	$\mathbf{x}^{(1)} \leftarrow (0.6757, 0.6637, 0.0010, 1.6617)$ $\mathbf{v}^{(1)} \leftarrow (1.9360, 4.0985), \mathbf{w}^{(1)} \leftarrow (2.8029, 2.8375, 4.1280, 1.7684)$ $\Delta \mathbf{x}^{(2)} \leftarrow (0.1693, 0.3704, 0.3034, -0.2364)$ $\Delta \mathbf{v}^{(2)} \leftarrow (-1.7024, 0.8203), \Delta \mathbf{w}^{(2)} \leftarrow (-1.6406, -2.5227, 3.4048, -0.7568)$	primal value = 37.3453 dual value = 30.6257 $g_1 = 6.7196, \mu_1 = 2.1000$ $\lambda \leftarrow 1.1236$
$t = 2$	$\mathbf{x}^{(2)} \leftarrow (0.8660, 1.0799, 0.3419, 1.3961)$ $\mathbf{v}^{(2)} \leftarrow (0.0231, 5.0203), \mathbf{w}^{(2)} \leftarrow (0.9594, 0.0028, 7.9538, 0.9160)$ $\Delta \mathbf{x}^{(3)} \leftarrow (-0.1703, -0.2018, -0.1913, 0.1808)$ $\Delta \mathbf{v}^{(3)} \leftarrow (0.6466, -0.0511), \Delta \mathbf{w}^{(3)} \leftarrow (0.1023, 0.6978, -1.2932, -0.4932)$	primal value = 39.9974 dual value = 35.1651 $g_2 = 4.8323, \mu_2 = 1.2600$ $\lambda \leftarrow 1.7858$

(continued)

TABLE 7.4 Continued

$t = 3$	$\mathbf{x}^{(3)} \leftarrow (0.5618, 0.7196, 0.0003, 1.7189)$ $\mathbf{v}^{(3)} \leftarrow (1.1778, 4.9289), \mathbf{w}^{(3)} \leftarrow (1.1421, 1.2489, 5.6444, 0.0354)$ $\Delta \mathbf{x}^{(4)} \leftarrow (-0.0146, 0.1272, 0.0799, -0.0327)$ $\Delta \mathbf{v}^{(4)} \leftarrow (-0.6867, 0.1525), \Delta \mathbf{w}^{(4)} \leftarrow (-0.3051, -0.8393, 1.3735, 0.2292)$	primal value = 37.2835 dual value = 35.6804 $g_3 = 1.6031, \mu_3 = 0.7560$ $\lambda \leftarrow 1.4865$
$t = 4$	$\mathbf{x}^{(4)} \leftarrow (0.5041, 0.9087, 0.1192, 1.6704)$ $\mathbf{v}^{(4)} \leftarrow (0.1569, 5.1557), \mathbf{w}^{(4)} \leftarrow (0.6887, 0.0012, 7.6862, 0.3761)$ $\Delta \mathbf{x}^{(5)} \leftarrow (-0.1676, -0.0286, -0.0750, 0.1213)$ $\Delta \mathbf{v}^{(5)} \leftarrow (0.2838, -0.0145), \Delta \mathbf{w}^{(5)} \leftarrow (0.0289, 0.2983, -0.5677, 0.2404)$	primal value = 38.1638 dual value = 36.2466 $g_4 = 1.9172, \mu_4 = 0.4536$ $\lambda \leftarrow 1.5625$
$t = 5$	$\mathbf{x}^{(5)} \leftarrow (0.2782, 0.8639, 0.0020, 1.8599)$ $\mathbf{v}^{(5)} \leftarrow (0.6004, 5.1331), \mathbf{w}^{(5)} \leftarrow (0.7339, 0.4674, 6.7992, 0.0004)$	primal value = 37.1544 dual value = 36.5319 $g_5 = 0.6225, \mu_5 = 0.2722$ $\lambda \leftarrow 1.5552$

7.6 COMPLEXITY OF LINEAR PROGRAMMING SEARCH

Sections 14.1 and 14.2 of Complexity Theory Chapter 14 detail the standard way effort required by various algorithms is reported and compared. A **problem** or model form is taken as an infinite collection of **instances**—specific data sets. A **computational order** (denoted $O(\cdot)$) for any algorithm is a function of the size/length of instance input that bounds the number of computational steps to complete its solution. The bound must be “worst-case” in the sense that it applies to any instance, however perverse.

This section briefly reviews the long trail of research on how the efficiency of algorithms for the Linear Program (LP) problem form can/should be analyzed in this way.

Length of Input for LP Instances

The length of the input for any instance of an optimization model form is described by such global parameters as $n \triangleq$ the number of variables and $m \triangleq$ the number of main constraints, plus the total number of digits required to input all its constant parameters. This leads to the following result for a linear program in standard form (7.20).

Principle 7.28 The length of a standard form instance of linear programming over n variables, m main constraints, integer costs \mathbf{c} , integer constraint matrix \mathbf{A} , and integer right-hand sides \mathbf{b} can be expressed as

$$L \triangleq n \cdot m + \sum_j [\log(|c_j| + 1)] + \sum_{ij} [\log(|a_{ij}| + 1)] + \sum_i [\log(|b_i| + 1)]$$

Notice that expressions $[\log(\text{parameter})] + 1$ capture the number of digits in different model parameters.

Complexity of Simplex Algorithms for LP

Details obviously vary, but the effort *per iteration* of any of the simplex algorithms of Chapters 5 and 6 can be shown to be bounded by a low order polynomial function (low constant power) of the instance size L in principle [7.28]. Furthermore, simplex algorithms have proved highly effective in solving large instances of linear programming for many decades because the number of iterations required is usually proportional to a low-order polynomial in the number of main constraints m . Still, bounding the worst-case number of simplex iterations for LP instances has proved an enigma.

Consider LPs of the following form first exhibited by Klee and Minty (1972).

$$\begin{array}{ll} \max & x_n \\ \text{s.t.} & \alpha \leq x_1 \leq 1 \\ & \alpha \cdot x_{j-1} \leq x_j \leq \alpha \cdot x_{j-1} \quad j = 1, \dots, n \end{array} \quad (7.29)$$

where constant $\alpha \in (0, 1/2)$

Figure 7.9 illustrates for $n = 2$ and $\alpha = 1/4$. Constraints form a slanted variant of a hypercube in dimension n with 2^n extreme-point solutions. Starting from solution $\mathbf{x}^{(1)}$, deviations from an actual hypercube make each solution $\mathbf{x}^{(t+1)}$ adjacent to predecessor $\mathbf{x}^{(t)}$ with better objective value $x_n^{(t+1)} > x_n^{(t)}$. A simplex algorithm might very well proceed from extreme-point to adjacent extreme-point, improving the objective value until all 2^n of them have been visited. The consequence is an important finding about worst-case performance of simplex algorithms.

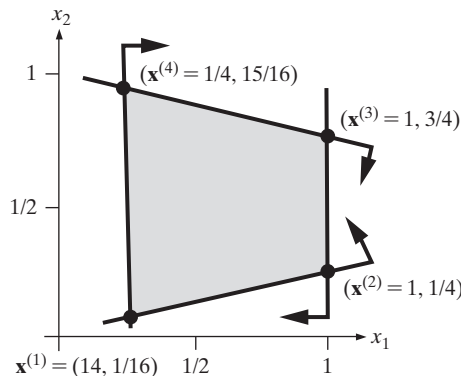


FIGURE 7.9 Klee-Minty Perverse LP Instances for Simplex Methods

Principle 7.29 Instances of linear programming exist for which simplex algorithms require an exponential number of iterations as a function of input length [7.28].

Complexity of Interior-Point Algorithms for LP

Although the typical performance of simplex algorithms is usually quite acceptable, the anomalies of principle 7.29 are troubling for the wide body of researchers who believe the accepted standard 14.5 for establishing a problem form is tractable. There must exist an algorithm that runs in time polynomial in the number of variables n and the length of the input L on even the worst instance. The investigation of interior-point methods actually began as a quest for alternatives to simplex ideas that could meet the polynomial-time standard.

Soviet mathematician Leonid Khachiyan provided the first answer in 1979 with his **ellipsoid** technique that proved formally polynomial with a bound of $O((mn^3 + n^4)L)$, but was quickly shown to be impractical as an algorithm for large linear programming applications. Development of more practically useful algorithms began in the 1980s with N. Karmarkar's **projective transformation** method, which bears some similarity to the Log-Barrier algorithms of Section 7.4. With some improvements through the years, it has achieved $O(\sqrt{n}L)$ bound on the number of iterations. Variations on the primal-dual methods of Section 7.5 have at least matched that limit.

Principle 7.30 A number of interior-point methods for linear programming, including refined versions of the Primal-Dual Interior-Point Algorithm 7C developed in Section 7.5 can be shown to achieve the polynomial standard for formal efficiency by requiring at most $O(\sqrt{n}L)$ iterations, each polynomial in the length of the input.

Details of all these methods, including the proofs of computational orders, are well beyond the scope of this book. Interested readers are referred to references at the end of this chapter.

EXERCISES

7-1 Consider the linear program

$$\begin{array}{ll} \max & 2w_1 + 3w_2 \\ \text{s.t.} & 4w_1 + 3w_2 \leq 12 \\ & w_2 \leq 2 \\ & w_1, w_2 \geq 0 \end{array}$$

- ✓ (a) Solve the problem graphically.
- ✓ (b) Determine the direction $\Delta \mathbf{w}$ of most rapid improvement in the objective function at any solution \mathbf{w} .
- (c) Explain why the direction of part (b) is feasible at any interior point solution to the model.
- ✓ (d) Show that $\mathbf{w}^{(0)} = (1, 1)$ is an interior point solution.

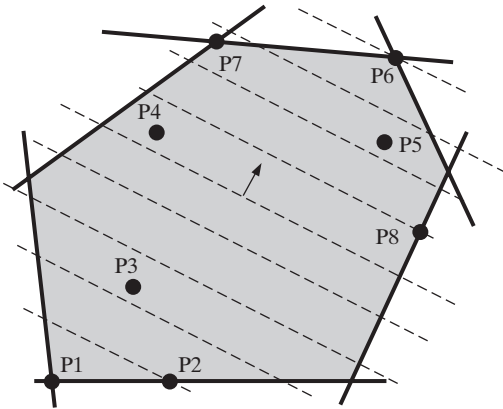
- ✓ (e) Determine the maximum step λ_{\max} from the point $\mathbf{w}^{(0)}$ that preserves feasibility in the direction of part (b).
- ✓ (f) Plot the move of part (e) and the resulting new point $\mathbf{w}^{(1)}$ in the graph of part (a).
- (g) Explain why it is easier to find a good move direction at $\mathbf{w}^{(0)}$ than at $\mathbf{w}^{(1)}$.

7-2 Do Exercise 7-1 for the LP

$$\begin{array}{ll} \min & 9w_1 + 1w_2 \\ \text{s.t.} & 3w_1 + 6w_2 \geq 12 \\ & 6w_1 + 3w_2 \geq 12 \\ & w_1, w_2 \geq 0 \end{array}$$

and point $\mathbf{w}^{(0)} = (3, 1)$.

7-3 The following plot shows several feasible points in a linear program and contours of its objective function.



Determine whether each of the following sequences of solutions could have been one followed by an interior point algorithm applied to the corresponding standard-form LP.

- ✓ (a) P1,P5,P6
- (b) P3,P4,P6
- ✓ (c) P3,P8,P6
- (d) P4,P3,P6
- ✓ (e) P3,P4,P5,P6
- (f) P5,P4,P6

7-4 Determine whether each of the following is an interior point solution to the standard-form LP constraints

$$\begin{aligned} 4x_1 + 1x_3 &= 13 \\ 5x_1 + 5x_2 &= 15 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

- ✓ (a) $\mathbf{x} = (3, 0, 1)$
- (b) $\mathbf{x} = (2, 1, 5)$
- ✓ (c) $\mathbf{x} = (1, 2, 9)$
- (d) $\mathbf{x} = (5, 1, 1)$
- ✓ (e) $\mathbf{x} = (2, 2, 1)$
- (f) $\mathbf{x} = (0, 3, 13)$

7-5 Write all conditions that a feasible direction $\Delta \mathbf{w}$ must satisfy at any interior point solution to each of the following standard-form systems of LP constraints.

- ✓ (a) $2w_1 + 3w_2 - 3w_3 = 5$
 $4w_1 - 1w_2 + 1w_3 = 3$
 $w_1, w_2, w_3 \geq 0$

(b) $11w_1 + 2w_2 - 1w_3 = 8$
 $2w_1 - 7w_2 + 4w_3 = -7$
 $w_1, w_2, w_3 \geq 0$

7-6 Table 7.4 shows several constraint matrices \mathbf{A} (or \mathbf{A}_i) of standard-form LPs and the corresponding projection matrices \mathbf{P} (or \mathbf{P}_i). Use these results to compute the feasible direction for the specified equality constraints that is nearest to the given direction \mathbf{d} , and verify that the result satisfies feasible direction conditions at any interior point solution.

- ✓ (a) $x_1 + 2x_2 + x_3 = 4$ and $\mathbf{d} = (3, -6, 3)$
 $-2x_1 + x_2 = -1$
- (b) $3x_1 + x_2 + 4x_3 = 4$ and $\mathbf{d} = (2, 1, -7)$
 $x_2 - 2x_3 = 1$

7-7 Consider the standard-form LP

$$\begin{aligned} \min \quad & 14z_1 + 3z_2 + 5z_3 \\ \text{s.t.} \quad & 2z_1 - z_3 = 1 \\ & z_1 + z_2 = 1 \\ & z_1, z_2, z_3 \geq 0 \end{aligned}$$

- ✓ (a) Determine the direction of most rapid objective function improvement at any solution \mathbf{z} .
- ✓ (b) Compute the projection matrix \mathbf{P} for the main equality constraints.
- ✓ (c) Apply your \mathbf{P} to project the direction of (a).
- ✓ (d) Verify that the result of part (c) is improving and feasible at any interior point solution.
- (e) Describe the sense in which the direction of part (c) is good for improving search at any interior point solution.

7-8 Do Exercise 7-7 for

$$\begin{aligned} \max \quad & 5z_1 - 2z_2 + 3z_3 \\ \text{s.t.} \quad & z_1 + z_3 = 4 \\ & 2z_2 = 12 \\ & z_1, z_2, z_3 \geq 0 \end{aligned}$$

7-9 An interior point search using scaling has reached current solution $\mathbf{x}^{(7)} = (2, 5, 1, 9)$. Compute the affine scaled \mathbf{y} that would correspond to each of the following \mathbf{x} 's.

- ✓ (a) $(1, 1, 1, 1)$
- (b) $(2, 1, 4, 3)$
- ✓ (c) $(3, 5, 1, 6)$
- (d) $(3, 5, 1, 7)$

TABLE 7.4 Projection Matrices

A or A _i			P or P _i		
1	-1	2	0.3333	-0.3333	-0.3333
0	1	-1	-0.3333	0.3333	0.3333
			-0.3333	0.3333	0.3333
1	10	3	0.2540	0.1016	-0.4232
-2	5	0	0.1016	0.0406	-0.1693
			-0.4232	-0.1693	0.7054
1	2	1	0.0333	0.0667	-0.1667
-2	1	0	0.0667	0.1333	-0.3333
			-0.1667	-0.3333	0.8333
3	1	4	0.0816	-0.2449	-0.1224
0	1	-2	-0.2449	0.7347	0.3673
			-0.1224	0.3673	0.1837
12	-3	4	0.0819	-0.0755	-0.1132
0	3	-2	-0.0755	0.3019	0.4528
			-0.1132	0.4528	0.6793
4	-3	2	0.0533	-0.0710	-0.2130
0	3	-1	-0.0710	0.0947	0.2840
			-0.2130	0.2840	0.8521
-1	2	0	0.6667	0.3333	0.3333
1	-1	-1	0.3333	0.1667	0.1667
			0.3333	0.1667	0.1667

7-10 Do Exercise 7-9 taking the listed vectors as scaled **y**'s and computing the corresponding **x**.

7-11 Consider the standard-form LP

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 + 5x_3 \\ \text{s.t.} \quad & 2x_1 + 5x_2 + 3x_3 = 12 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

with current interior point solution $\mathbf{x}^{(3)} = (2, 1, 1)$.

- ✓ (a) Sketch the feasible space in a diagram like Figure 7.4(a) and identify both the current solution and an optimal extreme point.
- ✓ (b) Sketch the corresponding affine-scaled feasible space showing scaled equivalents of all points in part (a).
- ✓ (c) Derive the associated affine-scaled standard form [7.10].

7-12 Do Exercise 7-11 using the LP

$$\begin{aligned} \max \quad & 6x_1 + 1x_2 + 2x_3 \\ \text{s.t.} \quad & x_1 + x_2 + 5x_3 = 18 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

and current solution $\mathbf{x}^{(3)} = (6, 7, 1)$.

7-13 Return to the LP of Exercise 7-11.

- ✓ (a) Compute in both **x** and **y** space the next move direction that would be pursued by affine scaling Algorithm 7A.
- ✓ (b) Verify that the $\Delta \mathbf{x}$ of part (a) is both improving and feasible.
- ✓ (c) Compute the step size λ that Algorithm 7A would apply to your move directions in part (a).
- ✓ (d) Plot the move of parts (a) and (c) in both original *x*-space and scaled *y*-space.

7-14 Do Exercise 7-13 on the LP of Exercise 7-12.

7-15 Consider the standard-form LP

$$\begin{aligned} \min \quad & 10x_1 + 1x_2 \\ \text{s.t.} \quad & x_1 - x_2 + 2x_3 = 3 \\ & x_2 - x_3 = 2 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- ✓ (a) Show that $\mathbf{x}^{(0)} = (4, 3, 1)$ is an appropriate point to start affine scaling Algorithm 7A.
- ✓ (b) Derive the associated scaled standard form corresponding to solution $\mathbf{x}^{(0)}$.
- ✓ (c) Compute the move direction $\Delta \mathbf{x}$ that would be pursued from $\mathbf{x}^{(0)}$ by Algorithm 7A (refer to Table 7.4 for projection matrices).
- ✓ (d) Show that your direction is improving and feasible at $\mathbf{x}^{(0)}$.
- ✓ (e) Compute the step size λ that Algorithm 7A would apply to your direction, and determine the new point $\mathbf{x}^{(1)}$ that results.

7-16 Do Exercise 7-15 using the LP

$$\begin{aligned} \max \quad & 6x_1 + 8x_2 + 10x_3 \\ \text{s.t.} \quad & 9x_1 - 2x_2 + 4x_3 = 6 \\ & 2x_2 - 2x_3 = -1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

and initial solution $\mathbf{x}^{(0)} = (\frac{2}{9}, 1, \frac{3}{2})$.

7-17 Suppose that Affine Scaling Algorithm 7A has reached current solution $\mathbf{x}^{(11)} = (3, 1, 9)$. Determine whether each of the following next move directions would cause the algorithm to stop and why. If not, compute $\mathbf{x}^{(12)}$.

- ✓ (a) $\Delta \mathbf{x} = (6, 2, 2)$
- (b) $\Delta \mathbf{x} = (0, 4, -9)$
- ✓ (c) $\Delta \mathbf{x} = (6, -6, 0)$
- (d) $\Delta \mathbf{x} = (0, -2, 0)$

7-18 Consider solving the following standard-form primal LP by Affine Scaling Algorithm 7A starting from solution $\mathbf{x} = (2, 3, 2, 3, 1/3)$.

	x_1	x_2	x_3	x_4	x_5	
min \mathbf{c}	5	4	3	2	16	\mathbf{b}
s.t.	2	0	1	0	6	8
	0	1	1	2	3	12

- (a) Verify that the given solution is an appropriate point at which to start Algorithm 7A.
- (b) Derive the scaled standard form in terms of variables \mathbf{y} that comes from the given starting \mathbf{x} . Also explain why it might be more convenient to work with this form than the original standard form.
- (c) Write an expression for the move direction \mathbf{x} in x -space to be followed on the first solution update of the algorithm. You need not explicitly do projections, just denote them symbolically. For example, to show that you want to project direction

$$\mathbf{d} = (1, 2) \text{ on matrix } \mathbf{A} = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix},$$

$$\text{just write } \Delta \mathbf{x} = \text{proj} \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Also, briefly justify the elements of this expression, that is, (i) what does it start with, (ii) how does it utilize results, and (iii) why is the final form what it is?

7-19 Consider the standard-form LP

$$\begin{aligned} \max \quad & 13w_1 - 2w_2 + w_3 \\ \text{s.t.} \quad & 3w_1 + 6w_2 + 4w_3 = 12 \\ & w_1, w_2, w_3 \geq 0 \end{aligned}$$

- ✓ (a) Sketch the feasible space in a plot like Figure 7.6, identify an optimal extreme point, and show $\mathbf{w}^{(1)} = (1.4, 0.7, 0.9)$ and $\mathbf{w}^{(2)} = (0.01, 0.01, 2.9775)$.
- ✓ (b) Form the corresponding log barrier problem with multiplier $\mu > 0$.
- ✓ (c) Using $\mu = 10$, evaluate the original and log barrier objective functions at $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$. Then comment on the effect of barrier terms at points far from the boundary versus ones near the boundary.

- ✓ (d) Use the class optimization software to solve log barrier form (b) with multipliers $\mu = 100, 10, \text{ and } 1$.
- (e) How does the trajectory of optimal solutions to part (b) evolve as $\mu \rightarrow 0$?

7-20 Do Exercise 7-19 for the LP

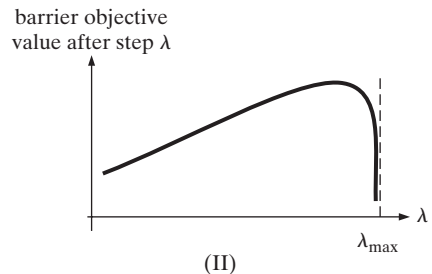
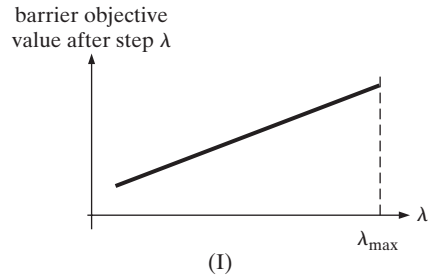
$$\begin{aligned} \min \quad & 2w_1 + 5w_2 - w_3 \\ \text{s.t.} \quad & w_1 + 6w_2 + 2w_3 = 18 \\ & w_1, w_2, w_3 \geq 0 \end{aligned}$$

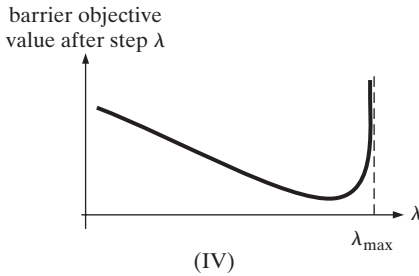
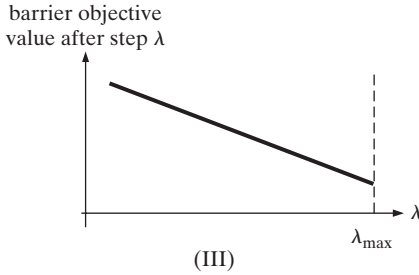
and points $\mathbf{w}^{(1)} = (8, 1, 2)$, $\mathbf{w}^{(2)} = (0.01, 0.02, 8.935)$.

7-21 Determine whether each of the following could be the first four barrier multiplier values μ used in the outer loop of a barrier search Algorithm 7B.

- ✓ (a) 100,80,64,51.2
- (b) 100,200,100,800
- ✓ (c) 100,500,1000,2000
- (d) 600,300,150,75

7-22 Assume that barrier Algorithm 7B has computed an appropriate move direction $\Delta \mathbf{x}$ for its standard-form LP and a maximum feasible step size λ_{\max} . For each of the original objective functions below, determine which of the following curves best depicts how the corresponding barrier objective function will vary with $\lambda \in [0, \lambda_{\max}]$.





- ✓ (a) $\max 34x_1 - 19x_2 - 23x_3 + 4x_4$
- ✓ (b) $\min 44x_1 + 15x_2 + 1x_3 + 9x_4$

7-23 Consider the LP

$$\begin{aligned} \min \quad & 4x_1 - x_2 + 2x_3 \\ \text{s.t.} \quad & 4x_1 - 3x_2 + 2x_3 = 13 \\ & 3x_2 - x_3 = 1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- ✓ (a) Show that $\mathbf{x}^{(0)} = (3, 1, 2)$ is an appropriate point to start barrier Algorithm 7B.
- ✓ (b) Form the corresponding log barrier problem with multiplier $\mu = 10$.
- ✓ (c) Compute the move direction $\Delta \mathbf{x}$ that could be pursued from $\mathbf{x}^{(0)}$ by barrier Algorithm 7B. (See Table 7.4 for the projection matrix required.)
- ✓ (d) Verify that your direction of part (c) is improving and feasible in the barrier model of part (b) at $\mathbf{x}^{(0)}$.
- ✓ (e) Determine the maximum step λ_{\max} from $\mathbf{x}^{(0)}$ that preserves feasibility in your direction of part (c) and the λ to be employed.
- ✓ (f) Will the barrier objective function first increase, then decrease, or first decrease, then increase, as the step λ applied to your direction of part (c) grows from 0 to λ_{\max} ? Explain.

(g) The next time that multiplier μ is changed, should it increase or decrease? Explain.

7-24 Do Exercise 7-23 for the LP

$$\begin{aligned} \max \quad & -x_1 + 3x_2 + 8x_3 \\ \text{s.t.} \quad & x_1 + 2x_2 + x_3 = 14 \\ & 2x_1 + x_2 = 11 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

at $\mathbf{x}^{(0)} = (5, 1, 7)$

7-25 Return to the LP and $\mathbf{x}^{(0)}$ of Exercise 7-15.

- ✓ (a) Show that $\mathbf{x}^{(0)} = (4, 3, 1)$ is an appropriate point to start barrier Algorithm 7B.
- ✓ (b) Compute the move direction $\Delta \mathbf{x}$ that would be pursued from $\mathbf{x}^{(0)}$ by Algorithm 7B with $\mu = 10$ (refer to Table 7.4 for projection matrices).
- ✓ (c) Show that your direction is improving and feasible at $\mathbf{x}^{(0)}$.
- ✓ (d) Compute the maximum step size λ_{\max} that can be applied to your direction of part (b) at $\mathbf{x}^{(0)}$ without losing feasibility.
- ✓ (e) Compute the step size λ that Algorithm 7B would apply to your direction, and determine the new point $\mathbf{x}^{(1)}$ that results.

7-26 Do Exercise 7-25 for the LP and $\mathbf{x}^{(0)}$ of Exercise 7-16.

7-27 Return to the LP of Exercise 7-18 and consider solving by Primal-Dual Interior Algorithm 7C.

- ✓ (a) Derive the corresponding dual formulation over variables \mathbf{v} and place it in standard form by adding slack variables \mathbf{s} as needed.
- ✓ (b) Show that Algorithm 7C could appropriately start with solutions $\mathbf{x} = (2, 3, 2, 3, 1/3)$, $\mathbf{v} = (1/2, 1/2)$, and the corresponding values of \mathbf{s} in your dual standard form of (a).
- ✓ (c) State all relevant complementary slackness conditions between primal and dual.
- ✓ (d) Demonstrate that the primal solution value and the dual solution value of (a) are separated by a duality gap equal to the total complementary slackness violation.
- ✓ (e) Now suppose we want to derive move directions to change primal and dual solutions in a way that will move each complementary slackness violation closer to $\mu = 5$. Write, but do not solve the system

of equation that must be solved to compute values of $\Delta \mathbf{x}$, $\Delta \mathbf{v}$, and $\Delta \mathbf{s}$. Those 3 directions should be left as symbols, but all other elements of the equations should be detailed numerically for the given example and starting points.

- ✔ (f) Build upon your analysis of (e) to conduct up to 3 iterations Primal-Dual Interior Point Algorithm 7C (fewer if optimality is reached earlier) to move toward an optimal solution. At each step show the parameter μ , change directions for $\Delta \mathbf{x}$, $\Delta \mathbf{v}$, and $\Delta \mathbf{s}$, step size λ , and new solutions, as well as the complementary-slackness/duality gap between primal and dual.

7-28 Do Exercise 7-27 on the LP

	x_1	x_2	x_3	x_4	x_5	
min	14	30	11	9	10	RHS
s.t.	2	-1	0	3	-2	10
	1	5	2	-1	1	15

with starting solutions $\mathbf{x} = (2, 1, 5, 3, 1)$ and $\mathbf{v} = (2, 5)$

7-29 Return to the standard form LP instance of Exercise 7-18.

- ✔ (a) Explain why this LP model is an **instance of problem (LP)** defined as

$$\begin{aligned} \min \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j = b_i \quad i = 1, \dots, m \\ & x_j \geq 0, j = 1, \dots, n \end{aligned}$$
- ✔ (b) Following Section 7.6 and 14.1, derive a binary encoding of the instance using alphabet symbols $\{0, 1, \dots, 9, -, /, \$\}$, with ‘-’ the minus sign, ‘\$’ separating individual constants, and ‘/’ separating words

of the instance. Also compute the length of this input for the instance.

- ✔ (c) Explain why the lengths of the encodings of parameters like costs and constraint coefficients are best computed as the logs (rounded up) of their magnitudes, rather than the magnitudes themselves.
- ✔ (d) Establish that your encoding of (a) has length proportional to the number of variables n , the number of main constraints m , and the logs (rounded up to integers) of objective coefficients c_j , main constraint parameters a_{ij} , and right-hand sides b_i .

7-30 Repeat Exercise 7–29 for the LP instance of (7.29).

7-31 Consider the 2-variable LP instance in Figure 7.9 and equation (7.29).

- ✔ (a) State the instance in the (LP) standard form format above, fixing $\alpha = 1/4$.
- ✔ (b) Construct the basic solution corresponding to $\mathbf{x}^{(1)}$ in the figure.
- ✔ (c) Show that movement to figure solution $\mathbf{x}^{(2)}$ can be understood as the simplex direction introducing the slack variable on constraint $x_1 \leq 1$ and that it is both feasible and improving in your formulation of (a).
- (d) Show how similar simplex moves lead to figure points $\mathbf{x}^{(3)}$ then $\mathbf{x}^{(4)}$.
- (e) Explain how this suggests that simplex can take exponentially many iterations to compute an optimum for simple instances like (7.29).
- (f) How can this exponential behavior on specific instances be reconciled with the generally excellent performance of the simplex algorithm on most LPs?

REFERENCES

Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.

Bertsimas, Dimitris and John N. Tsitkalis (1997), *Introduction to Linear Optimization*. Athena Scientific, Nashua, New Hampshire.

Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009), *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.

Martin, R. Kipp (1999), *Large Scale Linear and Integer Optimization*, Kluwer Academic, Boston, Massachusetts.

This page intentionally left blank

Multiobjective Optimization and Goal Programming

Most of the methods of this book address optimization models with a **single objective** function—a lone criterion to be maximized or minimized. Although practical problems almost always involve more than one measure of solution merit, many can be modeled quite satisfactorily with a single cost or profit objective. Other criteria are either represented as constraints or weighted in a composite objective function to produce a model tractable enough for productive analysis.

Other applications—especially those in the public sector—must simply be treated as **multiobjective**. When goals cannot be reduced to a common scale of cost or benefit, trade-offs have to be addressed. Only a model with multiple objective functions is satisfactory, even though analysis will almost certainly become more challenging.

This chapter introduces the key notions and approaches available when such multiobjective analysis is required. Emphasis is on **efficient solutions**, which are optimal in a certain multiobjective sense, and **goal programming**, which is the most commonly employed technique for dealing with multiobjective settings.

8.1 MULTIOBJECTIVE OPTIMIZATION MODELS

As usual, we begin our investigation of multiobjective optimization with some examples. In this section we formulate three cases illustrating the broad range of applications requiring multiobjective analysis. All are based on real contexts documented in published reports.

APPLICATION 8.1: BANK THREE INVESTMENT

Every investor must trade off return versus risk in deciding how to allocate his or her available funds. The opportunities that promise the greatest profits are almost always the ones that present the most serious risks.

Commercial banks must be especially careful in balancing return and risk because legal and ethical obligations demand that they avoid undo hazards. Yet their goal as a business enterprise is to maximize profit. This dilemma leads naturally to multiobjective optimization of investment that includes both profit and risk criteria.

Our investment example¹ adopts this multiobjective approach to a fictitious Bank Three. Bank Three has a modest \$20 million capital, with \$150 million in demand deposits (checking accounts) and \$80 million in time deposits (savings accounts and certificates of deposit).

Table 8.1 displays the categories among which the bank must divide its capital and deposited funds. Rates of return are also provided for each category together with other information related to risk.

TABLE 8.1 Bank Three Investment Opportunities

Investment Category, j	Return Rate (%)	Liquid Part (%)	Required Capital (%)	Risk Asset?
1: Cash	0.0	100.0	0.0	No
2: Short term	4.0	99.5	0.5	No
3: Government: 1 to 5 years	4.5	96.0	4.0	No
4: Government: 5 to 10 years	5.5	90.0	5.0	No
5: Government: over 10 years	7.0	85.0	7.5	No
6: Installment loans	10.5	0.0	10.0	Yes
7: Mortgage loans	8.5	0.0	10.0	Yes
8: Commercial loans	9.2	0.0	10.0	Yes

We model Bank Three’s investment decisions with a decision variable for each category of investment in Table 8.1:

$$x_j \triangleq \text{amount invested in category } j \text{ (\$ million)} \quad j = 1, \dots, 8$$

Bank Three Example Objectives

The first goal of any private business is to maximize profit. Using rates of return from Table 8.1, this produces objective function

$$\begin{aligned} \max \quad & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 \quad (\text{profit}) \\ & + 0.105x_6 + 0.085x_7 + 0.092x_8 \end{aligned}$$

It is less clear how to quantify investment risk. We employ two common ratio measures.

One is the *capital-adequacy ratio*, expressed as the ratio of required capital for bank solvency to actual capital. A low value indicates minimum risk. The “required capital” rates of Table 8.1 approximate U.S. government formulas used to compute this ratio, and Bank Three’s present capital is \$20 million. Thus we will express a second objective as

$$\begin{aligned} \min \quad & \frac{1}{20} (0.005x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5 \quad (\text{capital-adequacy}) \\ & + 0.100x_6 + 0.100x_7 + 0.100x_8) \end{aligned}$$

Another measure of risk focuses on illiquid *risk assets*. A low risk asset/capital ratio indicates a financially secure institution. For our example, this third measure of success is expressed as

$$\min \quad \frac{1}{20} (x_6 + x_7 + x_8) \quad (\text{risk-asset})$$

¹Based on J. L. Eatman and C. W. Sealey, Jr. (1979), “A Multiobjective Linear Programming Model for Commercial Bank Balance Sheet Management,” *Journal of Bank Research*, 9, 227–236.

Bank Three Example Model

To complete a model of Bank Three's investment plans, we must describe the relevant constraints. Our example will assume five types:

1. Investments must sum to the available capital and deposit funds.
2. Cash reserves must be at least 14% of demand deposits plus 4% of time deposits.
3. The portion of investments considered liquid (see Table 8.1) should be at least 47% of demand deposits plus 36% of time deposits.
4. At least 5% of funds should be invested in each of the eight categories, for diversity.
5. At least 30% of funds should be invested in commercial loans, to maintain the bank's community status.

Combining the 3 objective functions above with these 5 systems of constraints completes a multiobjective linear programming model of Bank Three's investment problem:

$$\begin{array}{ll}
 \max & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 & (\text{profit}) \\
 & + 0.105x_6 + 0.085x_7 + 0.092x_8 \\
 \min & \frac{1}{20}(0.005x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5 & (\text{capital-adequacy}) \\
 & + 0.100x_6 + 0.100x_7 + 0.100x_8) \\
 \min & \frac{1}{20}(x_6 + x_7 + x_8) & (\text{risk-asset}) \\
 \text{s.t.} & x_1 + \cdots + x_8 = (20 + 150 + 80) & (\text{invest all}) \\
 & x_1 \geq 0.14(150) + 0.04(80) & (\text{cash reserve}) \\
 & 1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 & (\text{liquidity}) \\
 & + 0.850x_5 \geq 0.47(150) + 0.36(80) \\
 & x_j \geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 & (\text{diversification}) \\
 & x_8 \geq 0.30(20 + 150 + 80) & (\text{commercial}) \\
 & x_1, \dots, x_8 \geq 0
 \end{array} \tag{8.1}$$

APPLICATION 8.2: DYNAMOMETER RING DESIGN

Multiobjective optimization also occurs in the engineering design of almost any product or service. Competing performance measures must be balanced to choose a best design.

We will illustrate with the design of a simple mechanical part, an octagonal ring used in dynamometer instrumentation of machine tools.² Figure 8.1 depicts the critical design variables:

- $w \triangleq$ width of the ring (in centimeters)
- $t \triangleq$ thickness of the outer surface (in centimeters)
- $r \triangleq$ radius of the two openings (in centimeters)

It also shows upper and lower limits allowed for the three dimensions.

²Based on N. Singh and S. K. Agarwal (1983), "Optimum Design of an Extended Octagonal Ring by Goal Programming," *International Journal of Production Research*, 21, 891–898.

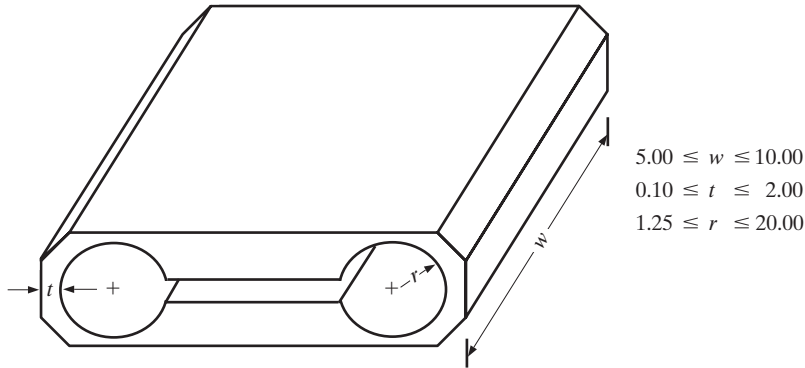


FIGURE 8.1 Dynamometer Ring Design Example

Dynamometer Ring Design Model

To model this machine part optimization, we must characterize performance in terms of decision variables w , t , and r . One consideration is sensitivity. We would like the instrument to be as sensitive as possible. Analysis of the relevant strains and deflections shows that sensitivity can be represented as

$$\max \frac{0.7r}{Ewt^2} \quad (\text{sensitivity})$$

where $E = 2.1 \times 10^6$, Young's modulus of elasticity for the ring material.

Another measure of effectiveness is rigidity. Increased rigidity produces greater accuracy. Again considering critical strains and deflections, rigidity can be modeled as

$$\max \frac{Ewt^3}{r^3} \quad (\text{rigidity})$$

Combining these two objectives with upper and lower bounds on the variables yields a multiobjective optimization model of our ring design problem:

$$\begin{aligned} \max \quad & \frac{0.7r}{Ewt^2} && (\text{sensitivity}) \\ \max \quad & \frac{Ewt^3}{r^3} && (\text{rigidity}) \\ & 5.0 \leq w \leq 10.0 \\ & 0.1 \leq t \leq 2.0 \\ & 1.25 \leq r \leq 20.0 \end{aligned} \tag{8.2}$$

Notice that this model is a multiobjective nonlinear program. It can, however, be handled by linear programming if we change variables to $\ln(w)$, $\ln(t)$, and $\ln(r)$ and maximize the logarithms of the two objective functions (see Section 17.10).

APPLICATION 8.3: HAZARDOUS WASTE DISPOSAL

Many, perhaps most, government planning problems are multiobjective because they involve competing interests and objectives that are difficult to quantify in

monetary terms. We illustrate with the planning of disposal sites for dangerous nuclear wastes.³

Figure 8.2 depicts the source points and potential disposal sites we assume in our fictitious version of the problem. Waste material is generated at the seven sources as nuclear power plants and other reactors use up fuel. The goal of the study is to choose 2 of the 3 possible disposal sites to receive the waste.

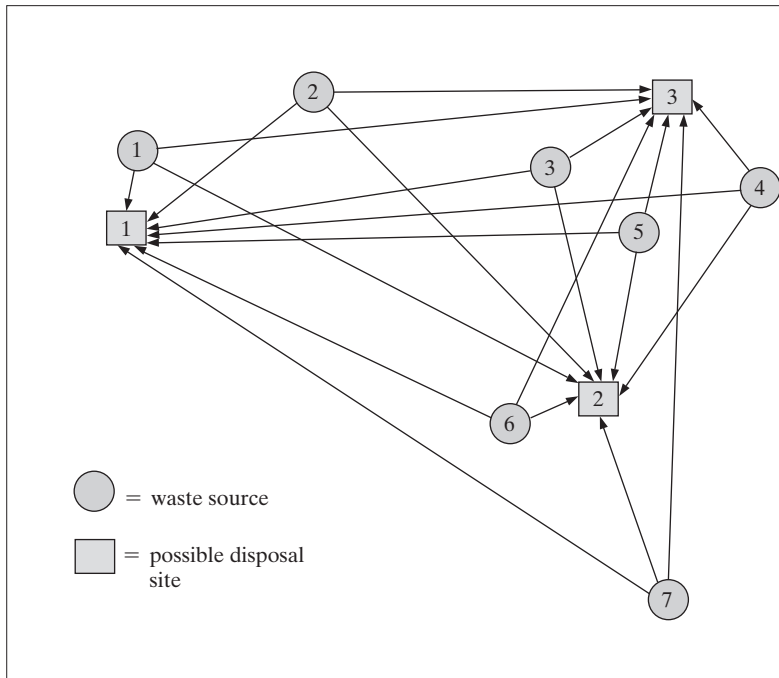


FIGURE 8.2 Hazardous Waste Example Sources and Disposal Sites

When the system is ready, wastes will be trucked from sources to disposal sites over the public highway system. Conflicting objectives arise in routing materials to the sites. Costs will be reduced, and in-transit time of wastes minimized if we ship along shortest routes from source to disposal site. But population densities must also be considered. Routes through densely populated areas threaten more people and incur greater risk of traffic mishaps.

Table 8.2 shows parameters

$s_i \triangleq$ amount of waste expected to be produced at source i

$d_{i,j,k} \triangleq$ distance from source i to site j along route k

$p_{i,j,k} \triangleq$ population along route k from source i to site j

for 2 possible routes from each source i to each possible disposal site. We wish to minimize some combination of distance and population to select the best sites and routes.

³Based on C. ReVelle, J. Cohon, and D. Shobrys (1991), "Simultaneous Siting and Routing in the Disposal of Hazardous Wastes," *Transportation Science*, 25, 138–145.

TABLE 8.2 Hazardous Waste Example

Source <i>i</i>		Site <i>j</i> = 1		Site <i>j</i> = 2		Site <i>j</i> = 3		Supply
		<i>k</i> = 1	<i>k</i> = 2	<i>k</i> = 1	<i>k</i> = 2	<i>k</i> = 1	<i>k</i> = 2	
1	Distance	200	280	850	1090	900	1100	1.2
	Population	50	15	300	80	400	190	
2	Distance	400	530	730	860	450	600	0.5
	Population	105	60	380	210	350	160	
3	Distance	600	735	550	600	210	240	0.3
	Population	300	130	520	220	270	140	
4	Distance	900	1060	450	570	180	360	0.7
	Population	620	410	700	430	800	280	
5	Distance	600	640	390	440	360	510	0.6
	Population	205	180	440	370	680	330	
6	Distance	900	1240	100	120	640	800	0.1
	Population	390	125	80	30	800	410	
7	Distance	1230	1410	400	460	1305	1500	0.2
	Population	465	310	180	105	1245	790	

Hazardous Waste Disposal Model

To model this shipping and site selection problem, we require the two sorts of decision variables typical of facility location models:

$$y_i \triangleq \begin{cases} 1 & \text{if site } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{i,j,k} \triangleq \text{amount shipped from source } i \text{ to site } j \text{ along route } k$$

Then a multiobjective integer linear programming model is

$$\begin{aligned}
 \min \quad & \sum_{i=1}^7 \sum_{j=1}^3 \sum_{k=1}^2 d_{i,j,k} x_{i,j,k} && \text{(distance)} \\
 \min \quad & \sum_{i=1}^7 \sum_{j=1}^3 \sum_{k=1}^2 p_{i,j,k} x_{i,j,k} && \text{(population)} \\
 \text{s.t.} \quad & \sum_{j=1}^3 \sum_{k=1}^2 x_{i,j,k} = s_i \quad i = 1, \dots, 7 && \text{(sources)} \\
 & \sum_{j=1}^3 y_j = 2 && \text{(2 sites)} \\
 & x_{i,j,k} \leq s_i y_j \quad i = 1, \dots, 7; \quad j = 1, \dots, 3; \quad k = 1, 2 \\
 & x_{i,j,k} \geq 0 \quad i = 1, \dots, 7; \quad j = 1, \dots, 3; \quad k = 1, 2 \\
 & y_j = 0 \text{ or } 1 \quad j = 1, \dots, 3
 \end{aligned} \tag{8.3}$$

The first main constraints assure that all material arising at each source is shipped somewhere, and the second selects 2 sites. Switching constraints makes sure that a site is opened before any material can be shipped there.

8.2 EFFICIENT POINTS AND THE EFFICIENT FRONTIER

The familiar notion of an “optimal” solution becomes somewhat murky when an optimization model has more than one objective function. A solution that proves best by one criterion may rate among the poorest on another. In this section we develop the concept of **efficient points** and the **efficient frontier**, also known as **Pareto optima** and **nondominated points**, which help to characterize “best” feasible solutions in multiobjective models.

Efficient Points

A feasible solution to an optimization model cannot be best if there are others that equal or improve on it by every measure. Efficient points are ones that cannot be dominated in this way.

Definition 8.1 A feasible solution to a multiobjective optimization model is an **efficient point** if no other feasible solution scores at least as well in all objective functions and strictly better in one.

Efficient points are also termed Pareto optimal and nondominated.

We can illustrate with the dynamometer ring design model of Section 8.1:

$$\begin{aligned}
 \max \quad & \frac{0.7r}{(2.1 \times 10^6)(wt^2)} && \text{(sensitivity)} \\
 \max \quad & \frac{(2.1 \times 10^6)(wt^3)}{r^3} && \text{(rigidity)} \\
 & 5.0 \leq w \leq 10.0 \\
 & 0.1 \leq t \leq 2.0 \\
 & 1.25 \leq r \leq 20.0
 \end{aligned} \tag{8.4}$$

Consider solution $(w^{(1)}, t^{(1)}, r^{(1)}) = (5, 0.1, 20)$. No feasible point has higher sensitivity, because the sole variable r in the numerator of the sensitivity objective is at its upper bound, and both w and t in the denominator are at lower bounds. But there are feasible points with higher rigidity; we need only decrease radius r . Even though it can be improved in one objective, the point is efficient. Any such change to improve rigidity results in a strict decrease in sensitivity.

Contrast with the solution $(w^{(2)}, t^{(2)}, r^{(2)}) = (7, 1, 3)$. Its values in the two objective functions are

$$\begin{aligned}
 \text{sensitivity} &= \frac{0.7(3)}{(2.1 \times 10^6)(7)(1)^2} = 1.429 \times 10^{-7} \\
 \text{rigidity} &= \frac{(2.1 \times 10^6)(7)(1)^3}{(3)^3} = 5.444 \times 10^5
 \end{aligned}$$

This feasible point cannot be efficient because it is dominated by $(w^{(3)}, t^{(3)}, r^{(3)}) = (7, 0.9, 2.7)$. The latter has the same rigidity and superior sensitivity 1.587×10^{-7} .

Identifying Efficient Points Graphically

When a multiobjective optimization model has only two decision variables, we can see graphically whether points are efficient. Consider, for example, the simple multiobjective linear program

$$\begin{aligned}
 &\max && + 3x_1 + 1x_2 \\
 &\max && - 1x_1 + 2x_2 \\
 &\text{s.t.} && x_1 + x_2 \leq 4 \\
 &&& 0 \leq x_1 \leq 3 \\
 &&& 0 \leq x_2 \leq 3
 \end{aligned} \tag{8.5}$$

plotted in Figure 8.3. Feasible solution $\mathbf{x} = (2, 2)$ of part (a) is efficient and the $\mathbf{x} = (3, 0)$ of part (b) is not.

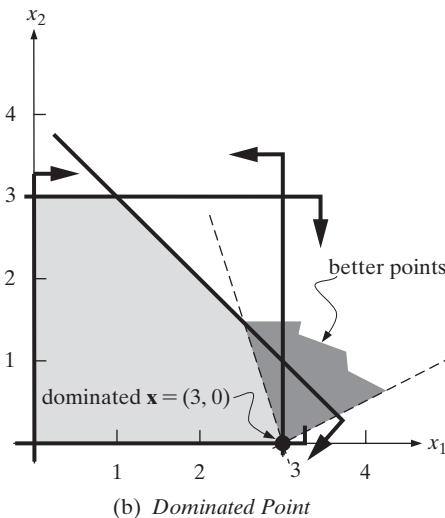
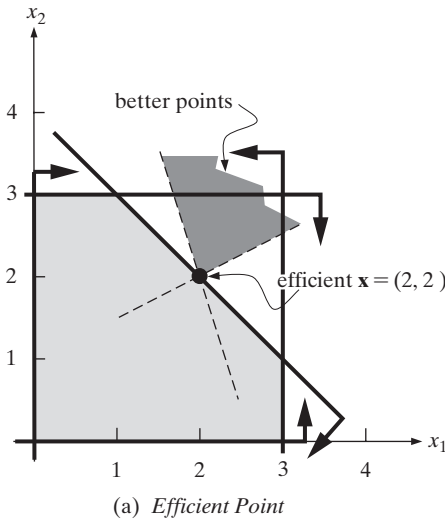


FIGURE 8.3 Graphical Characterization of Efficient Points

How can we be sure? In each case we have plotted the contours of the two objective functions that pass through the point. Other solutions with equal or better values of both objective functions must belong to the dark shaded areas bounded by those contours. In part (a) no such point is feasible. In part (b), others, such as $\mathbf{x}' = (3, 1)$, satisfy all constraints and thus dominate.

Principle 8.2 Efficient points show graphically as ones for which no other feasible point lies in the region bounded by contours of the objective functions through the point, which contains every solution with equal or superior value of all objectives.

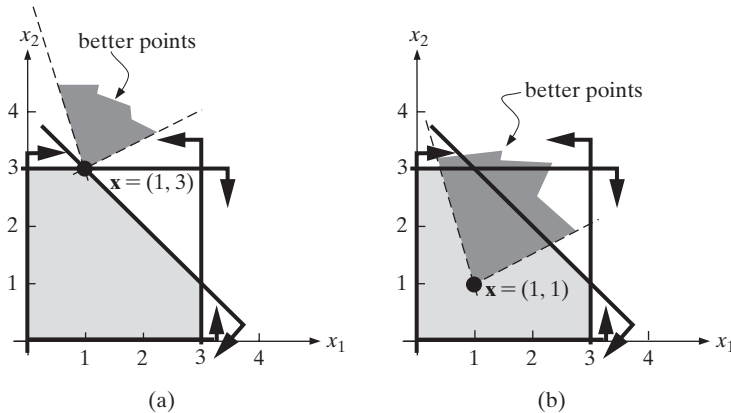
EXAMPLE 8.1: IDENTIFYING EFFICIENT POINTS

Determine whether each of the following points is efficient in model (8.5) and Figure 8.3.

(a) $\mathbf{x} = (1, 3)$

(b) $\mathbf{x} = (1, 1)$

Solution: We apply principle [8.2](#) in the following plots:



(a) In plot (a), the set of possibly dominant points, which is bounded by the contours, contains no other feasible point. Solution $\mathbf{x} = (1, 3)$ is efficient.

(b) In plot (b), the set of possibly dominant points, which is bounded by the contours, contains numerous feasible solutions. Solution $\mathbf{x} = (1, 1)$ cannot be efficient.

Efficient Frontier

When confronting a multiobjective optimization model, it seems natural to demand an efficient solution. But we have seen that very simple examples can have many efficient points. Some will score better by one criterion; others will evaluate superior by another.

To deal with such conflicts, we would often like to generate and consider a range of efficient solutions. The entire set is called the efficient frontier.

Definition 8.3 The **efficient frontier** of a multiobjective optimization model is the collection of efficient points for the model.

Plots in Objective Value Space

The term “efficient frontier” comes from an alternative way to plot solutions to multiobjective models. Instead of using axes corresponding to the decision variables, we plot in **objective value space** with axes for the different objective functions.

Figure 8.4 illustrates for our dynamometer ring design example (8.4). Every feasible solution for the model corresponds to a point in this plot, with horizontal dimension equal to its sensitivity, and vertical equal to its rigidity. For example, efficient point $(w^{(1)}, t^{(1)}, r^{(1)}) = (5, 0.1, 20)$ produces point $(1.333 \times 10^{-4}, 1.312)$ at the lower right. Dominated point $(w^{(2)}, t^{(2)}, r^{(2)}) = (7, 1, 3)$ graphs as $(1.429 \times 10^{-7}, 5.444 \times 10^5)$.

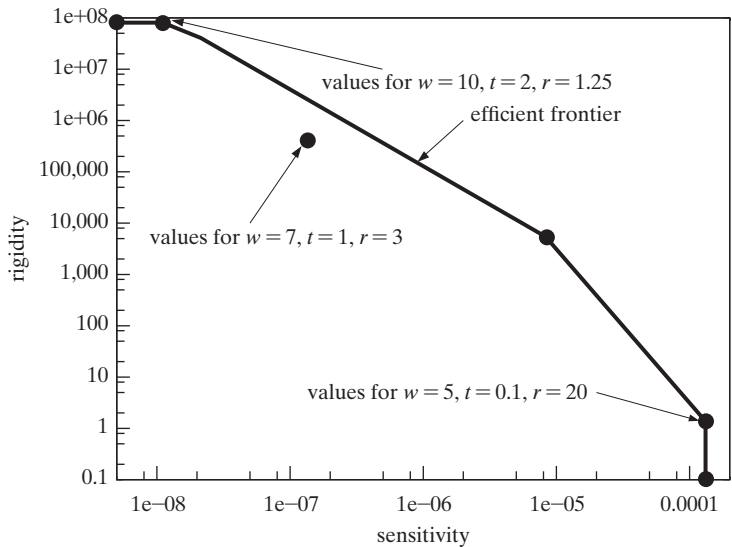


FIGURE 8.4 Efficient Frontier for Dynamometer Ring Application

The efficient frontier forms the boundary of the region defined by objective values for feasible points. Every efficient point lies along this boundary because no further progress is possible in one objective function without degrading another. On the other hand, dominated points plot in the interior; other feasible solutions along the boundary are equal in all objectives and superior in at least one.

Constructing the Efficient Frontier

When a multiobjective optimization model has only a few objectives, it is often practical to construct efficient frontier curves such as Figure 8.4. We have only to parametrically vary specified levels of all but one objective while optimizing the other.

Principle 8.4 The set of points on the efficient frontier can be constructed by repeated optimization. New constraints enforce achievement levels for all but one objective, and the other is treated as a single objective.

The process parallels the parametric sensitivity analysis of Section 6.7.

Table 8.3 details the computations used to construct the efficient frontier of the dynamometer ring example in Figure 8.4. We begin by maximizing the two objectives separately. Maximizing sensitivity without regard to rigidity produces the first point (1.333×10^{-4} , 1.312) in the table. Then maximizing rigidity without considering sensitivity yields to the last point (1.042×10^{-8} , 8.6×10^7).

TABLE 8.3 Efficient Frontier of the Dynamometer Ring Application

Sensitivity Objective	Rigidity Objective	Efficient Point		
		w	t	r
1.333×10^{-4}	1.312	5.00	0.100	20.00
6.776×10^{-5}	10^1	5.00	0.100	10.16
3.145×10^{-5}	10^2	5.00	0.100	4.72
1.460×10^{-5}	10^3	5.00	0.100	2.19
5.510×10^{-6}	10^4	5.00	0.123	1.25
1.187×10^{-6}	10^5	5.00	0.265	1.25
2.557×10^{-7}	10^6	5.00	0.571	1.25
5.510×10^{-8}	10^7	5.00	1.230	1.25
1.042×10^{-8}	8.6×10^7	10.00	2.000	1.25

We now know the range of relevant rigidity values. The remaining points in Table 8.3 are obtained by maximizing sensitivity subject to a constraint on rigidity. For example, the values for rigidity 10^3 result from solving

$$\begin{aligned}
 \max \quad & \frac{0.7r}{(2.1 \times 10^6)(wt^2)} && \text{(sensitivity)} \\
 \text{s.t.} \quad & \frac{(2.1 \times 10^6)(wt^3)}{r^3} \geq 10^3 && \text{(rigidity)} \\
 & 5.0 \leq w \leq 10.0 \\
 & 0.1 \leq t \leq 2.0 \\
 & 1.25 \leq r \leq 20.0
 \end{aligned}$$

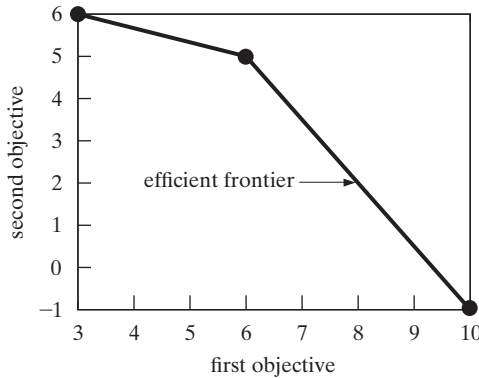
EXAMPLE 8.2: CONSTRUCTING THE EFFICIENT FRONTIER

Return to example model (8.5) of Figure 8.3 and construct an objective value space diagram of its efficient frontier like the one in Figure 8.4.

Solution: We begin by separately maximizing the first and second objectives to obtain solutions with objective values (10, -1) and (3,6), respectively. Now solving the LP

$$\begin{aligned}
 \max \quad & +3x_1 + 1x_2 \\
 \text{s.t.} \quad & -1x_1 + 2x_2 \geq \theta \\
 & x_1 + x_2 \leq 4 \\
 & 0 \leq x_1 \leq 3 \\
 & 0 \leq x_2 \leq 3
 \end{aligned}$$

for several values of $\theta \in [-1, 6]$ produces the following plot:



8.3 PREEMPTIVE OPTIMIZATION AND WEIGHTED SUMS OF OBJECTIVES

In a typical multiobjective model of realistic size, especially one with more than two objectives, the range of efficient solutions can be enormous. Explicit construction of an efficient frontier like Figure 8.4 is computationally impractical.

To obtain useful results, we must reduce the multiobjective model to a sequence of single objective optimizations. In this section, we explore two of the most straightforward ways: **preemptive** (or **lexicographic**) **optimization** and **weighted sums**.

Preemptive Optimization

Although a model may have several objective criteria, they are rarely of equal importance. Preemptive optimization takes them in priority order.

Definition 8.5 | **Preemptive or lexicographic optimization** performs multiobjective optimization by considering objectives one at a time. The most important is optimized; then the second most important is optimized subject to a requirement that the first achieve its optimal value; and so on.

Preemptive Optimization of the Bank Three Application

We can illustrate with the Bank Three application, model (8.1). That model has three objectives: profit, capital-adequacy ratio, and risk-asset ratio.

Suppose we decide that the third, risk-asset, objective is the single most important one. Preemptive optimization would then begin by minimizing risk-asset ratio in the single objective linear program

$$\begin{aligned}
 \min \quad & \frac{1}{20}(x_6 + x_7 + x_8) && \text{(risk-asset)} \\
 \text{s.t.} \quad & x_1 + \cdots + x_8 = 20 + 150 + 80 && \text{(invest all)} \\
 & x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\
 & 1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 && \text{(liquidity)} \\
 & \quad + 0.850x_5 \geq 0.47(150) + 0.36(80)
 \end{aligned}$$

$$\begin{aligned}
 x_j &\geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
 x_8 &\geq 0.30(20 + 150 + 80) && \text{(commercial)} \\
 x_1, \dots, x_8 &\geq 0
 \end{aligned}$$

An optimal solution allocates funds (in millions of dollars)

$$\begin{aligned}
 x_1^* &= 100.0, & x_2^* &= 12.5, & x_3^* &= 12.5, & x_4^* &= 12.5 \\
 x_5^* &= 12.5, & x_6^* &= 12.5, & x_7^* &= 12.5, & x_8^* &= 75.0
 \end{aligned} \tag{8.6}$$

with optimal risk-asset ratio 5.0, profit \$11.9 million, and capital-adequacy ratio 0.606.

Next, we introduce a constraint keeping risk-asset ratio at the optimal 5.0 and maximize the second priority, profit objective:

$$\begin{aligned}
 \max \quad & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\
 & + 0.105x_6 + 0.085x_7 + 0.092x_8 \\
 \text{s.t.} \quad & \frac{1}{20}(x_6 + x_7 + x_8) \leq 5.0 && \text{(risk-asset)} \\
 & x_1 + \dots + x_8 = 20 + 150 + 80 && \text{(invest all)} \\
 & x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\
 & 1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 && \text{(liquidity)} \\
 & + 0.850x_5 \geq 0.47(150) + 0.36(80) \\
 & x_j \geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
 & x_8 \geq 0.30(20 + 150 + 80) && \text{(commercial)} \\
 & x_1, \dots, x_8 \geq 0
 \end{aligned}$$

The result is (millions of dollars)

$$\begin{aligned}
 x_1^* &= 24.2, & x_2^* &= 12.5, & x_3^* &= 12.5, & x_4^* &= 12.5 \\
 x_5^* &= 88.3, & x_6^* &= 12.5, & x_7^* &= 12.5, & x_8^* &= 75.0
 \end{aligned} \tag{8.7}$$

with optimal risk-asset ratio still 5.0, but profit now \$17.2 million, and capital-adequacy ratio 0.890.

Notice that the character of the optimum has changed significantly from solution (8.6). Considerable funds have been shifted from x_1 = cash, to x_5 = long-term government bonds in order to increase profit.

Finally, we come to the capital-adequacy objective function. Imposing a new constraint for profit, we solve

$$\begin{aligned}
 \min \quad & \frac{1}{20}(0.005x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5) && \text{(capital-adequacy)} \\
 & + 0.100x_6 + 0.100x_7 + 0.100x_8 \\
 \text{s.t.} \quad & \frac{1}{20}(x_6 + x_7 + x_8) \leq 5.0 && \text{(risk-asset)} \\
 & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\
 & + 0.105x_6 + 0.085x_7 + 0.092x_8 \geq 17.2 \\
 & x_1 + \dots + x_8 = 20 + 150 + 80 && \text{(invest all)} \\
 & x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)}
 \end{aligned}$$

$$\begin{aligned}
 &1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 && \text{(liquidity)} \\
 &\quad + 0.850x_5 \geq 0.47(150) + 0.36(80) \\
 &x_j \geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
 &x_8 \geq 0.30(20 + 150 + 80) && \text{(commercial)} \\
 &x_1, \dots, x_8 \geq 0
 \end{aligned}$$

This time solution (8.7) remains optimal. The capital-adequacy ratio cannot be decreased without worsening other objectives.

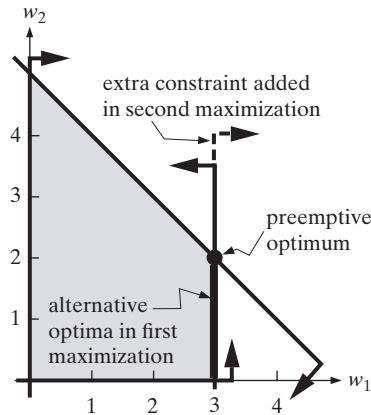
EXAMPLE 8.3: SOLVING MULTIOBJECTIVE MODELS PREEMPTIVELY

Consider the multiobjective mathematical program

$$\begin{aligned}
 &\max \quad w_1 \\
 &\max \quad 2w_1 + 3w_2 \\
 &\text{s.t.} \quad w_1 \leq 3 \\
 &\quad \quad w_1 + w_2 \leq 5 \\
 &\quad \quad w_1, w_2 \geq 0
 \end{aligned}$$

Solve the model graphically by preemptive optimization, taking objectives in the sequence given.

Solution: Graphic solution produces the following plot:



A first optimization maximizes w_1 subject to the given constraints. Any of the solutions along the line segment from (3, 0) to (3, 2) is optimal.

Next, we impose an extra constraint

$$w_1 \geq 3$$

and maximize the second objective. The result is final preemptive solution $\mathbf{w} = (3, 2)$.

Preemptive Optimization and Efficient Points

One advantage of the preemptive approach to multiobjective optimization is that it results in solutions that cannot be improved in one objective without degrading another.

Principle 8.6 | If each stage of preemptive optimization yields a single-objective optimum, the final solution is an efficient point of the full multiobjective model.

The preemptive process requires that we try objective functions in turn, trying to improve one without worsening the others. When we finish, no further improvement is possible. As usual, infeasible and unbounded cases can produce complications, but the typical outcome is an efficient point.

Preemptive Optimization and Alternative Optima

Although it usually will produce an efficient point, a moment's reflection on the preemptive optimization approach will reveal a major limitation.

Principle 8.7 | After one objective function has been optimized in preemptive processing of a multiobjective model, solutions obtained in subsequent stages must all be alternative optima in the first.

That is, preemptive optimization places great emphasis on the first objective addressed, with all later steps limited to alternative optima in the highest-priority objective.

The Bank Three computations above showed considerable change from initial optima (8.6) to final (8.7) because the first, risk-asset objective

$$\min \frac{1}{20} (x_6 + x_7 + x_8)$$

admits many alternative optima among the unmentioned x_1, \dots, x_5 . But with cases where alternative optima are rare, the preemptive approach becomes essentially one of optimizing a priority objective while ignoring all the others.

Weighted Sums of Objectives

An alternative scheme for dealing with multiobjective models that permits more balanced handling of the objectives is simply to combine them in a **weighted sum**.

Principle 8.8 | Multiple objective functions can be combined into a single composite one to be maximized by summing objectives with positive weights on maximizes and negative weights on minimizes. If the composite is to be minimized, weights on maximize objectives should be negative, and those on minimizes should be positive.

Signs orient all objectives in the same direction, and weights reflect their relative importance.

EXAMPLE 8.4: FORMING WEIGHTED OBJECTIVES

Form a single weighted-sum composite objective from each of the following collections of objective functions. Indicate whether weights γ_i should be positive or negative and whether the composite objective should be maximized or minimized.

$$\begin{aligned} \text{(a) min} \quad & +2w_1 \quad + 3w_2 \quad - 1w_3 \\ \text{max} \quad & +4w_1 \quad - 2w_2 \\ \text{max} \quad & \quad \quad + 1w_2 \quad + 1w_3 \end{aligned}$$

$$\begin{aligned} \text{(b) min} \quad & +3w_1 \quad - 1w_2 \\ \text{min} \quad & +4w_1 \quad + 2w_2 \quad + 9w_3 \end{aligned}$$

Solution: We apply principle [8.8].

(a) Using weights $\gamma_1, \dots, \gamma_3$, the weighted objective is

$$\max (2\gamma_1 + 4\gamma_2)w_1 + (3\gamma_1 - 2\gamma_2 + 1\gamma_3)w_2 + (-1\gamma_1 + 1\gamma_3)w_3$$

This maximize composite form requires that $\gamma_1 < 0$, $\gamma_2 > 0$, and $\gamma_3 > 0$.

(b) Using weights γ_1 and γ_2 , the weighted objective is

$$\min (3\gamma_1 + 4\gamma_2)w_1 + (-1\gamma_1 + 2\gamma_2)w_2 + (9\gamma_2)w_3$$

This minimize composite form requires that $\gamma_1 > 0$ and $\gamma_2 > 0$.

Weighted-Sum Optimization of the Hazardous Waste Application

Hazardous waste planning model (8.3):

$$\min \sum_{i=1}^7 \sum_{j=1}^3 \sum_{k=1}^2 d_{i,j,k} x_{i,j,k} \quad (\text{distance})$$

$$\min \sum_{i=1}^7 \sum_{j=1}^3 \sum_{k=1}^2 p_{i,j,k} x_{i,j,k} \quad (\text{population})$$

$$\text{s.t.} \quad \sum_{j=1}^3 \sum_{k=1}^2 x_{i,j,k} = s_i \quad i = 1, \dots, 7 \quad (\text{sources})$$

$$\sum_{j=1}^3 y_j = 2 \quad (2 \text{ sites})$$

$$x_{i,j,k} \leq s_i y_j \quad i = 1, \dots, 7; \quad j = 1, \dots, 3; \quad k = 1, 2$$

$$x_{i,j,k} \geq 0 \quad i = 1, \dots, 7; \quad j = 1, \dots, 3; \quad k = 1, 2$$

$$y_i = 0 \text{ or } 1 \quad j = 1, \dots, 3$$

illustrates a setting where weighted-sum analysis of a multiobjective model can be usefully applied. The first objective minimizes the shipping distance to chosen disposal sites. The second minimizes the population exposed along the way. Each source is provided 2 alternative routes to each potential disposal site, one denoted $k = 1$, emphasizing short distance, and the other $k = 2$, avoiding population.

Since both objectives minimize, we may produce a single composite objective by applying weights $\gamma_1, \gamma_2 > 0$ and minimizing the result (principle [8.8]):

$$\min \sum_{i=1}^7 \sum_{j=1}^3 \sum_{k=1}^2 (\gamma_1 d_{i,j,k} + \gamma_2 p_{i,j,k}) x_{i,j,k} \quad (\text{composite})$$

Table 8.4 illustrates the impact for some different combinations of weights. With comparatively high weight γ_1 on distance, the optimization routes almost all along short routes $k = 1$. As population is given greater relative weight, activity shifts to the longer routes that avoid population except when the optimal sites change at $\gamma_1 = \gamma_2 = 10$. Eventually, all shipping is along the safer routes. Confronted with a range of such alternatives, decision makers could decide an appropriate balance.

TABLE 8.4 Weighting Objectives in the Hazardous Waste Application

Weight		Ton-Miles		Total Ton-Miles	Total Ton-Population	Optimal Sites
γ_1	γ_2	$k = 1$	$k = 2$			
10	1	1155	0	1155	1334.5	1.3
10	5	754	591	1345	782.5	1.3
10	10	1046	404	1450	607.5	1.2
5	10	440	1114	1554	533.0	1.3
1	10	0	1715	1715	468.5	1.3

Weighted-Sum Optimization and Efficient Points

Although offering more flexibility in trading off objectives, using weighted totals still assures an efficient solution.

Principle 8.9 | If a single weighted-sum objective model derived from a multiobjective optimization as in [8.8] produces an optimal solution, the solution is an efficient point of the multiobjective model.

To see why this is true, we need only consider the nature of a weighted-sum objective:

$$\begin{aligned} & \left(\begin{array}{c} \text{weight} \\ 1 \end{array} \right) \left(\begin{array}{c} \text{objective} \\ 1 \text{ value} \end{array} \right) + \left(\begin{array}{c} \text{weight} \\ 2 \end{array} \right) \left(\begin{array}{c} \text{objective} \\ 2 \text{ value} \end{array} \right) + \dots \\ & + \left(\begin{array}{c} \text{weight} \\ p \end{array} \right) \left(\begin{array}{c} \text{objective} \\ p \text{ value} \end{array} \right) \end{aligned}$$

With signs as in construction [8.8], any solution that can improve in one objective without degrading the others would also score better in the weighted objective. Only an efficient point could be optimal.

8.4 GOAL PROGRAMMING

Multiobjective models of complex problems assume that we always want more of everything—lower values of objectives being minimized at the same time as higher values of criteria being maximized. Notice that this is independent of how much we may already have achieved on one or another objective. For example, a multiobjective model would keep the same priority or weight on a minimize cost objective whether or not we already have in hand a solution with extraordinarily low cost.

In this section, we explore the **goal programming** alternative, which is constructed in terms of target levels to be achieved rather than quantities to be maximized or minimized. It is probably more realistic to assume that the importance of any criterion diminishes once a target level has been achieved. We will certainly see that it is easier to implement. That is why goal programming is by far the most popular approach to finding good solutions in multicriteria optimization settings.

Goal or Target Levels

Goal program modeling of a multiobjective optimization begins by asking decision makers to specify new data: goal or target levels for each of the criteria used to evaluate solutions.

Definition 8.10 | **Goal or target levels** specify the values of the criteria functions in an optimization model that decision makers consider sufficient or satisfactory.

Goal Form of Bank Three Application

To illustrate, return to our Bank Three application of Section 8.1.

$$\begin{aligned}
 \max \quad & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\
 & + 0.105x_6 + 0.085x_7 + 0.092x_8 \\
 \min \quad & \frac{1}{20} (0.05x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5) && \text{(capital-adequacy)} \\
 & + 0.100x_6 + 0.100x_7 + 0.100x_8 \\
 \min \quad & \frac{1}{20} (x_6 + x_7 + x_8) && \text{(risk-asset)} \\
 \text{s.t.} \quad & x_1 + \cdots + x_8 = (20 + 150 + 80) && \text{(invest all)} \\
 & x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\
 & 1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 && \text{(liquidity)} \\
 & \quad + 0.85x_5 \geq 0.47(150) + 0.36(80) \\
 & x_j \geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
 & x_8 \geq 0.30(20 + 150 + 80) && \text{(commerical)} \\
 & x_1, \dots, x_8 \geq 0
 \end{aligned}$$

Here solutions are evaluated on three criteria: profit, capital-adequacy ratio, and risk-asset ratio. Assume that instead of seeking ever higher levels of the first criterion and lower values of the last two, we set some goals:

$$\begin{aligned} \text{profit} &\geq 18.5 \\ \text{capital-adequacy ratio} &\leq 0.8 \\ \text{risk-asset ratio} &\leq 7.0 \end{aligned} \tag{8.8}$$

Then the problem might be specified in goal format as

$$\begin{aligned} \text{goal } &0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\ &+ 0.105x_6 + 0.085x_7 + 0.092x_8 \geq 18.5 \\ \text{goal } &\frac{1}{20}(0.05x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5) && \text{(capital-adequacy)} \\ &+ 0.100x_6 + 0.100x_7 + 0.100x_8 \leq 0.8 \\ \text{goal } &\frac{1}{20}(x_6 + x_7 + x_8) \geq 7.0 && \text{(risk-asset)} \\ \text{s.t. } &x_1 + \cdots + x_8 = (20 + 150 + 80) && \text{(invest all)} \\ &x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\ &1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 && \text{(liquidity)} \\ &+ 0.85x_5 \geq 0.47(150) + 0.36(80) \\ &x_j \geq 0.05(20 + 150 + 80) \text{ for all } j = 1, \dots, 8 && \text{(diversification)} \\ &x_8 \geq 0.30(20 + 150 + 80) && \text{(commerical)} \\ &x_1, \dots, x_8 \geq 0 \end{aligned} \tag{8.9}$$

Soft Constraints

Goals in statement (8.9) may be thought of as soft constraints.

Definition 8.11 | **Soft constraints** such as the criteria targets of goal programming specify requirements that are desirable to satisfy but which may still be violated in feasible solutions.

The more familiar **hard constraints** still determine what solutions are feasible, leaving the soft ones to influence which solutions are preferred.

Deficiency Variables

Once target levels have been specified for soft constraints, we proceed to a more familiar mathematical programming formulation by adding constraints that enforce goal achievement. However, we cannot just impose the constraint that each objective meet its goal. There may be no solution that simultaneously achieves the desired levels of all soft constraints. Instead, we introduce new deficiency variables.

Definition 8.12 | Nonnegative **deficiency variables** are introduced to model the extent of violation in goal or other soft constraints that need not be rigidly enforced. With $a \geq$ target, the deficiency is the underachievement. With $a \leq$ target, it is the excess. With $=$ soft constraints, deficiency variables are included for both under- and overachievement.

In the 3-objective Bank Three application (8.9), we enforce goal levels (8.8) with deficiency variables

- $d_1 \triangleq$ amount profit falls short of its goal
- $d_2 \triangleq$ amount capital-adequacy ratio exceeds its goal
- $d_3 \triangleq$ amount risk-asset ratio exceeds its goal

Expressing Soft Constraints in Mathematical Programs

Goal and other soft constraints can now be expressed in the usual (hard) mathematical programming format with deficiency variables allowing violation.

Principle 8.13 | Goal or soft constraints use nonnegative deficiency variables to express $a \geq$ target

$$(\text{criterion function}) + (\text{deficiency variable}) \geq \text{target value}$$

and $a \leq$ target

$$(\text{criterion function}) - (\text{deficiency variable}) \leq \text{target value}$$

Equality-form soft constraints are modeled

$$(\text{criterion function}) - (\text{oversatisfaction deficiency variable}) + (\text{undersatisfaction deficiency variable}) = \text{target value}$$

For example, the three goals in Bank Three model (8.9) lead to the following main (and variable type) constraints:

$$\begin{aligned}
 &0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\
 &+ 0.105x_6 + 0.085x_7 + 0.092x_8 + d_1 \geq 18.5 \\
 &\frac{1}{20} (0.005x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5 && \text{(capital-adequacy)} \\
 &+ 0.100x_6 + 0.100x_7 + 0.100x_8) - d_2 \leq 0.8 \\
 &\frac{1}{20} (x_6 + x_7 + x_8) - d_3 \leq 7.0 && \text{(risk-asset)} \\
 &d_1, d_2, d_3 \geq 0
 \end{aligned}$$

The first keeps profit at least 18.5 or makes up the difference with deficiency variable d_1 . The other two main constraints force the capital-adequacy and risk-asset ratios below our target value unless the corresponding deficiency variables are nonzero.

Notice that nonnegativity constraints are required. We want the deficiency to compute as zero if the target is achieved.

EXAMPLE 8.5: FORMULATING GOAL CONSTRAINTS

Return to the multiobjective model

$$\begin{aligned} \max \quad & w_1 \\ \max \quad & 2w_1 + 3w_2 \\ \text{s.t.} \quad & w_1 \leq 3 \\ & w_1 + w_2 \leq 5 \\ & w_1, w_2 \geq 0 \end{aligned}$$

of Example 8.3, and assume that instead of seeking to maximize the two criteria, we decide to seek a target level of 2.0 on the first and 14.0 on the second. Introduce deficiency variables and formulate new goal constraints to model these soft constraints as those of a linear program.

Solution: We apply construction [8.13](#) using deficiency variables

$d_1 \triangleq$ undersatisfaction of the first goal

$d_2 \triangleq$ undersatisfaction of the second goal

Then the new linear constraints are

$$\begin{aligned} w_1 \quad \quad \quad & + d_1 \geq 2.0 \\ 2w_1 + 3w_2 \quad & + d_2 \geq 14.0 \\ d_1, d_2 & \geq 0 \end{aligned}$$

Goal Program Objective Function: Minimizing (Weighted) Deficiency

Having modeled undersatisfaction of goals with deficiency variables, we complete a formulation by minimizing violation.

Principle 8.14 The objective in a goal programming model expresses the desire to satisfy all goals as nearly as possible by minimizing a weighted sum of the deficiency variables.

Often, all deficiency is weighted equally.

Goal Linear Program Model of the Bank Three Application

Using equal goal weights in our Bank Three application (8.9) produces the goal linear program

$$\begin{aligned} \min \quad & d_1 + d_2 + d_3 && \text{(total deficiency)} \\ \text{s.t.} \quad & 0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\ & + 0.105x_6 + 0.085x_7 + 0.092x_8 + d_1 \geq 18.5 \end{aligned}$$

$$\begin{aligned}
 & \frac{1}{20} (0.005x_2 + 0.040x_3 + 0.050x_4 + 0.075x_5 + 0.100x_6 + 0.100x_7 + 0.100x_8) - d_2 \leq 0.8 && \text{(capital-adequacy)} \\
 & \frac{1}{20} (x_6 + x_7 + x_8) - d_3 \leq 7.0 && \text{(risk-asset)} \\
 & x_1 + \dots + x_8 = 20 + 150 + 80 && \text{(invest all)} \\
 & x_1 \geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\
 & 1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 + 0.850x_5 \geq 0.47(150) + 0.36(80) && \text{(liquidity)} \\
 & x_j \geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
 & x_8 \geq 0.30(20 + 150 + 80) && \text{(commercial)} \\
 & x_1, \dots, x_8 \geq 0 \\
 & d_1, d_2, d_3 \geq 0
 \end{aligned} \tag{8.10}$$

The goals have been expressed with new constraints involving deficiency variables. All original constraints are retained.

Alternative Deficiency Weights in the Objective

Table 8.5 shows an optimal solution to equal-weighted goal LP in column (1). Notice how it seeks only the \$18.5 million goal for profit and the 7.0 goal for risk-asset ratio. Once corresponding deficiency variables d_1 and d_2 are driven to = 0.0, effort can be directed toward the remaining capital-adequacy goal.

TABLE 8.5 Goal Programming Solution of the Bank Three Example

	(1) Equal Weights	(2) Unequal Weights	(3) Preempt Profit	(4) Preempt Profit, CA	(5) Preempt One Step
Profit goal weight	1	1	1	0	10,000
Cap-adequacy goal weight	1	10	0	1	100
Risk-asset goal weight	1	1	0	0	1
Extra constraints	—	—	—	$d_1 = 0$	—
Profit	18.50	17.53	18.50	18.50	18.50
Deficiency, d_1^*	0.00	0.97	0.00	0.00	0.00
Capital-adequacy ratio	0.928	0.815	0.943	0.919	0.919
Deficiency, d_2^*	0.128	0.015	0.143	0.119	0.119
Risk-asset ratio	7.000	7.000	7.097	7.158	7.158
Deficiency, d_3^*	0.000	0.000	0.097	0.158	0.158
Cash, x_1^*	24.20	24.20	24.20	24.20	24.20
Short term, x_2^*	16.03	48.30	12.50	19.73	19.73
Government: 1–5, x_3^*	12.50	12.50	12.50	12.50	12.50
Government: 5–10, x_4^*	12.50	12.50	12.50	12.50	12.50
Government: over 10, x_5^*	44.77	12.50	46.37	37.91	37.91
Installment, x_6^*	52.50	52.50	41.08	55.67	55.67
Mort gages, x_7^*	12.50	12.50	12.50	12.50	12.50
Commercial, x_8^*	75.00	75.00	88.36	75.00	75.00

There is no requirement that deficiency weights must be equal. In our Bank Three example, the magnitude of the capital-adequacy ratio is much smaller than that of the other two objectives. Thus better results might be obtained by a different weighting that scales deficiencies more equally.

Column (2) of Table 8.5 shows the effect of multiplying the capital adequacy weight by 10. Now the profit slips below its \$18.5 million goal (to \$17.53 million), but the capital-adequacy ratio deficiency is reduced.

EXAMPLE 8.6: FORMULATING GOAL PROGRAMS

Assigning equal weights to violations of the two goals, formulate the goal linear program corresponding to the constraints and target values of Example 8.5.

Solution: Including the goal constraints of Example 8.5 and minimizing total deficiency as in construction [8.14](#) produces the goal linear program

$$\begin{aligned} \min \quad & d_1 + d_2 \\ \text{s.t.} \quad & w_1 + d_1 \geq 2.0 \\ & 2w_1 + 3w_2 + d_2 \geq 14.0 \\ & w_1 \leq 3 \\ & w_1 + w_2 \leq 5 \\ & w_1, w_2 \geq 0 \\ & d_1, d_2 \geq 0 \end{aligned}$$

Notice that all original constraints have been retained.

Preemptive Goal Programming

The preemptive optimization of Section 8.4 (definition [8.5](#)) takes objective functions in order: optimizing one, then a second subject to the first achieving its optimal value, and so on. A similar preemptive goal programming approach is often adopted after criteria have been modeled as goals.

Definition 8.15 | **Preemptive or lexicographic goal programming** considers goals one at a time. Deficiency in the most important is minimized; then deficiency in the second most important is minimized subject to a requirement that the first achieve its minimum; and so on.

Preemptive Goal Programming of the Bank Three Application

Columns (3) and (4) of Table 8.5 illustrate this preemptive variant of goal programming. The first was obtained by focusing entirely on the profit goal. Using the objective function

$$\min \quad d_1$$

we concern ourselves only with achieving the desired profit of \$18.5 million. Column (3) shows a distribution of investments that achieves that goal, albeit at the cost of violating both the others.

Now we turn to the capital-adequacy ratio goal. After adding the extra constraint

$$d_1 = 0$$

to keep the profit goal fully satisfied, we address capital adequacy with objective function

$$\min d_2$$

The result [column (4) of Table 8.5] shifts investments to improve the capital-adequacy ratio while continuing to achieve a profit of \$18.5 million.

To complete the preemptive goal programming solution of this example, we should now address the last risk-asset ratio objective by

$$\min d_3$$

subject to extra constraints

$$d_1 = 0.0, \quad d_2 = 0.119$$

The effect is to seek a solution coming closer to the risk-asset ratio goal without losing ground on either of the other two. For this example the resulting solution is the same as column (4).

EXAMPLE 8.7: DOING PREEMPTIVE GOAL PROGRAMMING SEQUENTIALLY

Return to the weighted goal program of Example 8.6.

- (a) Formulate the first model to be solved if we wish to give highest priority to fulfilling the first goal.
- (b) Assuming that the first goal can be completely achieved in the model of part (a), formulate a second model to seek satisfaction of the second goal subject to satisfying the first.

Solution: We execute optimization sequence 8.15.

- (a) The first model to be solved emphasizes deficiency in goal 1:

$$\begin{aligned} \min \quad & d_1 \\ \text{s.t.} \quad & w_1 + d_1 \geq 2.0 \\ & 2w_1 + 3w_2 + d_2 \geq 14.0 \\ & w_1 \leq 3 \\ & w_1 + w_2 \leq 5 \\ & w_1, w_2 \geq 0 \\ & d_1, d_2 \geq 0 \end{aligned}$$

(b) Assuming that deficiency in the first goal can be completely eliminated, the second optimization to be solved is

$$\begin{aligned}
 \min \quad & d_2 \\
 \text{s.t.} \quad & w_1 + d_1 \geq 2.0 \\
 & 2w_1 + 3w_2 + d_2 \geq 14.0 \\
 & w_1 \leq 3 \\
 & w_1 + w_2 \leq 5 \\
 & w_1, w_2 \geq 0 \\
 & d_2 \geq 0 \\
 & d_1 = 0
 \end{aligned}$$

New constraint $d_1 = 0$ requires that the first goal be fully satisfied. Within that limitation, we minimize violation of the second goal.

Preemptive Goal Programming by Weighting the Objective

If a given model has a number of objective functions (or goals), preemptive goal programming in the successive optimization manner of principle [8.15](#) can become rather tedious. Fortunately, the same effect can be achieved in a single optimization by choosing appropriate weights.

Principle 8.16 Preemptive goal programming can be accomplished in a single step by solving with an objective function that puts great weight on the deficiency in the most important goal, less weight on the deficiency in the second goal, and so on.

We may implement the profit first, capital-adequacy ratio second, risk-asset ratio third prioritization of our Bank Three goal programming model (8.10) in this way by using the objective function

$$\min \quad 10,000d_1 + 100d_2 + 1d_3 \quad (\text{preemptive-weighted deficiency})$$

Column (5) of Table 8.5 shows that this weighting achieves the same solution as the one obtained sequentially in column (4).

Practical Advantage of Goal Programming in Multiobjective Problems

Although the same techniques can be used for any soft constraints, we have seen from our treatment of the Bank Three application how goal programming can provide a practical tool for dealing with multiobjective problem settings. If decision makers can be persuaded to specify target levels for their various objectives—and often that is not easy—modeling as a goal LP or ILP or NLP reduces the multiobjective analysis to standard mathematical programming form.

EXAMPLE 8.8: DOING PREEMPTIVE GOAL PROGRAMMING WITH WEIGHTS

Use appropriate weights to formulate a single goal program that implements the preemptive priority sequence in the example of Example 8.7.

Solution: We apply construction [8.16]. Using multiplier 100 on the first deficiency and 1 on the second assures that every effort will be made to fulfill the first goal before the second is considered. The result is the goal program

$$\begin{aligned}
 \min \quad & 100d_1 + d_2 \\
 \text{s.t.} \quad & w_1 + d_1 \geq 2.0 \\
 & 2w_1 + 3w_2 + d_2 \geq 14.0 \\
 & w_1 \leq 3 \\
 & w_1 + w_2 \leq 5 \\
 & w_1, w_2 \geq 0 \\
 & d_1, d_2 \geq 0
 \end{aligned}$$

Principle 8.17 Goal programming is the most popular approach to dealing with multiobjective optimization problems because it reduces complex multi-objective trade-offs to a standard, single-objective mathematical program in a way that decision makers often find intuitive.

Goal Programming and Efficient Points

We have seen in Section 8.2 that efficient points (definition [8.1]) provide the closest analog to an optimal solution in multiobjective settings because they cannot be improved in one objective without degrading another. Unfortunately, goal programming reformulation of a multiobjective model cannot always guarantee a solution with this desirable property.

Principle 8.18 If a goal program has alternative optimal solutions, some of them may not be efficient points of the corresponding multiobjective optimization model.

To see the difficulties that can occur, suppose that we modify Bank Three goal LP (8.10) by ignoring the capital-adequacy objective. The result is the two-goal form

$$\begin{aligned}
 \min \quad & d_1 + d_3 && \text{(total deficiency)} \\
 \text{s.t.} \quad & 0.04x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 && \text{(profit)} \\
 & + 0.105x_6 + 0.085x_7 + 0.092x_8 + d_1 \geq 18.5 \\
 & + (0.100x_6 + 0.100x_7 + 0.100x_8) - d_2 \leq 0.8 \\
 & \frac{1}{20}(x_6 + x_7 + x_8) - d_3 \leq 7.0 && \text{(risk-asset)}
 \end{aligned}$$

$$\begin{aligned}
x_1 + \cdots + x_8 &= (20 + 150 + 80) && \text{(invest all)} \\
x_1 &\geq 0.14(150) + 0.04(80) && \text{(cash reserve)} \\
1.00x_1 + 0.995x_2 + 0.960x_3 + 0.900x_4 &&& \text{(liquidity)} \\
+ 0.850x_5 &\geq 0.47(150) + 0.36(80) \\
x_j &\geq 0.05(20 + 150 + 80) \quad \text{for all } j = 1, \dots, 8 && \text{(diversification)} \\
x_8 &\geq 0.30(20 + 150 + 80) && \text{(commercial)} \\
x_1, \dots, x_8 &\geq 0 \\
d_1, d_3 &\geq 0
\end{aligned} \tag{8.11}$$

Solving gives the goal programming optimum

$$\begin{aligned}
x_1^* &= 24.2, \quad x_2^* = 12.5, \quad x_3^* = 12.5, \quad x_4^* = 12.5 \\
x_5^* &= 48.3, \quad x_6^* = 44.35, \quad x_7^* = 12.5, \quad x_8^* = 83.15
\end{aligned} \tag{8.12}$$

with profit \$18.5 million and risk-asset ratio 7.00.

Both goals are satisfied completely. Still, solution (8.12) is not efficient because either of the objectives can be improved without worsening the other. For example, the efficient solution (an alternative goal program optimum)

$$\begin{aligned}
x_1^* &= 24.2, \quad x_2^* = 12.5, \quad x_3^* = 12.5, \quad x_4^* = 12.5 \\
x_5^* &= 51.33, \quad x_6^* = 49.47, \quad x_7^* = 12.5, \quad x_8^* = 75.0
\end{aligned} \tag{8.13}$$

continues to produce \$18.5 million in profit but reduces the risk-asset ratio to 6.849.

What makes such nonefficient goal programming solutions occur? A moment's reflection will reveal that the problem is goals that are fully satisfied. Both objectives achieved their target values in (8.12). Once such a solution drives the corresponding deficiency variable = 0, there is nothing in the goal programming formulation to encourage further improvement. Progress stops without reaching an efficient point.

EXAMPLE 8.9: UNDERSTANDING NONEFFICIENT GOAL PROGRAM SOLUTIONS

Consider the simple multiobjective optimization model

$$\begin{aligned}
\min \quad & w_1 \\
\max \quad & w_2 \\
\text{s.t.} \quad & 0 \leq w_1 \leq 2 \\
& 0 \leq w_2 \leq 1
\end{aligned}$$

(a) Introduce deficiency variables to formulate a goal program with target values 1 for the first objective function and 2 for the second.

(b) Show that $(w_1, w_2) = (1, 1)$ is optimal in the goal program of part (a) but not efficient in the original multiobjective form.

(c) Show that $(w_1, w_2) = (0, 1)$ is also optimal in the goal program of part (a) but that it is efficient in the original multiobjective form.

Solution:

(a) Using deficiency variables d_1 in the first objective and d_2 in the second, the required goal program is

$$\begin{aligned} \min \quad & d_1 + d_2 \\ \text{s.t.} \quad & w_1 - d_1 \leq 1 \\ & w_2 + d_2 \geq 2 \\ & 0 \leq w_1 \leq 2 \\ & 0 \leq w_2 \leq 1 \\ & d_1, d_2 \geq 0 \end{aligned}$$

(b) At solution $(w_1, w_2) = (1, 1)$, the first goal is satisfied ($d_1 = 0$) and the second misses by $d_2 = 2 - 1 = 1$. However, no feasible solution can make $w_2 > 1$, so the solution is optimal in the goal program.

Even though this solution is optimal in the goal program, it is not efficient. We may feasibly make the first objective less than 1.0 while retaining 1.0 in the second.

(c) The solution $(w_1, w_2) = (0, 1)$ is feasible and achieves the same minimum total deficiency as the solution of (b). But this alternative optimum is efficient.

Modified Goal Program Formulation to Assure Efficient Points

In many applications, model constraints and objectives are so complex that goal programming’s potential for producing non-efficient points never arises. Fortunately, there is an easy correction when it is a concern.

Principle 8.19 To assure that goal programming treatment of a multiobjective optimization yields an efficient point, we need only add a small positive multiple of each original minimize objective function to the goal program objective and subtract the same multiple of each original maximize.

We may illustrate with a modified Bank Three goal program (8.11). Efficient solution (8.13) results when we change the goal program objective to

$$\begin{aligned} \min \quad & d_1 + d_3 \\ & - 0.001(0.040x_2 + 0.045x_3 + 0.055x_4 + 0.070x_5 + 0.105x_6 \\ & + 0.085x_7 + 0.092x_8) + \frac{0.001}{20}(x_6 + x_7 + x_8) \end{aligned}$$

Multiple 0.001 of the maximize profit objective has been subtracted and the same multiple of the risk-asset ratio added to the standard deficiency objective.

For the same reason that weighted-sum approaches produce efficient solutions (principle 8.9), any result from this modified goal program can be optimal only if no original objective can be improved without degrading another. Still, the underlying goal program resolution of the multiple objectives will not be affected as long as the weight on original objectives is kept small.

EXAMPLE 8.10: MAKING GOAL PROGRAMS PRODUCE EFFICIENT SOLUTIONS

Produce a revised version of the goal program in Example 8.9(a) that must yield an efficient solution in the original multiobjective optimization.

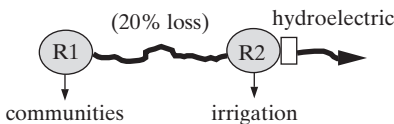
Solution: Following principle [8.19], we introduce a small multiple of the original objective functions. Using multiplier 0.001, the result is

$$\begin{aligned} \min \quad & d_1 + d_2 + 0.001(w_1) - 0.001(w_2) \\ \text{s.t.} \quad & w_1 - d_1 \leq 1 \\ & w_2 + d_2 \geq 2 \\ & 0 \leq w_1 \leq 2 \\ & 0 \leq w_2 \leq 1 \\ & d_1, d_2 \geq 0 \end{aligned}$$

The multiplier is positive for the first, minimize objective, and negative on the maximize objective. The goal program objective of minimizing deficiency is still dominant, but new terms now assure that alternative optima will resolve to efficient points.

EXERCISES

8-1 The sketch that follows shows the District 88 river system in the southwestern United States.



All water arises at mountain reservoir R1. The estimated flow is 294 million acre-feet. At least 24 million acre-feet of the water is contracted to go directly from R1 to nearby communities, but the communities would accept as much as could be supplied. The remainder flows through the desert to a second reservoir, R2, losing 20% to evaporation along the way. Some water at R2 can be allocated for irrigation of nearby farms. The remainder flows over a hydroelectric dam and passes downstream. To maintain the equipment, the flow over the dam must be at least 50 million acre-feet. District 88 sells water to communities at \$0.50 per acre-foot and to irrigation farmers at \$0.20 per acre foot. Water passing through the hydroelectric dam earns \$0.80 per acre-foot. The district would like to

maximize both water supplied for irrigation and sales income.

(a) Briefly explain how this problem can be modeled by the multiobjective linear program

$$\begin{aligned} \max \quad & x_2 \\ \max \quad & 0.50x_1 + 0.20x_2 + 0.80x_3 \\ \text{s.t.} \quad & x_1 + 1.25x_2 + 1.25x_3 = 294 \\ & x_1 \geq 24, x_2 \geq 0, x_3 \geq 50 \end{aligned}$$

(b) Use the class optimization software to sketch the efficient frontier for the model of part (a) in an objective value graph like Figure 8.4.

(c) Use the class optimization software to show that the optimal solutions taken separately for each objective do not coincide.

8-2 A semiconductor manufacturer has three different types of silicon wafers in stock to manufacture its three varieties of computer chips. Some wafer types cannot be used for some chips, but there are two alternatives for each chip. The table that follows shows the cost and on-hand supply of

each wafer type, the number of each chip needed, and a score (0 to 10) of the appropriateness match of each wafer type for making each chip.

Wafer Type	Chip Match			Unit Cost	On Hand
	1	2	3		
	1	7	8		
2	10	—	6	25	630
3	—	10	10	30	710
Need	440	520	380		

The company would like to minimize the total cost of the wafers used while maximizing the total match score.

(a) Briefly explain how this problem can be modeled by the multiobjective LP

$$\begin{aligned}
 \min \quad & 15x_{1,1} + 15x_{1,2} + 25x_{2,1} \\
 & \quad + 25x_{2,3} + 30x_{3,2} + 30x_{3,3} \\
 \max \quad & 7x_{1,1} + 8x_{1,2} + 10x_{2,1} \\
 & \quad + 6x_{2,3} + 10x_{3,2} + 10x_{3,3} \\
 \text{s.t.} \quad & x_{1,1} + x_{1,2} \leq 500 \\
 & x_{2,1} + x_{2,3} \leq 630 \\
 & x_{3,2} + x_{3,3} \leq 710 \\
 & x_{1,1} + x_{2,1} = 440 \\
 & x_{1,2} + x_{3,2} = 520 \\
 & x_{2,3} + x_{3,3} = 380 \\
 & \text{all } x_{i,j} \geq 0
 \end{aligned}$$

(b) and (c) as in Exercise 8-1.

8-3 A national commission is deciding which military bases to close in order to save at least \$85 million per year. The table that follows shows projected annual savings (millions of dollars) for each of five possible closings, together with the implied percent loss of military readiness and the number of civilian workers (thousands) who would lose their jobs.

	Base				
	1	2	3	4	5
Savings	24	29	45	34	80
Readiness	1.0	0.4	1.4	1.8	2.0
Workers	2.5	5.4	4.6	4.2	14.4

Each base must be left open or completely closed, and the commission wants to meet the required savings while minimizing both readiness loss and unemployment.

(a) Briefly explain how this base-closing problem can be modeled by the multiobjective ILP

$$\begin{aligned}
 \min \quad & 1.0x_1 + 0.4x_2 + 1.4x_3 \\
 & \quad + 1.8x_4 + 2.0x_5 \\
 \min \quad & 2.5x_1 + 5.4x_2 + 4.6x_3 \\
 & \quad + 4.2x_4 + 14.4x_5 \\
 \text{s.t.} \quad & 24x_1 + 29x_2 + 45x_3 \\
 & \quad - 34x_4 + 80x_5 \geq 85 \\
 & x_j = 0 \text{ or } 1, \quad j = 1, \dots, 5
 \end{aligned}$$

(b) and (c) as in Exercise 8-1.

8-4 A garden superstore sells 4 kinds of fertilizer from 20 pallet spaces aligned along a fence. Weekly demands for the 4 fertilizers are 20, 14, 9, and 5 pallets, respectively, and each pallet holds 10 bags. The sales counter is at the beginning of the fence, and every bag sold must be carried from its pallet to the counter. Whenever all pallets for a product are empty, the space allocated for that fertilizer is refilled completely using a fork truck from secondary storage. Store managers need to decide how many pallet spaces to assign each product to minimize both the number of times per week such refilling will be necessary and the total carrying distance of bags sold. Each fertilizer will have at least one pallet space.

(a) Assuming that fractional numbers of pallet spaces are allowed, briefly explain how this pallet allocation problem can be modeled by the multiobjective nonlinear program

$$\begin{aligned}
 \min \quad & 20/x_1 + 14/x_2 + 9/x_3 + 5/x_4 \\
 \min \quad & 200(x_1/2) + 140(x_1 + x_2/2) \\
 & \quad + 90(x_1 + x_2 + x_3/2) \\
 & \quad + 50(x_1 + x_2 + x_3 + x_4/2) \\
 \text{s.t.} \quad & \sum_{i=1}^4 x_j = 20 \\
 & x_{j \geq 1}, \quad j = 1, \dots, 4
 \end{aligned}$$

(b) and (c) as in Exercise 8-1.

8-5 Consider the multiobjective LP

$$\begin{aligned} \max \quad & x_1 + 5x_2 \\ \max \quad & x_1 \\ \text{s.t.} \quad & x_1 - 2x_2 \leq 2 \\ & x_1 + 2x_2 \leq 12 \\ & 2x_1 + x_2 \leq 9 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- ✓ (a) Show graphically that the optimal solutions taken separately for each objective do not coincide.
- ✓ (b) Determine graphically whether each of the following is an efficient point: (2,0), (4,7), (3,3), (2,5), (2,2), (0,6).
- ✓ (c) Use graphic solution to sketch the efficient frontier for this model in an objective value graph like Figure 8.4.

8-6 Do Exercise 8-5 for multiobjective LP

$$\begin{aligned} \min \quad & 5x_1 - x_2 \\ \min \quad & x_1 + 4x_2 \\ \text{s.t.} \quad & -5x_1 + 2x_2 \leq 10 \\ & x_1 + x_2 \geq 3 \\ & x_1 + 2x_2 \geq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

and points (4,0), (2,1), (4,4), (1,2), (5,0), (0,2).

8-7 Consider the multiobjective LP

$$\begin{aligned} \max \quad & 6x_1 + 4x_2 \\ \max \quad & x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 \leq 12 \\ & x_1 + 2x_2 \leq 10 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- ✓ (a) State and solve graphically a sequence of linear programs to compute a preemptive solution giving priority to the first objective. Also verify that the result is an efficient point.
- ✓ (b) State and solve graphically a sequence of linear programs to compute a preemptive solution giving priority to the second objective. Also verify that the result is an efficient point.

8-8 Do Exercise 8-7 for the multiobjective LP

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \min \quad & x_1 \end{aligned}$$

$$\begin{aligned} \text{s.t.} \quad & 2x_1 + x_2 \geq 4 \\ & 2x_1 + 2x_2 \geq 6 \\ & x_1 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

8-9 Combine each of the following sets of objective functions into a single weighted-sum objective of the specified maximize or minimize form, weighting individual objectives in the proportions indicated.

$$\begin{aligned} \text{✓ (a) } \min \quad & 3x_1 + 5x_2 - 2x_3 + 19x_4 \\ \max \quad & 17x_2 - 28x_4 \\ \min \quad & 34x_2 + 34x_3 \end{aligned}$$

Results form min, proportions 5:1:3.

$$\begin{aligned} \text{(b) } \max \quad & 20x_1 - 4x_2 + 10x_4 \\ \min \quad & 7x_2 + 9x_3 + 11x_4 \\ \max \quad & 23x_1 \end{aligned}$$

Results form max, proportions 3:1:1.

8-10 Return to the multiobjective linear program of Exercise 8-7.

- ✓ (a) Solve graphically with a weighted-sum objective that weights the first objective twice as heavily as the second, and verify that the result is an efficient point.
- ✓ (b) Solve graphically with a weighted-sum objective that weights the second objective twelve times as heavily as the first, and verify that the result is an efficient point.

8-11 Do Exercise 8-10 on the multiobjective linear program of Exercise 8-8, weighting the second objective twice the first.

8-12 Convert each of the following multiobjective optimization models to a goal program seeking the specified target levels, and minimizing the unweighted sum of all goal violations.

$$\begin{aligned} \text{✓ (a) } \min \quad & 3x_1 + 5x_2 - x_3 \\ \max \quad & 11x_2 + 23x_3 \\ \text{s.t.} \quad & 8x_1 + 5x_2 + 3x_3 \leq 40 \\ & x_2 - x_3 \leq 0 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Targets 20,100.

$$\begin{aligned} \text{(b) } \min \quad & 17x_1 - 27x_2 \\ \max \quad & 90x_2 + 97x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 100 \\ & 40x_1 + 40x_2 - 20x_3 \geq 8 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Targets 500, 5000.

(c) $\max 40x_1 + 23x_2$
 $\min 20x_1 - 20x_2$
 $\min 5x_2 + x_3$
 s.t. $x_1 + x_2 + 5x_3 \geq 17$
 $40x_1 + 4x_2 + 33x_3 \leq 300$
 $x_1, x_2, x_3 \geq 0$

Targets 700, 25, 65.

(d) $\max 12x_1 + 34x_2 + 7x_3$
 $\min x_2 - x_3$
 $\min 10x_1 + 7x_3$
 s.t. $5x_1 + 5x_2 + 15x_3 \leq 90$
 $x_2 \leq 19$
 $x_1, x_2, x_3 \geq 0$

Targets 600, 20, 180.

(e) $\min 22x_1 + 8x_2 + 13x_3$
 $\max 3x_1 + 6x_2 + 4x_3$
 s.t. $5x_1 + 4x_2 + 2x_3 \leq 6$
 $x_1 + x_2 + x_3 \geq 1$
 $x_1, x_2, x_3 = 0$ or 1

Targets 20, 12.

(f) $\min 4x_2 + \ln(x_2) + x_3 + \ln(x_3)$
 $\max (x_1)^2 + 9(x_2)^2 - x_1x_2$
 s.t. $x_1 + x_2 + x_3 \leq 10$
 $4x_2 + x_3 \geq 6$
 $x_1, x_2, x_3 \geq 0$

Targets 20, 40.

8-13 Consider the multiobjective LP

$\max x_1$
 $\max 2x_1 + 2x_2$
 s.t. $2x_1 + x_2 \leq 9$
 $x_1 \leq 4$
 $x_2 \leq 7$
 $x_1, x_2 \geq 0$

- (a) Sketch the feasible space and contours where the first objective equals 3 and the second equals 14.
- (b) Formulate a corresponding goal program seeking target levels 3 and 14 on the two objectives and minimizing the unweighted sum of goal violations.
- (c) Explain why $\mathbf{x} = (2, 5)$ is optimal in the goal program of part (b) by reference to the plot of part (a).

8-14 Do Exercise 8-13 on the multiobjective LP

$\min x_2$
 $\max 5x_1 + 3x_2$

s.t. $2x_1 + 3x_2 \geq 6$
 $x_1 \leq 5$
 $-x_1 + x_2 \leq 2$
 $x_1, x_2 \geq 0$

using target levels 1 and 30 and solution $\mathbf{x} = (5, 5/3)$.

8-15 Return to the multiobjective LP of Exercise 8-13 with target levels 3 and 14 for the two objectives.

- (a) State and solve graphically [in (x_1, x_2) space] a sequence of linear programs to compute a preemptive goal programming solution, prioritizing objectives in the sequence given and using the given target levels.
- (b) Formulate a single goal program that implements the preemptive goal programming process of part (a) by suitable deficiency variable weighting, and demonstrate graphically that it produces the same solution.
- (c) Determine whether your solution in part (a) is an efficient point of the original multiobjective model. Could the outcome have been different if the target for the first objective were revised to 1? Explain.

8-16 Return to the multiobjective LP of Exercise 8-14. Do Exercise 8-15 on the model using target levels of 1 and 30 and a revised objective 1 target of 3.

8-17 Return to the multiobjective District 88 water allocation problem of Exercise 8-1.

- (a) State and solve with class optimization software a sequence of linear programs to compute a preemptive solution for the model of part (a) taking objectives in the order given.
- (b) State and solve with class optimization software a weighted-sum objective LP that weights the first objective twice as heavily as the second.
- (c) Suppose that district management sets a goal of at least 100 million acre-feet for irrigation and \$144 million in income. Convert the model of Exercise 8-1(a) to a goal program seeking these target levels and minimizing the unweighted sum of goal violations.
- (d) Use class optimization software and your goal program of part (c) to show that all

goals cannot be fulfilled simultaneously by any feasible solution.

- ☑ (e) Solve the model of part (c) with class optimization software and compare to results for earlier methods.
- ☑ (f) State and solve with class optimization software a sequence of linear programs to compute a preemptive goal programming solution for the model of part (c) taking objectives in the order given.
- ☑ (g) Formulate an alternative objective function in the goal program of part (c) that accomplishes the preemptive goal programming optimization of part (f) in a single step.

8-18 Do Exercise 8-17 on the silicon wafer problem of Exercise 8-2 using target levels of 30,000 for cost and 13,000 for match.

☑ **8-19** Do Exercise 8-17 on the base closing problem of Exercise 8-3 using target levels of 3% for readiness loss and 12,000 for displaced workers.

8-20 Do Exercise 8-17 on the pallet allocation problem of Exercise 8-4 using target levels of 10 for refills and 2500 for walking.

8-21 Professor Proof is reconsidering where to invest the \$50,000 left in his retirement account. One available bond fund with estimated return of 5% per year has a published risk rating of 20. The other option is a hedge fund with estimated return of 15% per year but a risk rating of 80 (it may be a ponzi scheme). Proof wants to divide his entire account between the two, placing no more than \$40,000 in either fund. Within these limits he would like to both maximize total annual return and minimize the weighted (by investment size) average of the risk ratings.

- (a) Formulate Professor Proof's problem as a multiobjective linear program with one main constraint, two upper bounds, and two nonnegativities. Be sure to define all decision variables and annotate objectives and constraints to show their meanings.
- (b) Now show how to modify your formulation of (a) as a goal program with targets of \$6,000 annual return and a weighted risk average of 50. Weight shortfalls from the targets equally in the objective. Define all new decision variables and annotate new objectives and constraints to show their meanings.

☐ (c) Enter and solve your goal program of (b) with class optimization software.

8-22 A major auto manufacturer is planning the marketing campaign to introduce its new Bambi model of hybrid sport utility vehicle. Bambis will be introduced with a television advertising campaign totaling \$200 million. The table below shows the 3 kinds of programs being considered for placement of the ads, including the cost per 30-second (unit) commercial and the number of viewers exposed in each of the market segments. No more than 50% of the budget should be spent on any one type of program. Within these limits the company has set goals of 3,000 million under-30 exposures from the campaign, 5,000 million age 30–55 male exposures, and 3,400 million age 30–55 female exposures.

Program Type	Unit Cost (\$ million)	Exposures per Unit (millions)		
		Under 30	30–55 Male	30–55 Female
Desperate Housewives	2.2	24	12	37
Law and Order	2.5	27	42	11
Comedy Central	0.5	19	16	6

- (a) Formulate a Linear Goal Program to help the manufacturer decide how to optimally place their ads. Weight all goals equally. Be sure to define all decision variables, and annotate objectives and constraints to indicate their meaning.
- (b) Obviously numbers of commercial showings in (a) must be integer, but it is good OR modeling to treat them as continuous. Briefly explain why.
- ☐ (c) Enter and solve the your goal of (a) with class optimization software.

8-23 The State Highway Patrol (SHP) has only 60 officers to patrol $j = 1, \dots, 10$ major highway segments in rural and suburban areas of the state. Officers work $i = 1, \dots, 7$ different shift patterns across a 24 hour, 7 day work week. Exceptions can be made on late-night shifts $i = 1$ and 2, but all other shifts require at least 1 officer on each highway segment, and no segment ever receives more than 3.

An SHP goal is to concentrate the patrol resources in times and places of greatest need. One measure of this is $c_{i,j} \triangleq$ the traffic density on segment j during shift i . A second is $a_{i,j} \triangleq$ the rela-

tive accident rate on segment j during shift i . The following table shows values derived from history for both across shifts and segments.

Shift i		Highway Segments j									
		1	2	3	4	5	6	7	8	9	10
1	Cong	0.22	0.26	1.11	1.06	1.80	2.16	1.93	0.98	0.66	0.45
	Accid	0.92	2.16	1.18	1.49	0.90	4.11	1.10	1.15	2.18	0.77
2	Cong	0.32	0.36	1.31	1.26	1.90	2.26	2.03	1.05	0.86	0.55
	Accid	0.98	2.66	1.48	1.69	1.10	3.91	1.12	1.17	2.48	0.77
3	Cong	0.55	0.66	1.81	1.86	2.20	2.86	2.43	1.95	1.06	0.85
	Accid	0.88	3.16	1.88	1.89	1.80	4.11	1.62	1.67	2.88	1.07
4	Cong	0.65	0.76	1.91	1.96	2.30	2.96	2.53	2.05	1.26	0.95
	Accid	0.77	2.01	1.03	1.31	1.10	3.11	1.00	1.05	2.08	0.67
5	Cong	0.60	0.70	1.83	1.82	2.22	2.79	2.45	2.00	1.16	0.90
	Accid	0.90	2.76	1.38	1.52	0.98	3.81	1.40	1.45	2.48	0.87
6	Cong	0.62	0.68	1.85	1.82	2.25	2.74	2.40	2.02	1.18	0.94
	Accid	1.18	4.16	2.48	2.59	2.80	4.61	2.62	1.87	3.28	1.09
7	Cong	0.42	0.48	1.55	1.52	1.95	2.14	2.10	1.92	1.08	0.64
	Accid	1.08	5.16	2.28	2.19	2.20	4.11	3.22	1.67	3.58	1.19

SHP would like to decide how to feasibly allocate its officers to shifts and segments to maximize total coverage of congested shifts and segments, as well as total coverage of high accident shifts and coverage.

- (a) Formulate the problem as a multi-objective ILP over nonnegative integer decision variables $x_{i,j} \triangleq$ the number of officers assigned to each shift i and segment j .
- (b) Show how to modify your model of (a) as a goal ILP with targets for total congestion = 120 and for total accident level = 150. Weight under-satisfaction of the two goals equally, define all new decision variables, and annotate new objectives and constraints to show their meanings.
- (c) Enter and solve your goal ILP of (b) with class optimization software.

8-24 An architect is trying to decide what mix of single, double, and luxury rooms to include in a new hotel. Only \$10 million is available for the project, and single rooms cost \$40,000 to build, double rooms \$60,000, and luxury rooms \$120,000 each. Business

travelers (in groups of one or more) average 0.7 of single-room rentals, 0.4 of double-room rentals, and 0.9 of luxury room rentals. Family travelers occupy the remainder of each category. The architect would like her design to accommodate 100 total rooms for business travelers, but she would also like it to provide 120 total rooms for family travelers.

- (a) Formulate a weighted goal LP to decide how many of each room to include in the design. Weight the business traveler goal twice as heavily as the family traveler goal.
- (b) Enter and solve your goal program using class optimization software.

8-25 Trustees of a major state university are trying to decide how much tuition⁴ should be charged next year from four categories of full-time students: in-state undergraduates, in-state graduates, out-of-state undergraduates, and out-of-state graduates. The following table shows present tuition levels (thousands of dollars) for each category, together with projected enrollments (thousands) for next year and estimates of the true university cost to educate students.

⁴Based on A. G. Greenwood and L. J. Moore (1987), "An Inter-temporal Multi-goal Linear Programming Model for Optimizing University Tuition and Fee Structures," *Journal of the Operational Research Society*, 38, 599-613.

	Undergrad	Grad
In-state tuition	4	10
Out-of-state tuition	12	15
In-state enrollment	20	1.5
Out-of-state enrollment	10	4
True cost	20	36

New tuitions must raise \$292 million, but trustees would also like in-state tuitions to recover at least 25% of true cost and out-of-state, 50%. No tuition will be reduced from its present level, but the trustees would also like to assure that no tuition category increases by more than 10%.

- (a) Formulate a goal LP to compute suitable tuitions giving priority to the 10% increase goals over cost recovery.
- ☑ (b) Enter and solve your model with class optimization software.

8-26 A university library⁵ must cut annual subscription expenses s_j to some or all scientific journals $j = 1, \dots, 40$ to absorb a \$5000 per year budget cut. One consideration will be the sum of published counts c_j of the number of times other journals cite papers in journal j , which is a measure of how seminal a journal is to research. Another is the sum of usefulness ratings r_j (1 = low to 10 = high) solicited from university faculty. Finally, the library wants to consider the sum of ratings a_j of the relative availability (1 = low to 8 = high) in nearby libraries, believing that journals readily available elsewhere need not be retained.

- (a) Formulate a multiobjective ILP to choose journals to drop.
- (b) Convert your multiobjective model to a goal program with targets at most C for total citations, at most R for total faculty ratings, and at least A for total availability ratings. Weight all goals equally.

8-27 The household sector is the largest consumer of energy in India,⁶ requiring at least 10^8 kilowatt-hours per day in a large city such as Madras.

This energy can be obtained from sources $j = 1$ kerosene, $j = 2$ biogas, $j = 3$ photovoltaic, $j = 4$ fuelwood-generated electricity, $j = 5$ biogas-generated electricity, $j = 6$ diesel-generated electricity, and $j = 7$ electricity from the national power grid, with efficiencies η_j expressing the number of kilowatt-hours per unit of original fuel input of source j . Biogas is limited 1.3×10^9 units. Energy planners are looking for the best mix of these sources to meet the 10^8 requirement in terms of a variety of conflicting objectives. One is low total cost at p_j per unit source j . Another is to maximize local employment, which is estimated to grow at e_j per unit of energy source j . Finally, there are three types of pollution to be minimized: carbon oxides produces at c_j per unit of source j , sulfides at s_j per unit of j , and nitrogen oxides at n_j per unit of j .

- (a) Formulate a multiobjective LP to choose a best mix of sources.
- (b) Revise your model as a goal program with maximum targets P , C , S , and N for cost, carbon, sulfur, and nitrogen, together with minimum target E for employment.

8-28 The table that follows lists tasks like those required of the U.S. Food and Drug Administration (FDA) in carrying out its responsibilities to regulate drug laboratories, along with an importance weighting of each task and the estimated minimum number of staff hours required to accomplish the task fully.

j	Task	Import	Hours
1	Laboratory quality control program	10	1500
2	Analytical protocol	1	100
3	Drug protocol	2	50
4	Quality assurance guidelines	5	55
5	Safety programs	10	135
6	Stability of testing	5	490
7	Calibration of equipment	4	2000
Total			4330

⁵Based on M. J. Schniederjans and R. Santhanam (1989), "A Zero-One Goal Programming Approach for the Journal Selection and Cancellation Problem," *Computers and Operations Research*, 16, 557-565.

⁶Based on R. Ramanathan and L. S. Ganesh (1995), "Energy Alternatives for Lighting in Households: An Evaluation Using an Integrated Goal Programming-AHP Model," *Energy*, 20, 66-72.

Unfortunately only 3600 of the 4330 total hours are available to execute all tasks. FDA management seeks an allocation of the 3600 hours that addresses the goal hours in the table on the basis of minimizing total importance of undersatisfaction. Formulate a goal LP to determine the most appropriate allocation.

8-29 The Lake Lucky dam⁷ retains water in the reservoir lake of the same name to prevent flooding downstream and to assure a steady year-round supply of water for wildlife and nearby cities. The best available projections call for inflows i_t (cubic meters of water) to arrive at the reservoir on upcoming days $t = 1, \dots, 120$. A minimum of at least \underline{r} must be released through the dam every day to maintain downstream water quality, but the daily release can go as high as \bar{r} . The current storage in the reservoir is s_0 cubic meters, and storage levels must be kept between minimum and maximum limits \underline{s} and \bar{s} throughout the 120-day period. Reservoir managers seek a 120-day plan that releases, as close as possible to target, R cubic meters each day and maintains a target level of S cubic meters stored in the reservoir. Both below- and above-target outcomes are considered equally bad, and release deviations are weighted equally with those for storage. Formulate a goal LP to compute a reservoir operating plan. Assume that no water is lost (i.e., all water that enters the reservoir eventually flows over the dam).

8-30 A retail store has hired a marketing consultant to help it decide how to evaluate the quality⁸ of its service. The store would like to evaluate customer perceptions about $i = 1$, employee attitude; $i = 2$, employee competence; $i = 3$, product quality; and $i = 4$, sensory (look, sound, smell) appeal of the store. The consultant is considering 6 survey instruments for the task: $j = 1$, point-of-purchase cards measuring employee service delivery; $j = 2$, point-of-purchase cards measuring product quality; $j = 3$, open-ended comment cards; $j = 4$, a focus group of known customers;

and $j = 5$, a telephone survey of known customers. The instruments are not equally useful in measuring the desired quality attributes, so the consultant has developed (1 = poor to 10 = excellent) ratings $r_{i,j}$ scoring the value of instrument j in assessing attribute i . Instruments j would cost c_j dollars and require h_j hours of store employee time. Ideally, the instruments chosen should total at least 30 points on each rated attribute, but their total cost must not exceed \$10,000, and the total employee hours invested must not be more than 500. Formulate a goal ILP to choose a suitable combination of instruments.

8-31 A steel workpiece is to be cut⁹ in a lathe to a target depth of 0.04 inch. Empirical studies with this type of steel express the properties of the resulting product as

$$\begin{aligned} \text{finish} &\triangleq 0.41v^{3.97}f^{3.46}d^{0.91} \\ \text{power} &\triangleq vf^{0.75}d^{0.90} \\ \text{time} &\triangleq vf \end{aligned}$$

where v is the cutting speed in revolutions per minute, f is the feed rate in inches per revolution, and d is the depth of cut in inches. Finish variation must be kept no greater than 150, required power no more than 4.0, cutting speed between 285 and 680, and feed rate between 0.0075 and 0.0104. Within these limits, the first priority goal is to make the depth of cut as close as possible to 0.04 inch, and the second is to complete the cut in at most time 1.5 (minutes).

- (a) Formulate a preemptive goal NLP to choose cutting parameters.
- (b) Enter and (at least locally) solve your goal program using class optimization software.

8-32 Trees in available wild forest¹⁰ stands $j = 1, \dots, 300$ have been measured in terms of desirable traits $i = 1, \dots, 12$ such as rate of growth, disease resistance, and wood density. Computations of averages and standard deviations in traits over

⁷Based on K. K. Reznicek, S. P. Simonovic, and C. R. Bector (1991), "Optimization of Short-Term Operations of a Single Multipurpose Reservoir—A Goal Programming Approach," *Canadian Journal of Civil Engineering*, 18, 397–406.

⁸Based on M. J. Schniederjans and C. M. Karuppan (1995), "Designing a Quality Control System in a Service Organization: A Goal Programming Case Study," *European Journal of Operational Research* 81, 249–258.

⁹Based on R. M. Sundaram (1978), "An Application of Goal Programming Technique in Metal Cutting," *International Journal of Production Research*, 16, 375–382.

¹⁰Based on T. H. Mattheiss and S. B. Land (1984), "A Tree Breeding Strategy Based on Multiple Objective Linear Programming," *Interfaces*, 14:5, 96–104.

the stands then produced numbers $z_{i,j}$ of standard deviations that stand j falls above or below the average for trait i . Corresponding scores for the offspring of a planned breeding program may be assumed to combine linearly (i.e., the result will be a sum of scores for the wild stands used, weighted by the proportion each stand makes up of the breeding population).

- (a) Formulate a multiobjective LP to choose a breeding population that maximizes the offspring standard deviation score of each trait.
- (b) Convert your model to a goal program with a target of breeding offspring 2 standard deviations above the mean on all traits. Weight the undersatisfaction of each goal equally.

8-33 Soar, a retailer catering to upscale suburbanites,¹¹ wishes to be represented in all the major malls $j = 1, \dots, 5$ in the Atlanta region. However, Soar wants to divide its investment budget b in proportion to three major measures of mall attractiveness: weekly patronage, p_j ; average patron annual income, a_j ; and number of large anchor stores, s_j . That is, they would like ratios of investment in j to p_j all to be equal, and similarly for corresponding ratios to the a_j and s_j . Formulate a goal LP model to choose an appropriate allocation, weighting all deviations equally.

8-34 Shipley Company¹² is designing a new photoresist, which is a chemical coating used in the photoengraving of silicon chips. Important characteristics of photoresists are $i = 1$, high flow (liquefaction) temperature; $i = 2$, low minimum line-space resolution; $i = 3$, retention of unexposed areas when exposed circuits are developed away at least \underline{b}_3 ; $i = 4$, photospeed (minimum exposure time to create an image) between \underline{b}_4 and \bar{b}_4 ; and $i = 5$, exposure energy requirements between \underline{b}_5 and \bar{b}_5 . Ingredients $j = 1, \dots, 24$ produce highly nonlinear effects on these five important characteristics. However, designed

experiments have demonstrated that the effects can be assumed linear within ingredient quantity ranges $[l_j, u_j]$ under consideration. Between these limits, each additional unit of ingredient j adds $a_{i,j}$ to characteristic i of the photoresist.

- (a) Formulate a 2-objective LP model to choose the best composition for both flow temperature and line-space resolution.
- (b) Explain the meaning of an efficient point solution to your model in part (a), and describe how one could be computed.

8-35 South African wildlife management officials¹³ must decide annual quotas (in thousands of tons) for harvesting of pilchards, anchovies, and other fish of the pelagic family in the rich waters to the west of the Cape of Good Hope. One consideration is the maximization of fishing industry income, which is estimated at 110 rand/ton for pilchards, 30 rand/ton for anchovies, and 100 rand/ton for other pelagic. Still, the harvest should leave the year-end populations of the 3 types of fish as large as possible. Scientists estimate that initial biomasses of the 3 types are 140,000 tons, 1,750,000 tons, and 500,000 tons, respectively. Due to differences in breeding characteristics, each thousand tons of pilchards harvested will deplete the population by 0.75 thousand tons, each of anchovies by 1.2 thousand tons, and each of other pilchards by 1.5 thousand tons. A final consideration is maintaining the ocean ecosystem. Important measures are the number of breeding characteristics, each thousand tons of pilchards harvested will deplete the population by 0.75 thousand tons, each of anchovies by 1.2 thousand tons, and each of other pilchards by 1.5 thousand tons. A final consideration is maintaining the ocean ecosystem. Important measures are the number of breeding pairs of penguins (thousands), which is estimated as $70 + 0.60$ (ending biomass of pilchards), and the number of such pairs of gannets (thousands), which is estimated at $5 + 0.20$ (ending biomass of pilchards) $+ 0.20$ (ending biomass of anchovies).

¹¹Based on R. Khorramshahgol and A. A. Okoruwa (1994), "A Goal Programming Approach to Investment Decisions: A Case Study of Fund Allocation Among Different Shopping Malls," *European Journal of Operational Research*, 73, 17–22.

¹²Based on J. S. Schmidt and L. C. Meile (1989), "Taguchi Designs and Linear Programming Speed New Product Formulation," *Interfaces*, 19:5, 49–56.

¹³Based on T. J. Stewart (1988), "Experience with Prototype Multicriteria Decision Support Systems for Pelagic Fish Quota Determination," *Naval Research Logistics*, 35, 719–731.

- (a) Formulate a multiobjective LP to maximize all measures of success using the decision variables ($i = 1$, pilchards; $i = 2$, anchovies; $i = 3$, other pelagic)

$$x_i \triangleq \text{quota for fish type } i \text{ (thousand tons)}$$

$$y_i \triangleq \text{ending biomass of fish type } i \text{ (thousand tons)}$$

- (b) Revise your formulation as a goal program with minimum targets for the 6 objectives being 38 million rands income, ending biomasses of 100, 1,150,000 and 250,000 tons, and ending breeding populations of 130,000 and 25,000 pairs. Give preemptive priority to maintaining the populations of the 3 types of fish.

- (c) Enter and solve your goal program version using class optimization software.

8-36 The state of Calizona will soon be opening centers¹⁴ in 12 of its county seats $i = 1, \dots, 100$ to promote solid waste recycling programs. Each selected site will service a district of surrounding counties. The state wants to minimize the sum of distances $d_{i,j}$ populations p_i in counties i must travel to their district center at j . But they would also like to come as close as possible (in total deviation) to having 3 centers in each of the 4 Environment Department regions. Counties $i = 1, \dots, 12$ make up region 1, $i = 13, \dots, 47$ region 2, $i = 48, \dots, 89$ region 3, and $i = 90, \dots, 100$ region 4.

- (a) Formulate a 2-objective ILP model of this facility location problem using the main decision variables ($i, j = 1, \dots, 100$)

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if county } i \text{ is served by} \\ & \text{a center at } j \\ 0 & \text{otherwise} \end{cases}$$

$$x_i \triangleq \begin{cases} 1 & \text{if country } j \text{ gets a center} \\ 0 & \text{otherwise} \end{cases}$$

- (b) Explain the meaning of an efficient point solution to your model of part (a) and describe how one could be computed.

8-37 Every year the U.S. Navy must plan the reassignment of thousands of sailors finishing

one tour of duty in specialties $i = 1, \dots, 300$ and preparing for their next one, k . The reassignment may also require a move of the sailor and his or her dependents from present base location $j = 1, \dots, 25$ to a new base ℓ . To plan the move, personnel staff estimate numbers $s_{i,j}$ of sailors at base j in specialty i who are ready for reassignment, $d_{k,\ell}$ of sailors needed at base ℓ in specialty k , as well as average costs $c_{j,\ell}$ of moving a sailor and dependents from base j to base ℓ . Training is required for the new position if it involves a different specialty than the present one, and capacity limits u_k at training schools limit the number that can be trained for any specialty k during the year. One Navy objective in planning reassignment is to keep total relocation costs as low as possible. However, other considerations arise from the fact that full staffing levels $d_{k,\ell}$ can almost never all be met. Naturally, the Atlantic fleet (bases $j, \ell = 1, \dots, 15$) wishes to maximize the fraction of its slots actually filled, and the Pacific fleet (bases $j, \ell = 16, \dots, 25$) prefers to maximize the fraction of its needs accommodated. Formulate a 3-objective LP model to aid the Navy in planning reassignments using the decision variables

$$x_{i,j,k,\ell} \triangleq \text{number of sailors presently in specialty } i \text{ at base } j \text{ reassigned to specialty } k \text{ at location } \ell$$

8-38 The sales manager of an office systems distributor¹⁵ has sales representatives $i = 1, \dots, 18$ to assign to both current accounts $j = 1, \dots, 250$ and the development of new accounts. Each current account will be assigned at most one representative. From prior experience, the manager can estimate monotone-increasing nonlinear functions

$$r_{i,j}(w_{i,j}) \triangleq \text{current-period sales revenue that will result from allocating } w_{i,j} \text{ hours of representative } i \text{ time to account } j$$

$$a_i(x_i) \triangleq \text{present value of future sales revenue that will derive from assigning representative } i \text{ to } x_i \text{ hours of new account development}$$

Each representative has 200 hours to divide between his or her assignments during the

¹⁴Based on R. A. Gerrard and R. L. Church (1994), "Analyzing Tradeoffs between Zonal Constraints and Accessibility in Facility Location," *Computers and Operations Research*, 21, 79–99.

¹⁵Based on A. Stam, E. A. Joachimsthaler, and L. A. Gardiner (1992), "Interactive Multiple Objective Decision Support for Sales Force Sizing and Deployment," *Decision Sciences*, 23, 445–466.

planning period, but the manager would like to maximize both total current revenue and total new account value.

- (a) Formulate a 2-objective INLP to determine an allocation plan using the decision variables ($i = 1, \dots, 18, j = 1, \dots, 250$)

$w_{i,j} \triangleq$ hours spent by representative i on account j

$x_i \triangleq$ hours spent by representative i on new account development

$$y_{i,j} \triangleq \begin{cases} 1 & \text{if representative } i \text{ is assigned} \\ & \text{to account } j \\ 0 & \text{otherwise} \end{cases}$$

- (b) Explain the meaning of an efficient point solution to your model of part (a) and describe how one could be computed.

REFERENCES

Collette, Yann and Patrick Siarry (2004), *Multiobjective Optimization: Principles and Case Studies (Decision Engineering)*, Springer-Verlag, Berlin, Germany.

Goichoechea, Ambrose, Don R. Hansen, and Lucien Duckstein (1982), *Multiobjective Decision*

Analysis with Engineering and Business Applications, Wiley, New York, New York.

Winston, Wayne L. (2003), *Operations Research - Applications and Algorithms*, Duxbury Press, Belmont, California.

This page intentionally left blank

Shortest Paths and Discrete Dynamic Programming

From our earliest discussions in Chapters 1 and 2, we have seen a trade-off between generality of operations research models and tractability of their analysis. The more specialized the model form and the longer the list of underlying assumptions, the richer and more efficient the analysis that is possible.

In this chapter we extend that insight to a new extreme by introducing the most specialized and thus most efficiently solved of all broad classes of optimization models: **shortest paths and discrete dynamic programs**. Dynamic programming methods proceed by viewing the problem or problems we really want to solve as members of a family of closely related optimizations. When model forms are specialized enough to admit just the right sort of family, strong connections among optimal solutions to the various members can be exploited to organize a very efficient search.

9.1 SHORTEST PATH MODELS

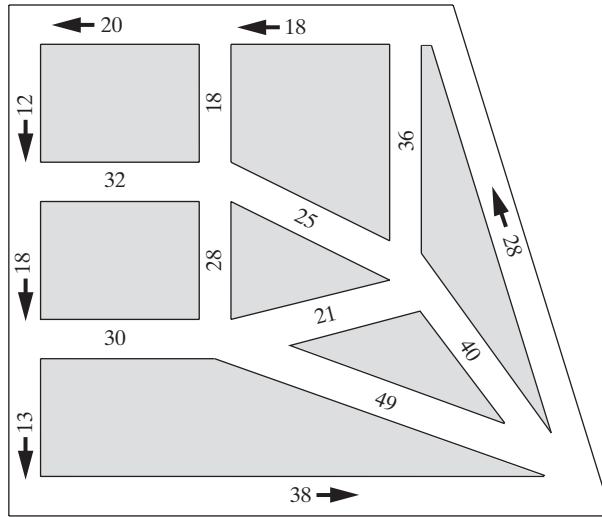
Whether in urban traffic, college hallways, satellite communications, or the surface of a microchip, it makes sense to follow the shortest route. Such **shortest path** problems are the first we will attack.

APPLICATION 9.1: LITTLEVILLE

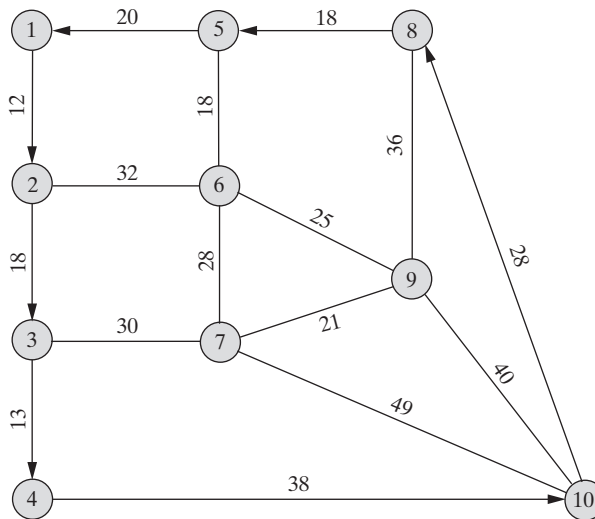
As usual, it will help to begin with an application setting. Suppose that you are the city traffic engineer for the town of Littleville. Figure 9.1(a) depicts the arrangement of one- and two-way streets in a proposed improvement plan for Littleville's downtown, including the estimated average time in seconds that a car will require to transit each block.

From survey and other data we can estimate how many driver trips originate at various origin points in the town, and the destination for which each trip is bound. But such survey data cannot indicate what streets will be selected by motorists to move from origin to destination in a street network that does not yet exist.

One of the tasks of a traffic engineer is to project the route that drivers will elect, so that city leaders can evaluate whether flows will concentrate where they



(a) Proposed street network



(b) Corresponding graph

FIGURE 9.1 Littleville Shortest Path Application

hope. A good starting point is to assume that drivers will do the most rational thing—follow the shortest time path from their origin to their destination. We need to compute all such shortest paths.

Nodes, Arcs, Edges, and Graphs

Figure 9.1(b) shows the first step. We abstract the given flow pattern—here a street system—into a graph or network.

These are not the newspaper sort of graphs comparing bars, or zigzag lines, or piles of little coins.

Definition 9.1 Mathematical **graphs** model travel, flow, and adjacency patterns in a **network**.

Such graphs begin with a collection of nodes (or vertices).

Definition 9.2 The **nodes** or **vertices** of a graph represent entities, intersections, and transfer points of the network.

Pairs of nodes may be linked in a graph by either **arcs** or **edges**.

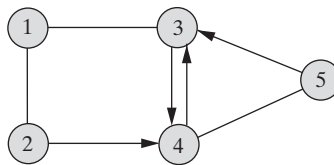
Definition 9.3 The **arcs** of a graph model available directed (one-way) links between nodes. **Edges** represent undirected (two-way) links.

In Figure 9.1(b) nodes represent intersections. For convenience we have numbered the nodes 1 to 10. One-way streets between intersections of the Littleville network yield arcs in our graph; two-way streets produce edges.

We denote arcs or edges by naming their terminal nodes. Thus (5, 1) and (3, 4) are arcs in Figure 9.1(b), while (7, 9) and (5, 6) are edges. Order matters for arcs, so that it would be incorrect, for example, to call the arc from node 10 to node 8 arc (8, 10). It must be (10, 8). There is more flexibility in referencing edges. An edge between nodes i and j could be called either (i, j) , or (j, i) , or both, although it is more common to name the smallest node number first. For example, we could refer to the edge between nodes 2 and 6 of Figure 9.1(b) as either (2, 6) or (6, 2), but (2, 6) would be more standard.

EXAMPLE 9.1: IDENTIFYING ELEMENTS OF A GRAPH

Consider the following graph:



Identify its nodes, arcs, and edges.

Solution: The node set of this graph is $V \triangleq \{1, 2, 3, 4, 5\}$. Arcs form $A \triangleq \{(2, 4), (3, 4), (4, 3), (5, 3)\}$. Edges are $E \triangleq \{(1, 2), (1, 3), (4, 5)\}$.

Paths

Our interest in this chapter is paths.

Definition 9.4 A **path** is a sequence of arcs or edges connecting two specified nodes in a graph. Each arc or edge must have exactly one node in common with its predecessor in the sequence, any arcs must be passed in the forward direction, and no node may be visited more than once.

Two of several paths from node 3 to node 8 in the Littleville application are illustrated in Figure 9.2(a). One proceeds 3–7–10–8, using edges (3, 7) and (7, 10), plus arc (10, 8). Another is 3–4–10–8, following arc sequence (3, 4), (4, 10), and (10, 8). The pattern 3–7–6–5–8 of Figure 9.2(b) is not a path because it transits arc (8, 5) in the wrong direction. Sequence 3–7–6–9–7–10–8 also fails the definition of a path; it repeats node 7.

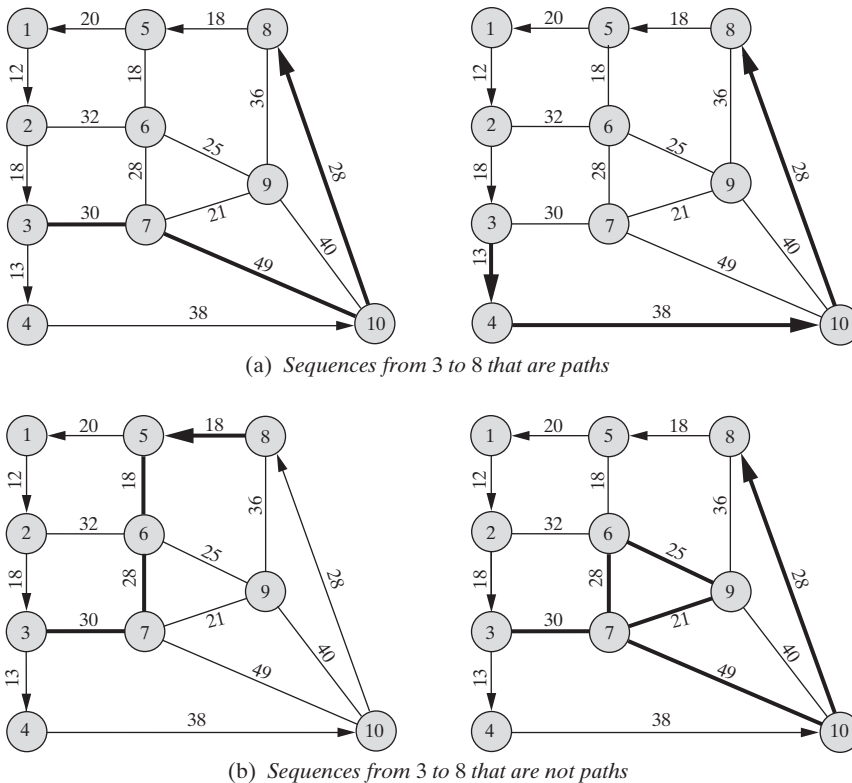


FIGURE 9.2 Paths of the Littleville Application

EXAMPLE 9.2: RECOGNIZING PATHS

Return to the graph of Example 9.1. Identify all paths from node 1 to node 5.

Solution: The two available paths are 1–2–4–5 and 1–3–4–5. Sequence 1–3–5 is not a path because it violates direction on arc (3, 5).

Shortest Path Problems

When an application includes costs or lengths on the arcs and edges of a graph, we are confronted with an optimization.

Definition 9.5 | **Shortest path problems** seek minimum total length paths between specified pairs of nodes in a graph.

In the Littleville application of Figure 9.1, lengths are travel times in seconds. The first path from node 3 to node 8 shown in Figure 9.2(a) totals $(30 + 49 + 28) = 107$ seconds in length. The shortest path from 3 to 8 is the second 3–4–10–8 sequence with length $(13 + 38 + 28) = 79$ seconds.

Classification of Shortest Path Models

Shortest path problems arise both as main decision questions and as steps in other computations. There are many variations, depending on the type of network and costs involved, and the pairs of nodes for which we need solutions.

We will see in the sections to come that shortest path algorithms are highly specialized to exploit particular properties. Thus it is important to distinguish cases.

Our Littleville application illustrates one combination. The graph has both arcs and edges, all link lengths are nonnegative, and our traffic engineering task requires shortest paths between all pairs of nodes. To summarize:

- **Name:** Littleville
- **Graph:** arcs and edges
- **Costs:** nonnegative
- **Output:** shortest paths
- **Pairs:** all nodes to all others

Before passing to algorithms, we introduce some examples illustrating other possibilities.

APPLICATION 9.2: TEXAS TRANSFER

Figure 9.3 displays a map of highway links between several major cities in Texas. Numbers on the edges in Figure 9.3 show standard driving distance in miles.

Texas Transfer, a major trucker in the southwest, ships goods from its hub warehouse in Ft. Worth to all the cities shown. Trucks leave the hub and proceed directly to their destination city, with no intermediate dropoffs or pickups.

Texas Transfer drivers are allowed to choose their own route from Ft. Worth to their destination. However, management's proposal in current labor negotiations calls for drivers to be paid on the basis of shortest standard mileage to the location. That is, they will be paid according to the length of the shortest path from Ft. Worth to their destination city in the network of Figure 9.3. To see the impact of this proposal, we need to compute such shortest path distances for all cities.

Notice that the character of this shortest path task differs significantly from the Littleville application. Here we need only optimal path lengths, not the paths themselves. Also, we need shortest path lengths for only one origin or **source**—the Ft. Worth hub.

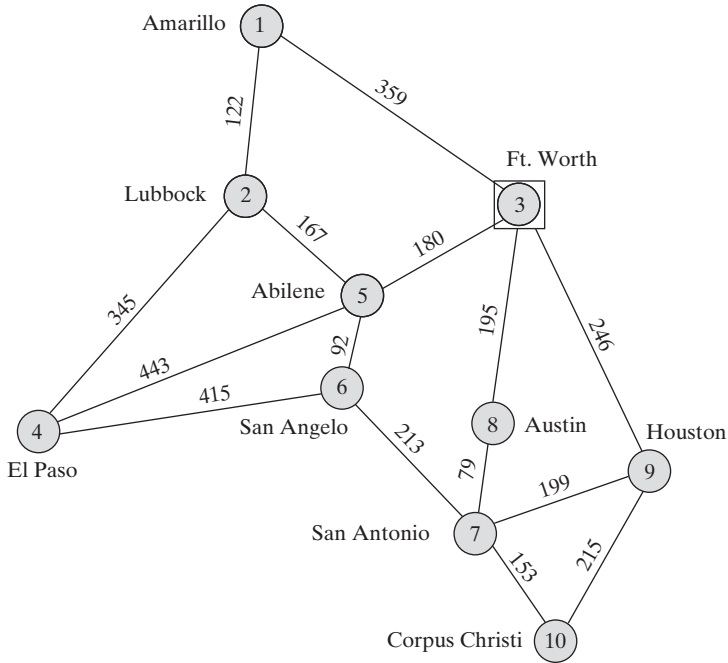


FIGURE 9.3 Network for Texas Transfer Application

- **Name:** Texas Transfer
- **Graph:** edges only
- **Costs:** nonnegative
- **Output:** shortest path lengths
- **Pairs:** one node to all others

APPLICATION 9.3: TWO RING CIRCUS

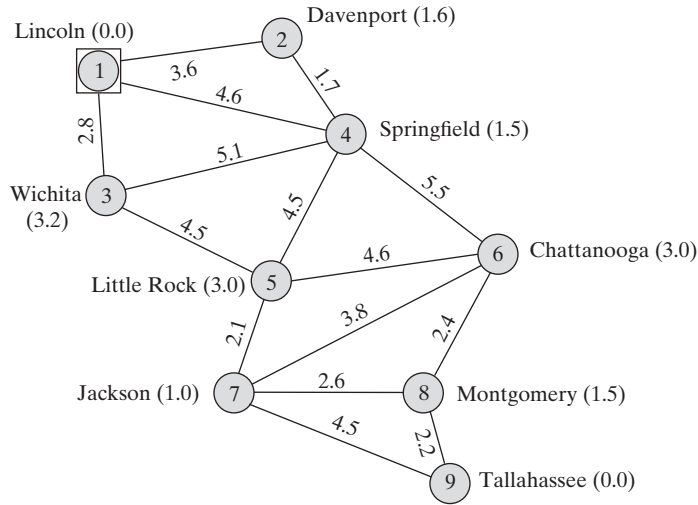
The Two Ring Circus is nearing the end of its season and planning a return to winter headquarters near Tallahassee, Florida. Present commitments will end in Lincoln, Nebraska, but there are still some opportunities for bookings in cities along the route home.

Figure 9.4(a) shows the travel routes available and the estimated costs (in thousands of dollars) of moving the circus over those routes. It also designates the cities where bookings have been offered and the anticipated net receipts (in thousands of dollars).

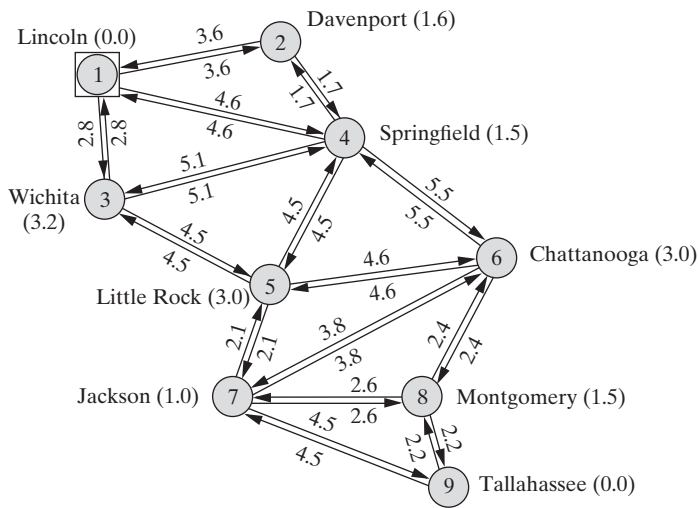
We want to compute the optimal path home for Two Ring. Notice that the cost of a path is the difference of travel costs and net receipts from performances along the way. But receipts occur at nodes of the network, not on edges as shortest path models require.

Undirected and Directed Graphs (Digraphs)

The Two Ring network of Figure 9.4(a) is an **undirected graph** because it has only edges (undirected links). The key to incorporating receipts at nodes is first to



(a) Original travel network



(b) Corresponding digraph

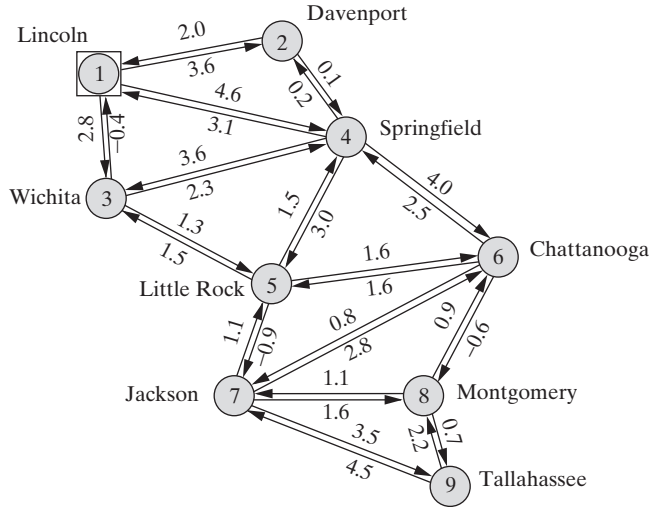
FIGURE 9.4 Two Ring Circus Application Graphs

convert to an equivalent **directed graph** or **digraph**, that is, a graph having only arcs (directed links).

Such a **directing** of a graph is easy:

Principle 9.6 | A shortest path problem including edges (i, j) of cost $c_{i,j}$ can be converted to an equivalent one on a digraph by replacing each edge with a pair of opposed arcs as





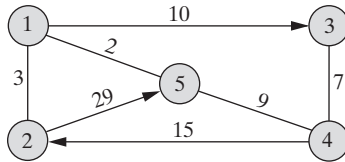
(c) After incorporating node values

FIGURE 9.4 Two Ring Circus Application Graphs (Continued)

The two parallel arcs replacing each edge merely make explicit the two directional options for using the edge in a path. Whether directed or not, a path will use the link no more than once because paths cannot repeat nodes.

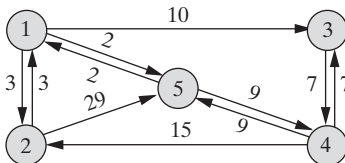
EXAMPLE 9.3: DIRECTING A GRAPH

Consider the following graph:



Show the equivalent digraph.

Solution: Applying 9.6, we replace all edges of the given graph with a pair of opposed arcs with the same length as the edge. The result is



Notice that arcs are unchanged and that the two arcs replacing each edge both have the same length as the edge.

Two Ring Application Model

Figure 9.4(b) shows the result of directing the Two Ring network. For example, Wichita-to-Little Rock edge (3, 5) becomes arcs (3, 5) and (5, 3), both with the same cost as the original edge.

Net receipt values remain on nodes in part (b) of the figure. Part (c) moves them to arcs by subtracting net receipts in each city from the cost of all outbound arcs there. For example, Little Rock-to-Jackson arc (5, 7) now has cost

$$2.1 - 3.0 = -0.9$$

to account for net receipts if Two Ring plays Little Rock. Reverse arc (7, 5) from Jackson to Little Rock has cost

$$2.1 - 1.0 = 1.1$$

to include Jackson receipts. If either of these arcs is used in an optimal path, the full effect of both travel costs and show revenues will be reflected.

It may seem wrong to subtract receipts from all outbound arcs from a city when show receipts can be realized only once. Remember, however, that a path can visit any node only once—entering on an inbound arc and leaving on an outbound. It follows that receipts for a town can be part of any path's length at most once.

The Two Ring digraph of Figure 9.4(c) also introduces another class of shortest path models. This time arc costs, being net dollar amounts, have unpredictable sign. Also, we require a shortest path for only one pair of nodes: Lincoln to Tallahassee.

- **Name:** Two Ring Circus
- **Graph:** directed
- **Costs:** arbitrary
- **Output:** shortest path
- **Pairs:** one source to one destination

9.2 DYNAMIC PROGRAMMING APPROACH TO SHORTEST PATHS

Dynamic programming methods exploit the fact that it is sometimes easiest to solve one optimization problem by taking on an entire family. If strong enough relationships can be found among optimal solutions to the various members of the family, attacking all together can be the most efficient way of solving the one(s) we really care about.

Families of Shortest Path Models

We will see later in the chapter how we sometimes have to invent a family of problems to solve a particular one by dynamic programming. With shortest path models the appropriate family is already at hand. We need only exploit the fact that most applications require optimal paths or path lengths for more than one pair of nodes.

Principle 9.7 Shortest path algorithms exploit relationships among optimal paths (and path lengths) for different pairs on nodes.

The Littleville and Texas Transfer applications of Section 9.1 already seek paths for multiple pairs of nodes. Littleville requires an optimal path between every pair of nodes, and Texas Transfer needs optimal path lengths from one node to every other.

Even if only one optimal path is required, our dynamic programming approach to shortest path problems embeds the case of application interest in such a family of problems for different origin–destination pairs. In particular, we will solve the Two Ring circus application, which needs an optimal path for only one origin–destination pair, by embedding it in the problem of finding shortest paths from the origin to all other nodes.

Functional Notation

With many optimizations going on at once, we will need some notation to keep score. Specifically, we denote optimal solutions and solution values as functions of the family member. Square brackets [. . .] enclose the parameters in such **functional notation**.

The Texas Transfer and Two Ring applications require optima for one source to all other nodes. Corresponding functional notation is

$$\nu[k] \triangleq \text{length of a shortest path from the source node to node } k$$

$$(\text{= } +\infty \text{ if no path exists})$$

$$x_{i,j}[k] \triangleq \begin{cases} 1 & \text{if arc/edge } (i, j) \text{ is part of the optimal} \\ & \text{path from the source node to node } k \\ 0 & \text{otherwise} \end{cases}$$

Notice that we adopt the convention that $\nu[k] = +\infty$ when there is no path to k .

Applications such as Littleville, which need optimal paths from all nodes to all others, have two parameters:

$$\nu[k, \ell] \triangleq \text{length of a shortest path from node } k \text{ to node } \ell$$

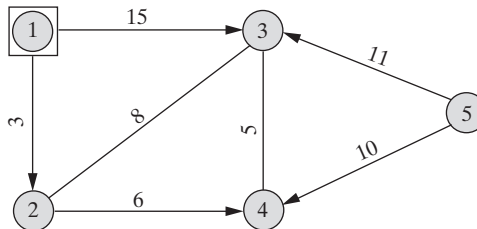
$$(\text{= } +\infty \text{ if no path exists})$$

$$x_{i,j}[k, \ell] \triangleq \begin{cases} 1 & \text{if arc/edge } (i, j) \text{ is part of the optimal path from node } k \text{ to node } \ell \\ 0 & \text{otherwise} \end{cases}$$

Again $\nu[k, \ell] = +\infty$ indicates that no path exists from k to ℓ .

EXAMPLE 9.4: UNDERSTANDING FUNCTIONAL NOTATION

Consider the problem of finding the shortest path from source node 1 in the following graph to all other nodes.



- (a) Use inspection to determine all required shortest paths.
- (b) Detail optimal solutions and solution values in functional notation.

Solution:

(a) It is easy to check that optimal paths to nodes 2, 3, and 4 are 1–2, 1–2–3, and 1–2–4, respectively. There is no path from source 1 to node 5.

(b) In functional notation these optimal paths imply that

$$\begin{aligned} \nu[1] &= 0 & x_{1,2}[1] &= x_{1,3}[1] = x_{2,3}[1] = x_{2,4}[1] = x_{3,4}[1] = 0 \\ \nu[2] &= 3 & x_{1,2}[2] &= 1, x_{1,3}[2] = x_{2,3}[2] = x_{2,4}[2] = x_{3,4}[2] = 0 \\ \nu[3] &= 11 & x_{1,2}[3] &= x_{2,3}[3] = 1, x_{1,3}[3] = x_{2,4}[3] = x_{3,4}[3] = 0 \\ \nu[4] &= 9 & x_{1,2}[4] &= x_{2,4}[4] = 1, x_{1,3}[4] = x_{2,3}[4] = x_{3,4}[4] = 0 \\ \nu[5] &= +\infty & & \text{(no path)} \end{aligned}$$

Optimal Paths and Subpaths

To formulate a dynamic programming approach to our shortest path models, we must identify connections among optimal solutions to problems for different pairs of nodes. How is the optimal path for one pair of nodes related to the optimal path for another?

To begin to see, examine Figure 9.5. The highlighted path through Austin and San Antonio is the shortest from the origin at Ft. Worth to Corpus Christi. Think now about San Antonio. The figure's Ft. Worth–Austin–San Antonio path is one way to get to San Antonio, but could any other path be shorter? Certainly not. If there were a better way to get from Ft. Worth to San Antonio than the path indicated in Figure 9.5, that better route would also yield an improvement on the

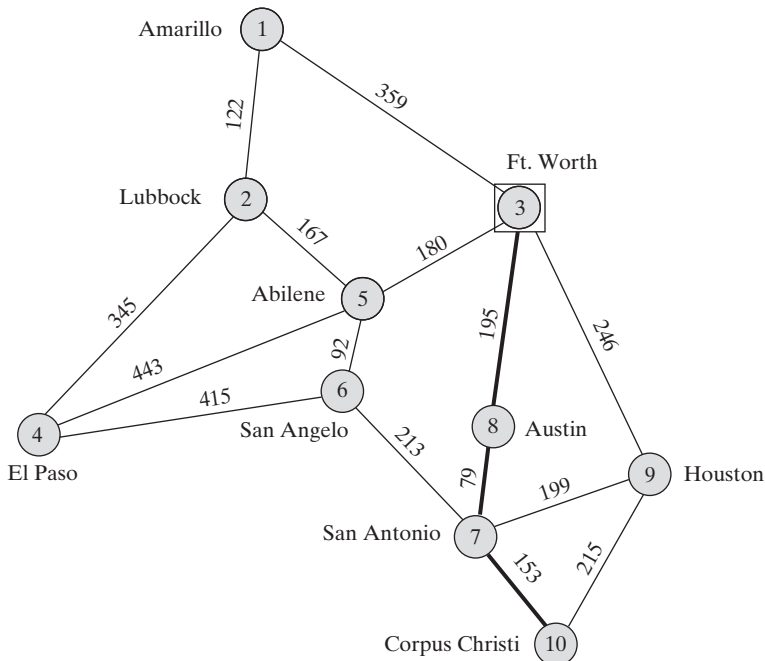


FIGURE 9.5 Optimal Path of the Texas Transfer Application

Corpus Christi optimum. We need only follow it to San Antonio and finish with the link from San Antonio to Corpus Christi.

Negative Dicycles Exception

It seems safe to conclude from examples like Figure 9.5 that optimal paths must always have optimal subpaths. Unfortunately, there is a glitch.

Consider the example in Figure 9.6. The shortest path from source $s = 1$ to node 3 clearly proceeds $1-2-3$ with $v[3] = 5$. Still, the subpath $1-2$ is not optimal. Sequence $1-3-4-2$ has lesser length $v[2] = -3$.

The logic of the preceding subsection suggests that there must be a contradiction. Simply extending the optimal path to node 2 with arc $(2, 3)$ should improve the optimum at node 3. But that extension produces the sequence $1-3-4-2-3$, which is not a path because it repeats node 3. Eliminating the repetition gives path $1-3$, but its length 10 is worse than $1-2-3$.

The troublemaker is a negative dicycle.

Definition 9.8 | A **dicycle** is a path that begins and ends at the same node, and a **negative dicycle** is a dicycle of negative total length.

Figure 9.6 contains negative dicycle $3-4-2-3$ with length

$$12 + (-25) + 3 = -10$$

Its presence permits nonoptimal subpath $1-2$ because extending shortest 1 to 2 path $1-3-4-2$ with arc $(2, 3)$ closes a dicycle that cannot be eliminated without increasing the length of the route it offers to node 3.

Negative dicycles do not occur often in applied shortest path models because they imply a sort of economic perpetual motion. Each time we transit a negative dicycle we end up closer (in total length) to any source than we were when we started. However, when negative dicycles are present, shortest path models become so dramatically more difficult that our whole solution approach must change.

Principle 9.9 | Shortest path models with negative dicycles are much less tractable than other cases because dynamic programming methods usually do not apply.

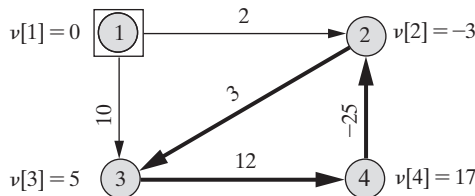
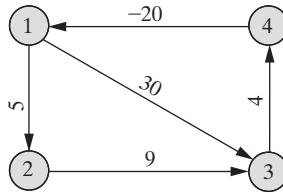


FIGURE 9.6 Example with a Negative Dicycle

EXAMPLE 9.5: IDENTIFYING NEGATIVE DICYCLES

Identify all dicycles and negative dicycles of the following graph:



Solution: The dicycles of this example are 1–2–3–4–1 of length -2 , and 1–3–4–1 of length 14. Only the first is a negative dicycle. Sequence 1–2–3–1 is not a dicycle because it violates direction.

Principle of Optimality

Fortunately, negative dicycles are the only bad case in shortest path analysis. Just as with the instance of Figure 9.6, a subpath of a shortest path can fail to be optimal only if a negative dicycle is present. Other forms satisfy the simple insight known as the **principle of optimality** for shortest path models.

Principle 9.10 In a graph with no negative dicycles, optimal paths must have optimal subpaths.

Functional Equations

To see how principle [9.10](#) helps us to compute optimal paths, let us limit our attention to one node to all other circumstances of models such as that of Texas Transfer and the Two Ring Circus. If optimal paths must have optimal subpaths, it follows that a shortest path can be constructed by extending known shortest subpaths.

Functional equations detail such recursive relationships.

Definition 9.11 **Functional equations** of a dynamic program encode the recursive connections among optimal solution values that are implied by a principle of optimality.

Functional Equations for One Node to All Others

In one node to all other shortest path cases, functional equations relate shortest path lengths $\nu[k]$.

Principle 9.12 The functional equations for shortest path problems from a single source s in a graph with no negative dicycles are

$$\begin{aligned}\nu[s] &= 0 \\ \nu[k] &= \min \{ \nu[i] + c_{i,k} : (i, k) \text{ exists} \} \quad \text{all } k \neq s\end{aligned}$$

The minimization in the second expression is taken over all neighbors i leading to k in the graph. If none exist, we adopt the convention that $\min\{\text{nothing}\} \triangleq +\infty$.

The first part of the functional equations in [9.12](#) simply says that the length of the shortest path from s to itself should = 0. The rest of the functional equations fix the recursive relationships implied by principle [9.10](#). In other words, they say that the shortest path length for node j must be the best single arc/edge extension of optimal paths to neighboring nodes i .

For an example, return again to San Antonio node $j = 7$ in Texas Transfer Figure 9.5. The corresponding equation [9.12](#) is

$$\begin{aligned} \nu[7] &= \min\{\nu[6] + c_{6,7}, \nu[8] + c_{8,7}, \nu[9] + c_{9,7}, \nu[10] + c_{10,7}\} \\ &= \min\{\nu[6] + 213, \nu[8] + 79, \nu[9] + 199, \nu[10] + 153\} \end{aligned}$$

That is, the length of the shortest path to San Antonio node 7 must be the least of 1-edge extensions to optimal paths for neighboring nodes $i = 6$ (San Angelo), 8 (Austin), 9 (Houston), and 10 (Corpus Christi). One of those must constitute an optimal subpath of the shortest path from Ft. Worth to San Antonio.

EXAMPLE 9.6: UNDERSTANDING FUNCTIONAL EQUATIONS

Return to Example 9.4. Write all corresponding functional equations and verify that they are satisfied by shortest path lengths computed in Example 9.4(b).

Solution: Following [9.12](#), functional equations are

$$\begin{aligned} \nu[1] &= 0 \\ \nu[2] &= \min\{\nu[1] + c_{1,2}, \nu[3] + c_{3,2}\} \\ \nu[3] &= \min\{\nu[1] + c_{1,3}, \nu[2] + c_{2,3}, \nu[4] + c_{4,3}\} \\ \nu[4] &= \min\{\nu[2] + c_{2,4}, \nu[3] + c_{3,4}\} \\ \nu[5] &= \min\{\} \end{aligned}$$

Now substituting optimal values yields

$$\begin{aligned} \nu[1] &= 0 \\ \nu[2] &= \min\{0 + 3, 11 + 8\} = 3 \\ \nu[3] &= \min\{0 + 15, 3 + 8, 9 + 5\} = 11 \\ \nu[4] &= \min\{3 + 6, 11 + 5\} = 9 \\ \nu[5] &= \min\{\} = +\infty \end{aligned}$$

Sufficiency of Functional Equations in the One to All Case

In the absence of negative dicycles, shortest path lengths satisfy functional equations [9.12](#) because they satisfy principle of optimality [9.10](#). Optimal paths must extend optimal subpaths.

The algorithms that make up most of this chapter depend on the fact that the result works in both directions.

Principle 9.13 Node values $\nu[k]$ in a graph with no negative dicycles are lengths of shortest paths from a given source s if and only if they satisfy functional equations [9.12](#).

That is, we can compute shortest path lengths $\nu[k]$ simply by computing values that satisfy the functional equations.

To see why, examine the $\nu[k]$ values of Texas Transport Figure 9.7. Certainly, the value a Ft. Worth source node $s = 3$ is correct. The length of a shortest path from Ft. Worth to Ft. Worth must = 0.

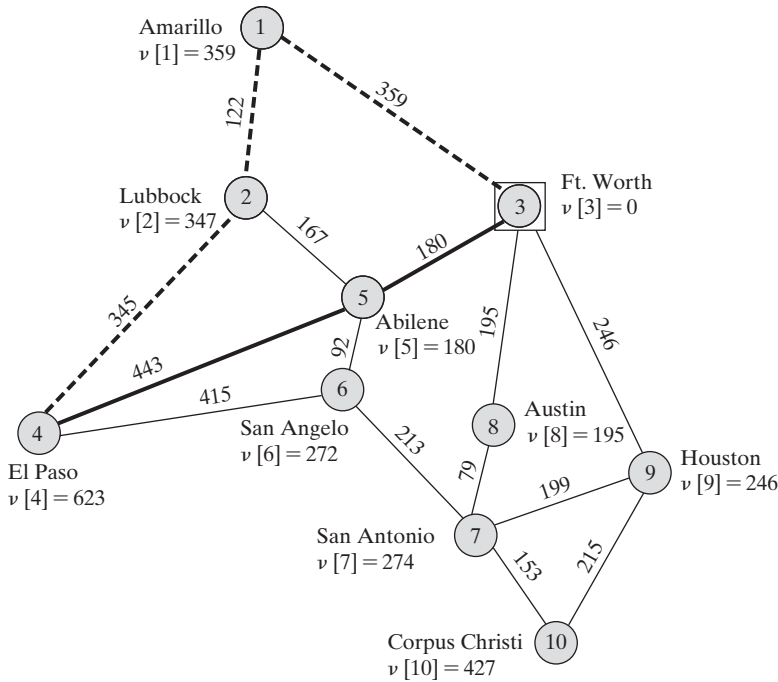


FIGURE 9.7 Sufficiency of Functional Equations

For other nodes, say El Paso node 4, we make two observations. First, functional equations [9.12] assure that $\nu[4]$ is the length of some path from s to node 4. This is true because there must be at least one neighbor i of node 4 that has

$$\nu[4] = \nu[i] + c_{i,4}$$

Here $i = 5$ (Abilene) works because it achieves the minimum in [9.12]. But Abilene node 5 must, in turn, have a neighbor j with

$$\nu[5] = \nu[j] + c_{j,5}$$

Here it is $j = 3$. We continue in this way until source node s is reached (as it just was). Then summing the relationships, simplifying, and using $\nu[s] = 0$ produces

$$\nu[4] + \nu[5] = \nu[5] + \nu[s] + c_{s,5} + c_{5,4}$$

$$\nu[4] = \nu[s] + c_{s,5} + c_{5,4}$$

$$\nu[4] = c_{s,5} + c_{5,4}$$

That is, $\nu[4]$ is the length of path $s - 5 - 4$.

The other half of the argument is to show that no other path can have shorter length than the one associated with $\nu[4]$. Consider the $s-1-2-4$ path highlighted with dashed lines in Figure 9.7. Because all values satisfy functional equations

$$\begin{aligned} \nu[1] &\leq \nu[s] + c_{s,1} \\ \nu[2] &\leq \nu[1] + c_{1,2} \\ \nu[4] &\leq \nu[2] + c_{2,4} \end{aligned}$$

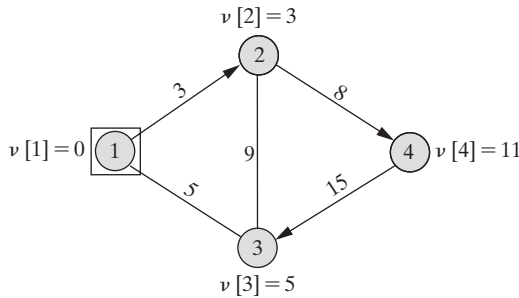
Again summing, simplifying, and using $\nu[s] = 0$, we obtain

$$\begin{aligned} \nu[1] + \nu[2] + \nu[4] &\leq \nu[s] + \nu[1] + \nu[2] + c_{s,1} + c_{1,2} + c_{2,4} \\ \nu[4] &\leq \nu[s] + c_{s,1} + c_{1,2} + c_{2,4} \\ \nu[4] &\leq c_{s,1} + c_{1,2} + c_{2,4} \end{aligned}$$

The path indicated cannot have length less than $\nu[4]$.

EXAMPLE 9.7: VERIFYING SUFFICIENCY OF FUNCTIONAL EQUATIONS

Consider the following graph and associated node values $\nu[k]$:



- (a) Verify that given values $\nu[k]$ satisfy functional equations [9.12].
- (b) Identify the path associated with the $\nu[k]$ of each node k .
- (c) Use functional equations to show why the length of path 1-3-2-4 must be $\geq \nu[4]$.

Solution:

$$\begin{aligned} \text{(a)} \quad \nu[1] &= 0 \\ \nu[2] &= \min\{\nu[1] + c_{1,2}, \nu[3] + c_{3,2}\} = \min\{0 + 3, 5 + 9\} = 3 \\ \nu[3] &= \min\{\nu[1] + c_{1,3}, \nu[2] + c_{2,3}\} = \min\{0 + 5, 3 + 9\} = 5 \\ \nu[4] &= \min\{\nu[2] + c_{2,4}, \nu[3] + c_{3,4}\} = \min\{3 + 8, 5 + 15\} = 11 \end{aligned}$$

(b) Label $\nu[s] = 0$ is the length of a null path from $s = 1$ to itself. For $k = 2$, we look for the neighbor achieving the minimum in the functional equation of $\nu[2]$. Here it is $s = 1$, so the path is 1-2 with length $\nu[2] = 3$. For $k = 3$, similar thinking yields path 1-3 with length $\nu[3] = 5$. Node $k = 4$ takes two steps. At node 4 the neighbor achieving the minimum is $i = 2$. Its minimum, in turn, occurred for $i' = 1$. Thus the path is 1-2-4 with length $\nu[4] = 11$.

(c) Functional equations for nodes 4, 2, and 3 imply that

$$\nu[4] \leq \nu[2] + c_{2,4}$$

$$\nu[2] \leq \nu[3] + c_{3,2}$$

$$\nu[3] \leq \nu[1] + c_{3,1}$$

Summing, simplifying, and using $\nu[1] = 0$, we have

$$\nu[4] + \nu[2] + \nu[3] \leq \nu[1] + \nu[2] + \nu[3] + c_{1,3} + c_{3,2} + c_{2,4}$$

$$\nu[4] \leq \nu[1] + c_{1,3} + c_{3,2} + c_{2,4}$$

$$\nu[4] \leq c_{1,3} + c_{3,2} + c_{2,4}$$

Functional Equations for All Nodes to All Others

For shortest path problems requiring optimal paths between all pairs of nodes (Littleville is an example), almost everything said so far generalizes immediately. Only the functional equations change.

Principle 9.14 The functional equations for shortest path problems from all nodes to all other nodes in a graph with no negative dicycles are

$$\nu[k, k] = 0 \text{ all } k$$

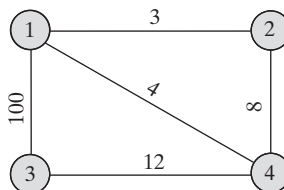
$$\nu[k, \ell] = \min \{ c_{k,\ell}, \{ \nu[k, i] + \nu[i, \ell] : i \neq k, \ell \} \} \quad \text{all } k \neq \ell$$

As with single-source cases, functional equations [9.14](#) merely detail the principle of optimality ([9.10](#)) insight that optimal paths must have optimal subpaths. Thus the shortest path from k to ℓ must consist either of arc/edge (k, ℓ) , or an optimal path from k to some intermediate node i , plus an optimal path from i to ℓ . Sufficiency for computation follows exactly as it did for earlier model forms.

Principle 9.15 Node pair values $\nu[k, \ell]$ in a graph with no negative dicycles are lengths of shortest paths from k to ℓ if and only if they satisfy functional equations [9.14](#).

EXAMPLE 9.8: VERIFYING FUNCTIONAL EQUATIONS FOR ALL PAIRS CASES

Consider the problem of finding shortest path between all pairs of nodes in the following graph.



- (a) Use inspection to identify optimal paths for all pairs.
- (b) Write the functional equations for $(i, j) = (1, 4)$ and $(2, 3)$.
- (c) Verify that your optimal path lengths of part (a) satisfy these equations.

Solution:

(a) Optimal paths are detailed in the following table:

<i>i</i>	<i>j</i> = 1		<i>j</i> = 2		<i>j</i> = 3		<i>j</i> = 4	
	<i>v</i>	Path	<i>v</i>	Path	<i>v</i>	Path	<i>v</i>	Path
1	0	—	3	1–2	16	1–4–3	4	1–4
2	3	2–1	0	—	19	2–1–4–3	7	2–1–4
3	16	3–4–1	19	3–4–1–2	0	—	12	3–4
4	4	1–4	7	4–1–2	12	4–3	0	—

(b) Following [9.14](#), the functional equations for $(i, j) = (1, 4)$ and $(2, 3)$ are

$$v[1, 4] = \min \{c_{1,4}, v[1, 2] + v[2, 4], v[1, 3] + v[3, 4]\}$$

$$v[2, 3] = \min \{c_{2,3}, v[2, 1] + v[1, 3], v[2, 4] + v[4, 3]\}$$

(c) Substitution of optimal values from the table in functional equations gives

$$v[1, 4] = \min \{4, 3 + 7, 16 + 12\} = 4$$

$$v[2, 3] = \min \{+\infty, 3 + 16, 7 + 12\} = 19$$

Solving Shortest Path Problems by Linear Programming

All the shortest path problems of this chapter can be formulated and solved by linear programming as long as the given graphs contain no negative dicycles. Details are provided in Chapter 10. Still, it is important to understand that the dynamic programming methods of the remainder of this chapter are far more efficient.

Principle 9.16 Although shortest path problems over graphs with no negative dicycles can be solved by linear programming, dynamic programming methods based on the principle of optimality are far more efficient.

9.3 SHORTEST PATHS FROM ONE NODE TO ALL OTHERS: BELLMAN–FORD

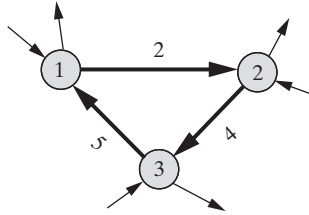
We have seen that the task of computing shortest path lengths from single source *s* to all other nodes reduces to a search for quantities $v[k]$ satisfying functional equations [9.12](#). Principle [9.13](#) guarantees that such $v[k]$ will provide shortest path lengths whenever the given graph contains no negative dicycles.

Solving the Functional Equations

The central issue in designing shortest path algorithms is how to compute functional equation solutions efficiently. Notice that the task is not as straightforward as, say, solving a system of linear equations. The $\min\{\dots\}$ operator of our functional equations makes them nonlinear.

The format of equations 9.12 suggests another way. Each equation shows an expression for a single value $\nu[k]$. Why not just evaluate those expressions?

Sometimes we can employ a one-pass evaluation strategy (see Sections 9.6 and 9.7). But if the graph contains a dicycle like the following one:



there is a difficulty. Notice that this dicycle has positive total length; the issue is not a negative dicycle. The difficulty comes from the form of the three corresponding functional equations:

$$\nu[1] = \min\{\nu[3] + 5, \dots\}$$

$$\nu[2] = \min\{\nu[1] + 2, \dots\}$$

$$\nu[3] = \min\{\nu[2] + 4, \dots\}$$

Evaluating the expression for $\nu[2]$ requires $\nu[1]$; evaluating the expression for $\nu[3]$ requires $\nu[2]$; and evaluating the expression for $\nu[1]$ requires $\nu[3]$.

Principle 9.17 Dicycles introduce circular dependencies in functional equations that preclude their solution by one-pass evaluation, even if the length of all dicycles is nonnegative.

Repeated Evaluation Algorithm: Bellman–Ford

Algorithm 9A, which is attributed to R. E. Bellman and L. R. Ford, Jr., draws on the notion of evaluation of functional equations to produce an algorithm that does work for any graph with no negative dicycles. The key insight is repeated evaluation. Each major iteration of the search (i.e., each t) evaluates the functional equation for each $\nu[k]$ using results from the preceding iteration. We stop when no result changes.

Just as with all the other searches of this book, Algorithm 9A distinguishes search values at different iterations by attaching superscripts. That is,

$$\nu^{(t)}[k] \triangleq \text{value of } \nu[k] \text{ obtained on the } t\text{th iteration}$$

When the algorithm finishes, we may want to know both shortest path lengths $\nu[k]$ and the actual paths that achieve those optimal values. Labels $d[k]$ keep notes that will allow us to recover the optimal paths. Specifically,

$$d[k] \triangleq \text{node preceding } k \text{ in the best-known path from } s \text{ to } k$$

ALGORITHM 9A: ONE TO ALL (NO NEGATIVE DICYCLES); BELLMAN-FORD SHORTEST PATHS

Step 0: Initialization. With s the source node, initialize optimal path lengths

$$\nu^{(0)}[k] \leftarrow \begin{cases} 0 & \text{if } k = s \\ +\infty & \text{otherwise} \end{cases}$$

and set iteration counter $t \leftarrow 1$.

Step 1: Evaluation. For each k evaluate

$$\nu^{(t)}[k] \leftarrow \min \{ \nu^{(t-1)}[i] + c_{i,k} : (i, k) \text{ exists} \}$$

If $\nu^{(t)}[k] < \nu^{(t-1)}[k]$, also set $d[k] \leftarrow$ the number of a neighboring node i achieving the minimum $\nu^{(t)}[k]$.

Step 2: Stopping. Terminate if $\nu^{(t)}[k] = \nu^{(t-1)}[k]$ for every k , or if $t =$ the number of nodes in the graph. Values $\nu^{(t)}[k]$ then equal the required shortest path lengths unless some $\nu^{(t)}[k]$ changed at the last t , in which case the graph contains a negative dicycle.

Step 3: Advance. If some $\nu[k]$ changed and $t <$ the number of nodes, increment $t \leftarrow t + 1$ and return to Step 1.

Bellman-Ford Solution of the Two Ring Circus Application

Before dealing with all the details of Bellman-Ford Algorithm 9A, let us apply it to the Two Ring Circus application [Figure 9.4(c)]. Table 9.1 provides details.

TABLE 9.1 Bellman-Ford Algorithm Solution of Two Ring Application

t	$\nu^{(t)}[1]$	$\nu^{(t)}[2]$	$\nu^{(t)}[3]$	$\nu^{(t)}[4]$	$\nu^{(t)}[5]$	$\nu^{(t)}[6]$	$\nu^{(t)}[7]$	$\nu^{(t)}[8]$	$\nu^{(t)}[9]$
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1		3.6	2.8	4.6					
2				3.7	4.1				
3						5.7	3.2		
4								4.8	6.7
5									5.5
6									
t	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$	$d[8]$	$d[9]$
1		1	1	1					
2				2	3				
3						5	5		
4								7	7
5									8
6									

The process begins by initializing node values $\nu[k]$. Lacking better information, the algorithm initializes all except $\nu[s]$ at the worst possible value for a minimizing optimization, $+\infty$.

The first main iteration of the algorithm computes new $\nu^{(1)}[k]$ by evaluating the functional equations using the $\nu^{(0)}[k]$. For example,

$$\begin{aligned} \nu^{(1)}[2] &= \min \{ \nu^{(0)}[1] + c_{1,2}, \nu^{(0)}[4] + c_{4,2} \} \\ &= \min \{ 0 + 3.6, \infty + 0.2 \} \\ &= 3.6 \end{aligned}$$

Values change for $\nu^{(1)}[2] = 3.6$, $\nu^{(1)}[3] = 2.8$, and $\nu^{(1)}[4] = 4.6$.

Whenever a new value is assigned to any path length $\nu[k]$, we also want to record the decision option that produced it. Here that means keeping track of the neighboring node i through which a new value for $\nu[k]$ was derived. For example, $d[2] \leftarrow 1$ because the minimum establishing $\nu^{(1)}[2]$ was achieved by extending the best known path to neighboring node $i = 1$. Similarly, $d[3] \leftarrow 1$ and $d[4] \leftarrow 1$.

Advancing to iteration $t = 2$, we repeat the process. This time, values change at nodes $k = 4$ and 5. The first of these illustrates the temporary nature of $\nu[k]$ values in the Bellman–Ford algorithm. At iteration 1 we set $\nu^{(1)} = 4.6$ because

$$\begin{aligned} \nu^{(1)}[4] &= \min \{ \nu^{(0)}[1] + c_{1,4}, \nu^{(0)}[2] + c_{2,4}, \nu^{(0)}[3] + c_{3,4}, \nu^{(0)}[5] + c_{5,4}, \nu^{(0)}[6] + c_{6,4} \} \\ &= \min \{ 0 + 4.6, \infty + 0.1, \infty + 2.3, \infty + 1.5, \infty + 2.5 \} \\ &= 4.6 \end{aligned}$$

After iteration 2, however, updated values produce

$$\begin{aligned} \nu^{(2)}[4] &= \min \{ \nu^{(1)}[1] + c_{1,4}, \nu^{(1)}[2] + c_{2,4}, \nu^{(1)}[3] + c_{3,4}, \nu^{(1)}[5] + c_{5,4}, \nu^{(1)}[6] + c_{6,4} \} \\ &= \min \{ 0 + 4.6, 3.6 + 0.1, 2.8 + 2.3, \infty + 1.5, \infty + 2.5 \} \\ &= 3.7 \end{aligned}$$

Processing continues in this way through iterations $t = 3, 4, 5$, with better and better values for the $\nu[k]$ being computed. However, evaluation of functional equations at $t = 6$ produces no changed values. This is the signal to stop and accept values $\nu^{(6)}[k]$ as optimal. Figure 9.8 shows corresponding optimal paths.

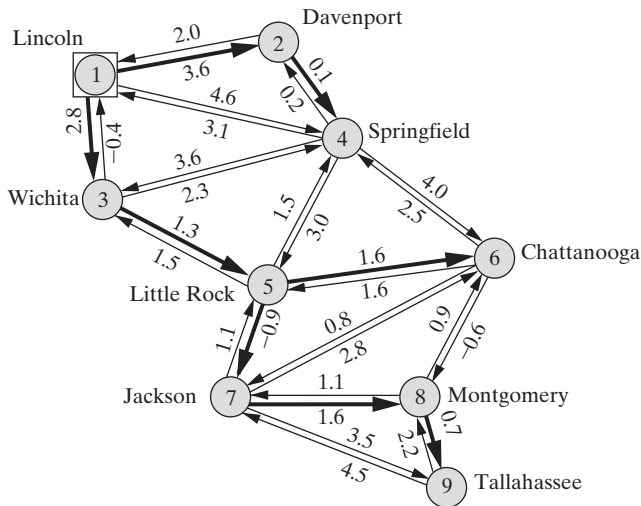
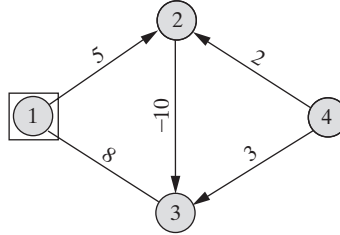


FIGURE 9.8 Optimal Paths in Two Ring Application

EXAMPLE 9.9: APPLYING THE BELLMAN–FORD ALGORITHM

Apply Bellman–Ford Algorithm 9A to compute the lengths of shortest paths from source node $s = 1$ to all other nodes of the following graph.



Solution: Initialization Step 0 sets

$$\nu^{(0)}[1] = 0 \quad \nu^{(0)}[2] = \nu^{(0)}[3] = \nu^{(0)}[4] = \infty$$

On iteration $t = 1$, evaluation of functional equations gives

$$\nu^{(1)}[1] = 0$$

$$\begin{aligned} \nu^{(1)}[2] &= \min\{\nu^{(0)}[1] + c_{1,2}, \nu^{(0)}[4] + c_{4,2}\} \\ &= \min\{0 + 5, \infty + 2\} = 5 \quad (\text{making } d[2] = 1) \end{aligned}$$

$$\begin{aligned} \nu^{(1)}[3] &= \min\{\nu^{(0)}[1] + c_{1,3}, \nu^{(0)}[2] + c_{2,3}, \nu^{(0)}[4] + c_{4,3}\} \\ &= \min\{0 + 8, \infty - 10, \infty + 3\} = 8 \quad (\text{making } d[3] = 1) \end{aligned}$$

$$\begin{aligned} \nu^{(1)}[4] &= \min\{\} \\ &= \infty \end{aligned}$$

Continuing in tabular form, we have

t	$\nu^{(t)}[1]$	$\nu^{(t)}[2]$	$\nu^{(t)}[3]$	$\nu^{(t)}[4]$
0	0	$+\infty$	$+\infty$	$+\infty$
1		5	8	
2			-5	
3				

t	$d[1]$	$d[2]$	$d[3]$	$d[4]$
1		1	1	
2			2	
3				

Iteration $t = 2$ revises $\nu^{(2)}[3] = -5$, which makes $d[3] = 2$. The algorithm terminates when all values repeat at iteration $t = 3$.

Extracting final values from the table, the shortest path to node 2 has length $\nu[2] = 5$, and the shortest path to node 3 has length $\nu[3] = -5$. Since the final $\nu[4] = \infty$, there is no path to node 4.

Justification of the Bellman–Ford Algorithm

Having seen how Algorithm 9A works, we now need to investigate why. What guarantees that the $\nu^{(t)}[k]$ correspond to optimal values when termination criteria are fulfilled?

Look again at the complex of optimal paths in Figure 9.8. Some nodes are one arc away from the source along optimal paths, some are two arcs away, some three, and so on. Notice, however, that none is more than

$$\text{number of nodes} - 1 = 8$$

arcs away. A path in a 9 node network can contain no more than $(9 - 1) = 8$ arcs. With any greater number, some node would have to repeat.

The value $\nu^{(t)}[1] = 0$ at the source is correct from $t = 0$. After iteration $t = 1$, all optimal lengths that directly depend upon it (i.e., all $\nu^{(1)}[k]$ with optimal paths of 1 arc) must also be correct. Similarly, after $t = 2$, those $\nu^{(2)}[k]$ with optimal paths of 2 arcs must also be final. In general, intermediate $\nu^{(t)}[k]$ reflect optimal paths of t or fewer steps. All must be final before $t =$ the number of nodes. Furthermore, if we go a whole iteration without changing any node's value, no more changes will occur. We might as well terminate computation.

As we consider different shortest paths and other algorithms, it will be important to track how rapidly computational effort grows with dimensions of the instance. **Computational orders**, denoted $O(\cdot)$, are functions of instance size that bound the effort required by algorithms in the worst case (see Section 14.2 for details). For Bellman–Ford on a graph of n nodes, our analysis of algorithm correctness implies at most $O(n)$ major iterations. Each involves checking all of the n nodes vs. nodes that flow into them—possibly all nodes. Thus the computation is bounded by $O(n^2)$ effort per iteration, and $O(n^3)$ overall.

Recovering Optimal Paths

Some shortest path models require only shortest path lengths, but many require the optimal paths that achieve those lengths.

To recover optimal paths, we exploit a consequence of principle of optimality [9.10](#). When optimal paths must have optimal subpaths, the shortest path to any node is just a 1-arc/edge extension of the shortest path to a neighbor. This implies that the last link is enough to recover the full path. We only need to move to the indicated neighbor and follow its optimal path.

The $d[k]$ labels we record during computation provide the needed information because they record the last link of the sequence establishing optimal path length $\nu[k]$.

Principle 9.18 | At the completion of Algorithm 9A, a shortest path from source s to any other node k can be recovered by starting at k , backtracking to neighboring node $d[k]$, and continuing with an optimal path from s to the neighbor until source s is encountered.

To illustrate, consider the Lincoln-to-Tallahassee path we really need in Two Ring Circus Figure 9.8. Starting at Tallahassee node 9, final $d[9] = 8$ tells us the optimal path enters through node 8. Since the optimal path to node 9 must include an optimal subpath to node 8, we can continue the process by focusing on the shortest path to node 8. There $d[8] = 7$ tells us the path comes from node 7.

Continuing through $d[7] = 5$, $d[5] = 3$, and $d[3] = 1$ we finally reach the source. Thus the full optimal path is 1–3–5–7–8–9: Lincoln to Wichita to Little Rock to Jackson to Montgomery to Tallahassee.

EXAMPLE 9.10: RECOVERING PATHS WITH BELLMAN–FORD

Return to the graph of Example 9.9. Recover an optimal path to node 3 using the $d[k]$ labels assigned during Example 9.9 processing.

Solution: Applying principle 9.18, we begin at destination node 3. Since $d[3] = 2$, the optimal path enters from node 2. The optimal path to node 2, in turn, enters through node $d[2] = 1$. Having reached the source, we conclude that a shortest path is 1–2–3.

Encountering Negative Dicycles with Bellman–Ford

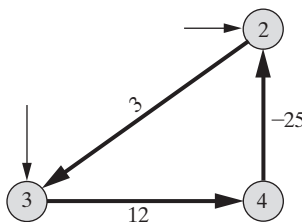
We know that shortest path models with negative dicycles usually cannot be solved by the dynamic programming methods of Algorithm 9A (principle 9.9). But if we confront a very large model, it could be quite difficult to know whether negative dicycles are present.

To see what happens, we could try applying the Bellman–Ford algorithm to our negative dicycle example of Figure 9.6:

t	$\nu^{(t)}[1]$	$\nu^{(t)}[2]$	$\nu^{(t)}[3]$	$\nu^{(t)}[4]$	$d[1]$	$d[2]$	$d[3]$	$d[4]$
0	0	$+\infty$	$+\infty$	$+\infty$				
1		2	10			1	1	
2			5	22			2	3
3		-3		17		4		
4		-8	0					

Initial values $\nu^{(0)}[k]$ are chosen just as in other examples, and each iteration updates $\nu[k]$ and $d[k]$ labels. The critical difference is that labels continued to change on iteration $t = 4$, even though the graph has only 4 nodes. Evaluation around negative dicycle drives $\nu^{(t)}$ lower and lower because

$$\begin{aligned} \nu^{(t)}[2] &\leftarrow \nu^{(t-1)}[4] - 25 \\ \nu^{(t)}[3] &\leftarrow \nu^{(t-1)}[2] + 3 \\ \nu^{(t)}[4] &\leftarrow \nu^{(t-1)}[3] + 12 \end{aligned}$$



This behavior illustrates why the Bellman–Ford algorithm terminates if labels continue to change on iteration $t =$ the number of nodes.

Principle 9.19 If Algorithm 9A encounters negative dicycles, it will demonstrate their presence by continuing to change $\nu^{(t)}[k]$ on iteration $t =$ the number of nodes. Any node k with such a changing $\nu^{(t)}[k]$ belongs to a negative dicycle.

We can actually exhibit a negative dicycle by following the $d[k]$ labels of any k for which $\nu^{(t)}[k]$ changed on the final iteration. For example, we could begin at $k = 2$ because $\nu^{(4)}[2] = -8 \neq \nu^{(3)}[2] = -3$. Tracing backward through the $d[k]$ labels, we find that

$$d[2] = 4$$

$$d[4] = 3$$

$$d[3] = 2$$

As soon as node 2 repeats, we know we have completed negative dicycle 2–3–4–2.

9.4 SHORTEST PATHS FROM ALL NODES TO ALL OTHERS: FLOYD–WARSHALL

When shortest paths between all pairs of nodes are required in a graph with no negative dicycles (e.g., in our Littleville application of Figure 9.1), the task is to compute $\nu[k, \ell]$ efficiently, satisfying all-to-all functional equations [9.14].

Floyd–Warshall Algorithm

Algorithm 9B, which is attributed to R. W. Floyd and S. Warshall, does just that. Search quantities

$$\nu^{(t)}[k, \ell] \triangleq \text{length of a shortest path for } k \text{ to } \ell \text{ using only} \\ \text{intermediate nodes numbered less than or equal to } t$$

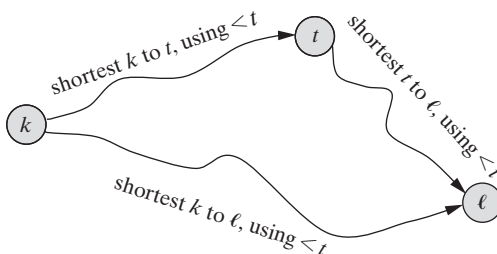
converge to the required shortest path lengths $\nu[k, \ell]$. Corresponding decision labels

$$d[k, \ell] \triangleq \text{node just before } \ell \text{ on the current path from } k \text{ to } \ell$$

track the associated paths.

The key to Algorithm 9B's effectiveness is the clever sequence in which quantities are calculated. Initialization correctly sets the $\nu^{(0)}[k, \ell] = c_{k,\ell}$ because the only path from k to ℓ that has no intermediate nodes (i.e., none with positive node number ≤ 0) is an arc/edge (k, ℓ) with cost $c_{k,\ell}$.

Subsequent iterations consider concatenating previous results as indicated in the following sketch:



ALGORITHM 9B: ALL TO ALL (NO NEGATIVE DICYCLES); FLOYD–WARSHALL SHORTEST PATHS

Step 0: Initialization. All nodes should have consecutive positive numbers starting with 1. For all arcs and edges (k, ℓ) in the graph, initialize

$$\begin{aligned} v^{(0)}[k, \ell] &\leftarrow c_{k, \ell} \\ d[k, \ell] &\leftarrow k \end{aligned}$$

For k, ℓ pairs with no arc/edge (k, ℓ) , assign

$$v^{(0)}[k, \ell] \leftarrow \begin{cases} 0 & \text{if } k = \ell \\ +\infty & \text{otherwise} \end{cases}$$

Also set iteration counter $t \leftarrow 1$.

Step 1: Evaluation. For all $k, \ell \neq t$ update

$$v^{(t)}[k, \ell] \leftarrow \min \{ v^{(t-1)}[k, \ell], v^{(t-1)}[k, t] + v^{(t-1)}[t, \ell] \}$$

If $v^{(t)}[k, \ell] < v^{(t-1)}[k, \ell]$, also set $d[k, \ell] \leftarrow d[t, \ell]$.

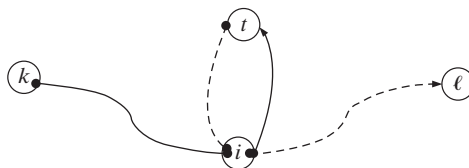
Step 2: Stopping. Terminate if $t =$ the number of nodes in the graph, or if $v^{(t)}[k, k] < 0$ for any node k . Values $v^{(t)}[k, \ell]$ then equal the required shortest path lengths unless some $v^{(t)}[k, k]$ is negative, in which case the graph contains a negative dicycle through k .

Step 3: Advance. If $t <$ the number of nodes and all $v^{(t)}[k, k] \geq 0$, increment $t \leftarrow t + 1$ and return to Step 1.

By iteration t we already know optimal paths using nodes numbered $< t$ as intermediaries. Thus $v^{(t)}[k, \ell]$, which is the shortest using nodes numbered $\leq t$, must be either the current best or one visiting t via an optimal path from k to t followed by one from t to ℓ . The latter subpaths must involve intermediaries numbered $< t$, so they are already recorded in labels $v^{(t-1)}[k, t]$ and $v^{(t-1)}[t, \ell]$. At termination, all the possibilities of the minimization in functional equations [9.14](#) have been checked, which implies that the final $v[k, \ell]$ are optimal.

In terms of computational orders on a graph of n nodes (see Section 14.2), this analysis is easily seen to yield $O(n)$ major iterations, each involving $O(n^2)$ checking of all pairs of nodes. Thus the computation is $O(n^3)$ overall.

There remains one concern. How can we be sure that piecing together a shortest path from k to t in the above figure with a shortest path from t to ℓ produces a path from k to ℓ ? It certainly does produce a sequence of arcs leading from k to ℓ , and if we adopt it, the total length must be less than previous best $v^{(t-1)}[k, \ell]$. But as illustrated below, there could be one or more nodes $i < t$ in common along the two components. Then the arc sequence would not be a path.



To see that such cases cannot occur, notice first that the dicycle from i to t and back in the sequence depicted above cannot have negative total length; negative dicycles are precluded for Algorithm 9B. Thus removing it from the rest leaves a sequence through nodes numbered $< t$ with length no greater than the full pieced sequence. Continuing in this way to remove nonnegative dicycles will eventually leave us with a path from k to ℓ using only nodes number $< t$ with length at most that of the original pieced sequence. But we know that must be $< \nu^{(t-1)}[k, \ell]$, which violates optimality of that prior best length. The only way to avoid such conflicts is that the full pieced sequence should itself be a path to in the first place.

Floyd–Warshall Solution of the Littleville Application

To illustrate Floyd–Warshall Algorithm 9B, we apply it to the Littleville Application of Figure 9.1. Table 9.2 presents initial values, along with the results for iterations $t = 1, 9,$ and 10 .

The algorithm begins by initializing $\nu^{(0)}[k, \ell]$ for all arcs/edges (k, ℓ) at direct cost $c_{k, \ell}$, and fixing labels $d[k, \ell]$ accordingly. For example, $\nu^{(0)}[6, 7] \leftarrow c_{6,7} = 28$, and $d[6, 7] \leftarrow 6$ to indicate that the current (direct) path from node 6 to node 7 enters 7 via 6.

Values for which there is no arc/edge (k, ℓ) begin at $+\infty$ except for k -to- k path lengths $\nu^{(0)}[k, k]$, which are assigned value 0. Thus $\nu^{(0)}[1, 10] \leftarrow \infty$ in Table 9.2, because the Littleville network has no direct link from node 1 to node 10, and $\nu^{(0)}[9, 9] \leftarrow 0$ to indicate that the shortest path from node 9 to itself has length = 0.

Iteration $t = 1$ advances to Algorithm 9B, Step 1. For all node pairs k and ℓ , both different than $t = 1$, we update $\nu[k, \ell]$ to the minimum of what it was before and the concatenation the current path from k to t with the current path from t to ℓ .

Table 9.2 shows that the only new (boxed) value occurs at $k = 5, \ell = 2$. There is no direct path from node 5 to node 2 [refer back to Figure Example 9.1(b)], so $\nu^{(0)}[5, 2] = +\infty$. But iteration $t = 1$ evaluates node 1 as an intermediary. The length of the path 5–1–2 is recorded by evaluating the corresponding functional equation [\[9.14\]](#):

$$\begin{aligned} \nu^{(1)}[5, 2] &\leftarrow \min\{\nu^{(0)}[5, 2], \nu^{(0)}[5, 1] + \nu^{(0)}[1, 2]\} \\ &= \min\{\infty, 20 + 12\} \\ &= 32 \end{aligned}$$

Decision label $d[5, 2]$ is also corrected to indicate that the current path from $k = 5$ to $\ell = 2$ now enters through node $d[5, 2] = 1$.

Skipping ahead, Table 9.2 shows other values changing as nodes $t = 9$ and $t = 10$ are tried as intermediate nodes. One is

$$\begin{aligned} \nu^{(10)}[3, 8] &\leftarrow \min\{\nu^{(9)}[3, 8], \nu^{(9)}[3, 10] + \nu^{(9)}[10, 8]\} \\ &= \min\{87, 51 + 28\} \\ &= 79 \end{aligned}$$

Another value changed at $t = 10$ is

$$\begin{aligned} \nu^{(10)}[4, 1] &\leftarrow \nu^{(9)}[4, 10] + \nu^{(9)}[10, 1] \\ &= 38 + 66 = 104 \end{aligned}$$

TABLE 9.2 Floyd–Warshall Algorithm Solution of Littleville Application

Initial Values										
$\nu^{(0)}[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	0	12	∞	∞	∞	∞	∞	∞	∞	∞
$k = 2$	∞	0	18	∞	∞	32	∞	∞	∞	∞
$k = 3$	∞	∞	0	13	∞	∞	30	∞	∞	∞
$k = 4$	∞	∞	∞	0	∞	∞	∞	∞	∞	38
$k = 5$	20	∞	∞	∞	0	18	∞	∞	∞	∞
$k = 6$	∞	32	∞	∞	18	0	28	∞	25	∞
$k = 7$	∞	∞	30	∞	∞	28	0	∞	21	49
$k = 8$	∞	∞	∞	∞	18	∞	∞	0	36	∞
$k = 9$	∞	∞	∞	∞	∞	25	21	36	0	40
$k = 10$	∞	∞	∞	∞	∞	∞	49	28	40	0
$d[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	—	1	—	—	—	—	—	—	—	—
$k = 2$	—	—	2	—	—	2	—	—	—	—
$k = 3$	—	—	—	3	—	—	3	—	—	—
$k = 4$	—	—	—	—	—	—	—	—	—	4
$k = 5$	5	—	—	—	—	5	—	—	—	—
$k = 6$	—	6	—	—	6	—	6	—	6	—
$k = 7$	—	—	7	—	—	7	—	—	7	7
$k = 8$	—	—	—	—	8	—	—	—	8	—
$k = 9$	—	—	—	—	—	9	9	9	—	9
$k = 10$	—	—	—	—	—	—	10	10	10	—
After Iteration $t = 1$										
$\nu^{(1)}[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	0	12	∞	∞	∞	∞	∞	∞	∞	∞
$k = 2$	∞	0	18	∞	∞	32	∞	∞	∞	∞
$k = 3$	∞	∞	0	13	∞	∞	30	∞	∞	∞
$k = 4$	∞	∞	∞	0	∞	∞	∞	∞	∞	38
$k = 5$	20	32	∞	∞	0	18	∞	∞	∞	∞
$k = 6$	∞	32	∞	∞	18	0	28	∞	25	∞
$k = 7$	∞	∞	30	∞	∞	28	0	∞	21	49
$k = 8$	∞	∞	∞	∞	18	∞	∞	0	36	∞
$k = 9$	∞	∞	∞	∞	∞	25	21	36	0	40
$k = 10$	∞	∞	∞	∞	∞	∞	49	28	40	0
$d[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	—	1	—	—	—	—	—	—	—	—
$k = 2$	—	—	2	—	—	2	—	—	—	—
$k = 3$	—	—	—	3	—	—	3	—	—	—
$k = 4$	—	—	—	—	—	—	—	—	—	4
$k = 5$	5	1	—	—	—	5	—	—	—	—
$k = 6$	—	6	—	—	6	—	6	—	6	—
$k = 7$	—	—	7	—	—	7	—	—	7	7
$k = 8$	—	—	—	—	8	—	—	—	8	—
$k = 9$	—	—	—	—	—	9	9	9	—	9
$k = 10$	—	—	—	—	—	—	10	10	10	—

TABLE 9.2 Continued

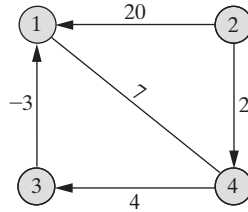
After Iteration $t = 9$										
$\nu^{(9)}[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	0	12	30	43	62	44	60	105	69	81
$k = 2$	70	0	18	31	50	32	48	93	57	69
$k = 3$	96	90	0	13	76	58	30	87	51	51
$k = 4$	∞	∞	∞	0	∞	∞	∞	∞	∞	38
$k = 5$	20	32	50	63	0	18	46	79	43	83
$k = 6$	38	32	50	63	18	0	28	61	25	65
$k = 7$	66	60	30	43	46	28	0	57	21	49
$k = 8$	38	50	68	81	18	36	57	0	36	76
$k = 9$	63	57	51	64	43	25	21	36	0	40
$k = 10$	66	78	79	92	46	64	49	28	40	0
$d[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	—	1	2	3	6	2	3	9	6	4
$k = 2$	5	—	2	3	6	2	3	9	6	4
$k = 3$	5	6	—	3	6	7	3	9	7	4
$k = 4$	—	—	—	—	—	—	—	—	—	4
$k = 5$	5	1	2	3	—	5	6	9	6	9
$k = 6$	5	6	2	3	6	—	6	9	6	9
$k = 7$	5	6	7	3	6	7	—	9	7	7
$k = 8$	5	1	2	3	8	5	9	—	8	9
$k = 9$	5	6	7	3	6	9	9	9	—	9
$k = 10$	5	1	7	3	8	5	10	10	10	—
After Iteration $t = 10$										
$\nu^{(10)}[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	0	12	30	43	62	44	60	105	69	81
$k = 2$	70	0	18	31	50	32	48	93	57	69
$k = 3$	96	90	0	13	76	58	30	79	51	51
$k = 4$	104	116	117	0	84	102	87	66	78	38
$k = 5$	20	32	50	63	0	18	46	79	43	83
$k = 6$	38	32	50	63	18	0	28	61	25	65
$k = 7$	66	60	30	43	46	28	0	57	21	49
$k = 8$	38	50	68	81	18	36	57	0	36	76
$k = 9$	63	57	51	64	43	25	21	36	0	40
$k = 10$	66	78	79	92	46	64	49	28	40	0
$d[k, \ell]$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	$\ell = 7$	$\ell = 8$	$\ell = 9$	$\ell = 10$
$k = 1$	—	1	2	3	6	2	3	9	6	4
$k = 2$	5	—	2	3	6	2	3	9	6	4
$k = 3$	5	6	—	3	6	7	3	10	7	4
$k = 4$	5	1	7	—	8	5	10	10	10	4
$k = 5$	5	1	2	3	—	5	6	9	6	9
$k = 6$	5	6	2	3	6	—	6	9	6	9
$k = 7$	5	6	7	3	6	7	—	9	7	7
$k = 8$	5	1	2	3	8	5	9	—	8	9
$k = 9$	5	6	7	3	6	9	9	9	—	9
$k = 10$	5	1	7	3	8	5	10	10	10	—

Look carefully at the corresponding $d[4, 1]$. Notice that it is set to $d[10, 1] = 5$, not $t = 10$. We want $d[4, 1]$ to be the next-to-last node of the newly recorded path, which is information available in $d[10, 1]$. This may or may not equal the intermediate t .

After iteration $t = 10$ every node has been tried as an intermediary; the Littleville network has only 10 nodes. Thus the algorithm terminates, and values $\nu^{(10)}[k, \ell]$ are optimal. That is, they provide the lengths of shortest paths from each k to each ℓ .

EXAMPLE 9.11: APPLYING THE FLOYD–WARSHALL ALGORITHM

Consider the following digraph:



Apply Floyd–Warshall Algorithm 9B to compute the lengths of shortest paths between all pairs of nodes.

Solution: Paralleling Table 9.2, we show tables of $\nu^{(\ell)}[k, \ell]$ and $d[k, \ell]$, boxing changed values.

$\nu^{(0)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	∞	∞	7	—	—	—	1
2	20	0	∞	2	2	—	—	2
3	-3	∞	0	∞	3	—	—	—
4	7	∞	4	0	4	—	4	—

$\nu^{(1)}[k, \ell] = \nu^{(2)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	∞	∞	7	—	—	—	1
2	20	0	∞	2	2	—	—	2
3	-3	∞	0	4	3	—	—	1
4	7	∞	4	0	4	—	4	—

$\nu^{(3)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	∞	∞	7	—	—	—	1
2	20	0	∞	2	2	—	—	2
3	-3	∞	0	4	3	—	—	1
4	1	∞	4	0	3	—	4	—

k	$\nu^{(4)}[k, \ell]$				$d[k, \ell]$			
	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	∞	1	7	—	—	4	1
2	3	0	6	2	3	—	4	2
3	−3	∞	0	4	3	—	—	1
4	1	∞	4	0	3	—	4	—

The table for $t = 1$ is identical to that of $t = 2$ because no paths use node 2 as an intermediary.

With only 4 nodes in the graph, computations are complete after iteration $t = 4$. Value $\nu^{(4)}[1, 2] = \infty$ indicates that there is no path from node 1 to node 2.

Recovering Optimal Paths

Just as with earlier cases (principle 9.18), final labels $d[k, \ell]$ allow us to recover an optimal path between any pair of nodes.

Principle 9.20 At the completion of Algorithm 9B, a shortest path from any node k to any other node ℓ can be recovered by starting at ℓ , backtracking to neighboring node $d[k, \ell]$, and continuing with an optimal path from k to the neighbor until k itself is encountered.

To illustrate, refer again to final $t = 10$ labels for the Littleville application displayed in Table 9.2. A shortest path from node $k = 3$ to node $\ell = 8$ is recovered by beginning at destination ℓ . Label $d[3, 8] = 10$ indicates that an optimal path enters from node 10. Now backtracking to node 10, we apply principle of optimality 9.10 and follow the best path from 3 to 10. Label $d[3, 10] = 4$ tells us that that path enters through node 4. Backtracking again, we discover label $d[3, 4] = 3$. Origin $k = 3$ has been reached, and the optimal path is 3–4–10–8.

EXAMPLE 9.12: RECOVERING A FLOYD–WARSHALL PATH

Use final labels of Example 9.11 to recover an optimal path from $k = 2$ to $\ell = 1$.

Solution: Applying principle 9.20, we backtrack from destination $\ell = 1$. Label $d[2, 1] = 3$ indicates the optimal path enters through node 3. Continuing, the optimal path from 2 to 3 enters through $d[2, 3] = 4$, and the one for 4 enters through $d[2, 4] = 2$, which brings us to origin $k = 2$. Thus the optimal path is 2–4–3–1.

Detecting Negative Dicycles with Floyd–Warshall

Just as with the Bellman–Ford algorithm for a single source, Floyd–Warshall Algorithm 9B guarantees optimal paths and path lengths only if the given graph contains no negative dicycles. Functional equations 9.14, on which the algorithm is based, need not hold in the presence of negative dicycles.

What happens when Algorithm 9B is applied to a graph with negative dicycles? The secondary stopping criterion of algorithm Step 2 comes into play, and computation terminates with the conclusion that a negative dicycle exists. The actual condition flagging presence of a negative dicycle in the Floyd–Warshall procedure is $\nu^{(t)}[k, k] < 0$ (i.e., a shortest path from any k to itself turning negative).

Principle 9.21 If Algorithm 9B is applied to a graph with negative dicycles, it will demonstrate their presence by making some $\nu^{(t)}[k, k] < 0$ and terminating. Under those circumstances, the implied negative dicycle includes node k .

To illustrate, return to the negative dicycle example of Figure 9.6. Application of Algorithm 9B produces the following:

$\nu^{(0)}[k, \ell] = \nu^{(1)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	2	10	∞	—	1	1	—
2	∞	0	3	∞	—	—	2	—
3	∞	∞	0	12	—	—	—	3
4	∞	-25	∞	0	—	4	—	—

$\nu^{(2)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	2	5	∞	—	1	1	—
2	∞	0	3	∞	—	—	2	—
3	∞	∞	0	12	—	—	—	3
4	∞	-25	-22	0	—	4	2	—

$\nu^{(3)}[k, \ell]$					$d[k, \ell]$			
k	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	2	5	17	—	1	2	3
2	∞	0	3	15	—	—	2	3
3	∞	∞	0	12	—	—	—	3
4	∞	-25	-22	-10	—	4	3	3

Notice that on iteration $t = 3$, value

$$\begin{aligned} \nu^{(3)}[4, 4] &\leftarrow \min\{0, \nu^{(2)}[4, 3] + \nu^{(2)}[3, 4]\} \\ &= \min\{0, -20 + 12\} \\ &= -10 \end{aligned}$$

This would cause the algorithm to terminate at Step 2 with the conclusion that a negative dicycle involving node 4 is present in the graph.

As usual, final $d[k, \ell]$ labels allow us to retrieve the dicycle if needed. Starting at node 4 in the table for $t = 3$,

$$d[4, 4] = 3$$

$$d[4, 3] = 3$$

$$d[4, 2] = 3$$

indicating the negative dicycle 4–2–3–4.

9.5 SHORTEST PATH FROM ONE NODE TO ALL OTHERS WITH COSTS NONNEGATIVE: DIJKSTRA

The Bellman–Ford and Floyd–Warshall algorithms of Sections 9.3 and 9.4 can solve any of the shortest path models presented in Section 9.1 and any of the others we will encounter in Sections 9.6 and 9.7. They require only that the given graph contains no negative dicycle. Still, those two algorithms are not the most efficient option when given graphs satisfy further assumptions. In this section we develop the algorithm credited to E. W. Dijkstra, which is considerably more efficient for cases where we need shortest paths from one node to all others, and all costs are nonnegative ($c_{i,j} \geq 0$). Algorithm 9C provides a formal statement.

ALGORITHM 9C: ONE TO ALL (NONNEGATIVE COSTS); DIJKSTRA SHORTEST PATHS

Step 0: Initialization. With s the source node, initialize optimal path lengths

$$\nu[i] \leftarrow \begin{cases} 0 & \text{if } i = s \\ +\infty & \text{otherwise} \end{cases}$$

Then mark all nodes temporary, and choose $p \leftarrow s$ as the next permanently labeled node.

Step 1: Processing. Mark node p permanent, and for every arc/edge (p, i) leading from p to a temporary node, update

$$\nu[i] \leftarrow \min\{\nu[i], \nu[p] + c_{p,i}\}$$

If $\nu[i]$ changed in value, also set $d[i] \leftarrow p$.

Step 2: Stopping. If no temporary nodes remain, stop; values $\nu[i]$ now reflect the required shortest path lengths.

Step 3: Next Permanent. Choose as next permanently labeled node p a temporary node with least current value $\nu[i]$, that is,

$$\nu[p] = \min\{\nu[i] : i \text{ temporary}\}$$

Then return to Step 1.

Permanently and Temporarily Labeled Nodes

A graph with nonnegative arc/edge costs certainly contains no negative total length dicycles. Thus functional equations [\[9.12\]](#) apply.

What is new about Dijkstra Algorithm 9C is how values satisfying functional equations are computed. Bellman–Ford Algorithm 9A simply evaluates functional equations for all nodes on all iterations. Thus it processes all the inbound arcs and edges at a node many times while computing the minimum in [9.12].

Dijkstra’s method processes outbound arcs and edges instead of inbounds. Much more important, it processes each arc/edge only once. Each major iteration makes one new node p permanent.

Definition 9.22 | Once a node is classified **permanent** by Dijkstra Algorithm 9C, its $\nu[p]$ and $d[p]$ labels never change again. Nodes that are not yet permanently labeled are classified **temporary**.

As each new permanent node p is selected, we correct

$$\nu[i] \leftarrow \min \{ \nu[i], \nu[p] + c_{p,i} \}$$

at all temporarily labeled neighbors i reachable from p . Knowing that the labels for p are final, we need never again consider arc/edge (p, i) in our search for the final value of $\nu[i]$. This finalizing of a node label at each step is what makes Algorithm 9C the best for its class.

Principle 9.23 | Dijkstra Algorithm 9C is the most efficient method available for computing shortest paths from one node to all others in (general) graphs and digraphs having all arc/edge costs nonnegative.

Of course, even faster schemes can be developed if the given graph or digraph has special features beyond nonnegative costs (see Section 9.6).

Least Temporary Criterion for Next Permanent Node

It should be obvious that the heart of Dijkstra’s algorithm is a rule for selecting the next temporary node to make permanent. The one required is elegantly simple.

Principle 9.24 | Each iteration of Dijkstra Algorithm 9C selects as the new permanently labeled node p a temporary node of minimum $\nu[i]$.

All current $\nu[i]$ on temporary nodes are examined, and a p is selected with

$$\nu[p] = \min \{ \nu[i] : i \text{ temporary} \}$$

Dijkstra Algorithm Solution of the Texas Transfer Application

Before investigating why principle [9.24] works, we will apply Dijkstra Algorithm 9C to the Texas Transfer Application reproduced in Figure 9.9. Table 9.3 provides computational details.

Solution begins in exactly the same way as previous methods. Values $\nu[i] \leftarrow \infty$ except at the source node 3 with $\nu[3] \leftarrow 0$. All nodes start temporary. Source 3 is automatically the first permanent.

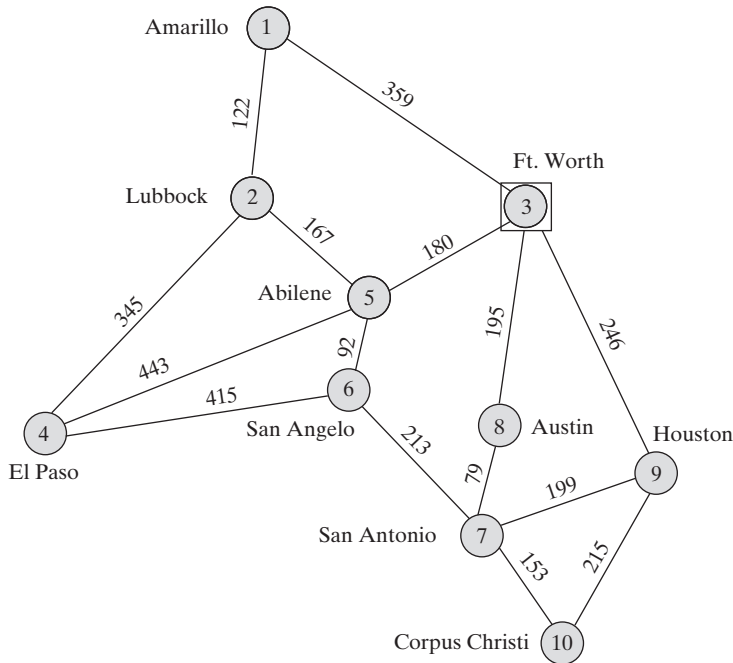


FIGURE 9.9 Texas Transfer Application Network

Processing a new permanent node p means checking the $\nu[i]$ values of all temporary nodes i reachable from p in one step. For $p = 3$ these include nodes 1, 5, 8, and 9. Thus

$$\begin{aligned}\nu[1] &\leftarrow \min\{\nu[1], \nu[3] + c_{3,1}\} \\ &= \min\{\infty, 0 + 359\} \\ &= 359\end{aligned}$$

$$\begin{aligned}\nu[5] &\leftarrow \min\{\nu[5], \nu[3] + c_{3,5}\} \\ &= \min\{\infty, 0 + 180\} \\ &= 180\end{aligned}$$

$$\begin{aligned}\nu[8] &\leftarrow \min\{\nu[8], \nu[3] + c_{3,8}\} \\ &= \min\{\infty, 0 + 195\} \\ &= 195\end{aligned}$$

$$\begin{aligned}\nu[9] &\leftarrow \min\{\nu[9], \nu[3] + c_{3,9}\} \\ &= \min\{\infty, 0 + 246\} \\ &= 246\end{aligned}$$

Corresponding labels $d[1] \leftarrow d[5] \leftarrow d[8] \leftarrow d[9] \leftarrow 3$ because all four $\nu[i]$ changed from 3.

It is now time to apply criterion [9.24](#). The next permanent node p must be a temporary i of minimum $\nu[i]$. With all nodes temporary except 3,

TABLE 9.3 Dijkstra Algorithm Solution of Texas Transfer Application

p	$\nu[1]$	$\nu[2]$	$\nu[3]$	$\nu[4]$	$\nu[5]$	$\nu[6]$	$\nu[7]$	$\nu[8]$	$\nu[9]$	$\nu[10]$
(init)	∞	∞	0	∞	∞	∞	∞	∞	∞	∞
3	359		(perm)		180			195	246	
5		347		623	(perm)	272				
8							274	(perm)		
9									(perm)	461
6						(perm)				
7							(perm)			427
1	(perm)									
2		(perm)								
10										(perm)
4				(perm)						
(final)	359	347	0	623	180	272	274	195	246	427
p	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$	$d[7]$	$d[8]$	$d[9]$	$d[10]$
3	3				3			3	3	
5		5		5		5				
8							8			
9										9
6										7
7										
1										
2										
10										
4										
(final)	3	5	-	5	3	5	8	3	3	7

$$\begin{aligned} & \min \{ \nu[1], \nu[2], \nu[4], \nu[5], \nu[6], \nu[7], \nu[8], \nu[9], \nu[10] \} \\ & = \min \{ 359, \infty, \infty, 180, \infty, \infty, 195, 246, \infty \} \\ & = 180 \end{aligned}$$

The smallest value occurs at node 5 and $p = 5$.
 Correcting labels on outbound edges gives

$$\begin{aligned} \nu[2] & \leftarrow \min \{ \nu[2], \nu[5] + c_{5,2} \} \\ & = \min \{ \infty, 180 + 167 \} \\ & = 347 \\ \nu[4] & \leftarrow \min \{ \nu[4], \nu[5] + c_{5,4} \} \\ & = \min \{ \infty, 180 + 443 \} \\ & = 623 \\ \nu[6] & \leftarrow \min \{ \nu[6], \nu[5] + c_{5,6} \} \\ & = \min \{ \infty, 180 + 92 \} \\ & = 272 \end{aligned}$$

Corresponding $d[2] \leftarrow d[4] \leftarrow d[6] \leftarrow 5$.

To choose the next p , we examine the remaining temporary nodes:

$$\begin{aligned} & \min \{ \nu[1], \nu[2], \nu[4], \nu[6], \nu[7], \nu[8], \nu[9], \nu[10] \} \\ & = \min \{ 359, 347, 623, 272, \infty, 195, 246, \infty \} \\ & = 195 \end{aligned}$$

Here $p = 8$, leading to $\nu[7] \leftarrow 195 + 79 = 274$ and $d[7] \leftarrow 8$.

The next permanent node is $p = 9$ because

$$\begin{aligned} & \min \{ \nu[1], \nu[2], \nu[4], \nu[6], \nu[7], \nu[9], \nu[10] \} \\ & = \min \{ 359, 347, 623, 272, 274, 246, \infty \} \\ & = 246 \end{aligned}$$

Processing yields

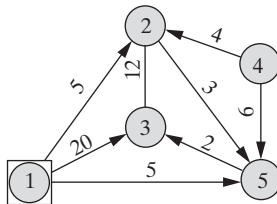
$$\begin{aligned} \nu[7] & \leftarrow \min \{ \nu[7], \nu[9] + c_{9,7} \} \\ & = \min \{ 274, 246 + 199 \} \\ & = 274 \\ \nu[10] & \leftarrow \min \{ \nu[10], \nu[9] + c_{9,10} \} \\ & = \min \{ \infty, 246 + 215 \} \\ & = 461 \end{aligned}$$

Notice that $\nu[7]$ did not change. Thus only label $d[10] \leftarrow 9$.

Remaining processing of Table 9.3 follows in a similar way. Successive iterations make permanent nodes 6, 7, 1, 2, 10, and 4. When all nodes are permanent, the algorithm terminates. Final $\nu[i]$ now represent the required lengths of shortest paths from source node 3.

EXAMPLE 9.13: APPLYING DIJKSTRA’S ALGORITHM

Apply Dijkstra Algorithm 9C to compute the length of a shortest path from node $s = 1$ in the following graph to every other node.



Solution: Paralleling Table 9.3, we may summarize computations in tables for $\nu[i]$ and $d[i]$:

p	$\nu[1]$	$\nu[2]$	$\nu[3]$	$\nu[4]$	$\nu[5]$
(init)	0	∞	∞	∞	∞
1	(perm)	5	20		
2		(perm)	17		
5			7		(perm)
3			(perm)		
4				(perm)	
(final)	0	5	7	∞	5

p	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$
1		1	1		1
2			2		
5			5		
3					
4					
(final)	—	1	5	—	1

One new feature of this example occurs with the selection of the second permanent node $p = 2$. Criterion [9.24] requires us to choose a temporary node i of minimum $\nu[i]$, but there is a tie. Both $\nu[2] = 5$ and $\nu[5] = 5$. Either could be selected. The computations above arbitrarily chose $p = 2$.

One other different element is the presence of a node $i = 4$ to which there is no path. As with other shortest path algorithms, this condition is indicated by final $\nu[4] = \infty$.

Recovering Paths

The Texas Transfer application required only shortest path lengths, not the paths themselves. However, decision labels $d[i]$ make it possible to recover any needed paths using principle [9.18]. For example, the shortest path from source node $s = 3$ to $i = 10$ could be traced backward from $i = 10$ as

$$\begin{aligned} d[10] &= 7 \\ d[7] &= 8 \\ d[8] &= 3 \end{aligned}$$

The resulting shortest path is 3–8–7–10.

Justification of the Dijkstra Algorithm

The Dijkstra algorithm designation of permanently labeled nodes certainly speeds computation. But how do nonnegative costs and selection criterion [9.24] make it work?

The key insight is an interpretation of $\nu[i]$ values during computation.

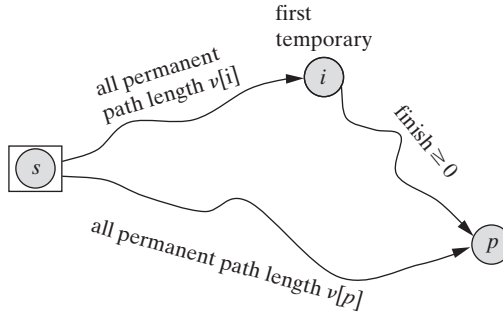
Principle 9.25 After each major iteration, Dijkstra Algorithm 9C values $\nu[i]$ represent lengths of shortest paths from source s to i that use only permanently labeled nodes.

That is, interim results reflect optimal paths through nodes already classified permanent.

For an example, focus on $\nu[10]$ in Table 9.3. There is no path from $s = 3$ to $i = 10$ in the Texas Transfer network of Figure 9.9 that uses only nodes 3, 5, and 8. That is why $\nu[10] = \infty$ through iterations making those nodes permanent. When node 9 becomes permanent on the fourth iteration, an all-permanent path to node 10 does become available. Value $\nu[10] = 461$ to account for that path 3–9–10. Although the path is shortest among nodes classified permanent through 4 iterations, it is not

the shortest overall. Optimal path 3–8–7–10 (length $\nu[10] = 427$) is discovered only 2 iterations later when node 7 is made permanent.

With interpretation [9.25], we can easily see why the node p chosen by criterion [9.24] is ready to become permanent.



Label $\nu[p]$ corresponds to the shortest path to node p using only permanently labeled nodes. Noting that s becomes permanent on the very first iteration, any other path begins among permanent nodes, passes along a subpath to a first temporary i , and completes with an i to p subpath. The length of such a path is

$$(s\text{-to-}i \text{ subpath}) + (i\text{-to-}p \text{ subpath}) \geq \nu[i] + 0$$

because $\nu[i]$ is the length of a shortest all-permanent path to i , and arc/edges costs of the i -to- p subpath are nonnegative. But since p was chosen (principle [9.24]) as the temporary node with smallest ν -value,

$$\nu[i] + 0 = \nu[i] \geq \nu[p]$$

Thus no path can be shorter than the one for $\nu[p]$, and we may justly term it permanent.

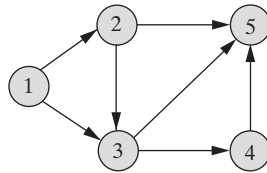
In terms of computational orders on a graph of n nodes (see Section 14.2), this analysis implies $O(n)$ major iterations where each new permanent node is chosen. Then the algorithm processes the new permanent by checking all outbound arcs to temporary nodes—all nodes in the worst case. Summarizing the computation is bounded by $O(n)$ effort per iteration, and thus $O(n^2)$ steps overall.

9.6 SHORTEST PATHS FROM ONE NODE TO ALL OTHERS IN ACYCLIC DIGRAPHS

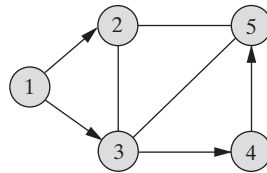
The key to Dijkstra Algorithm 9C's efficiency was a careful selection of permanently labeled nodes, so that arcs needed to be processed only once. Computation becomes even more efficient if a sequence for permanent-node processing can be determined before we begin.

Acyclic Digraphs

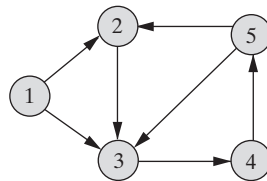
A predetermined sequence is possible if the model arises on an **acyclic digraph**, which is a directed graph with no dicycles. Figure 9.10 illustrates the definition. Part (a) shows a graph that is acyclic because it has only arcs (no edges) and has



(a) *Acyclic*



(b) *Has edges*



(c) *Has dicycles*

FIGURE 9.10 Definition of an Acyclic Digraph

no directed cycles (dicycles). Graphs of parts (b) and (c) fail the definition. The first is only partly directed; for example, it contains edge (2, 3). The second is fully directed, but contains dicycles; one is 2–3–4–5–2.

We can determine whether small graphs such as those of Figure 9.10 are acyclic simply by inspection. For larger graphs we require an easy-to-check condition.

Principle 9.26 A digraph (completely directed graph) is acyclic if and only if its nodes can be numbered so that every arc (i, j) has $i < j$.

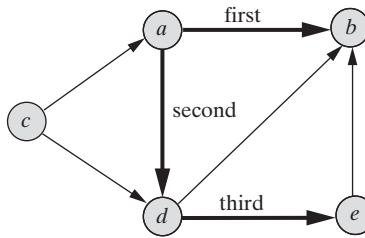
Figure 9.10(a) is already numbered in this way. Each arc goes from a lower-numbered to a higher-numbered node. Notice how this guarantees that the digraph contains no dicycles. Since each arc takes us to a higher-numbered node, no path could ever close a cycle by repeating a node that we have already visited.

Conversely, it is not hard to find a suitable numbering for a digraph that is acyclic.

Principle 9.27 Any acyclic digraph can be numbered as required in principle [9.26] by transiting the graph in depth-first fashion, numbering nodes in decreasing sequence as soon as all their outbound arcs lead to nodes already visited.

Here **depth-first** simply means that we pass to new nodes before backtracking to ones already visited.

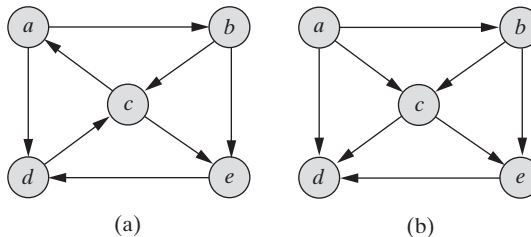
We can illustrate by computing the node numbers of Figure 9.10(a). To avoid confusion, suppose that the graph was given to us with alphabetic node labels as follows:



Beginning arbitrarily at node a , we pass first to node b . Since it has no outbound arcs, we assign it the highest possible number, $b = 5$, and backtrack to a . There, an outbound arc remains to node d , and it has an arc to e . Node e can now be numbered $e = 4$ because it leads only to already visited node $b = 5$. Backtracking to node d , we now assign it number $d = 3$. Similarly, node a is numbered $a = 2$. Having completed all arcs from our origin $a = 2$, we must start again at some still unnumbered node. Here the only choice is c , which immediately becomes node $c = 1$.

EXAMPLE 9.14: DETERMINING WHETHER A DIGRAPH IS ACYCLIC

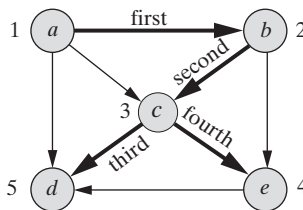
Determine whether each of the following digraphs is acyclic.



Solution:

(a) This graph is not acyclic. For example, it contains dicycle $a-b-c-a$.

(b) This graph is acyclic. To demonstrate that it is, we apply principle [9.27]. Starting at node a , depth-first processing proceeds as follows:



We advance to b , then c , then d before being blocked. Numbering $d = 5$ and backtracking, we can now go to e . It leads nowhere, so we number $e = 4$ and backtrack to c . With no further unexplored neighbors, $c = 3$, and further backtracking yields $b = 2$ and $a = 1$. Node numbers now satisfy condition [9.26].

Shortest Path Algorithm for Acyclic Digraphs

An acyclic digraph certainly has no negative dicycles, because it has no dicycles at all. Thus functional equations [9.12] do apply.

Still, the most important convenience of acyclic digraphs in shortest path computation arises from the numbering of principle [9.26]. If we evaluate functional equations for nodes in that acyclic number sequence, each $\nu[k]$ is permanent as soon as it is computed. This is true because all terms in equations [9.12] involve inbound arcs from lower-numbered nodes with previously fixed $\nu[i]$. Algorithm 9D provides details, and it should be no surprise that it is the most efficient possible.

**ALGORITHM 9D: SHORTEST ONE TO ALL PATHS
(ACYCLIC DIGRAPH) SHORTEST PATHS**

Step 0: Initialization. Number nodes so that each arc (i, j) of the digraph has $i < j$. Then set source s optimal path length

$$\nu[s] \leftarrow 0$$

Step 1: Stopping. Terminate if all $\nu[k]$ have now been fixed. Otherwise, let p be the lowest number of an unprocessed node.

Step 2: Processing. If there are no inbound arcs at node p , set $\nu[p] \leftarrow +\infty$. Otherwise, compute

$$\nu[p] \leftarrow \min\{\nu[i] + c_{i,p} : (i, p) \text{ exists}\}$$

and let $d[p] \leftarrow$ the number of a node i achieving the minimum. Then return to Step 1.

Principle 9.28 Algorithm 9D is the most efficient possible for computing shortest paths from one node to all others in acyclic digraphs.

Careful examination will show that Algorithm 9D works on each arc only once. As a new node p is processed, inbound arcs then are considered. But those arcs cannot be inbound at any other node, so they are processed only once. This makes the computational effort (see Section 14.2) of the algorithm on an acyclic digraph with m arcs $O(m)$. Furthermore, no alternative could do better because any algorithm needs to look at all arcs at least once.

Acyclic Shortest Path Example

None of the models of Section 9.1 are posed on acyclic digraphs, so we will illustrate Algorithm 9D on the digraph of Figure 9.11. Table 9.4 provides results.

Notice that the digraph in Figure 9.11 is already numbered as in principle [9.26], and we will assume that paths are to begin at source $s = 1$. Initialization sets $\nu[1] \leftarrow 0$.

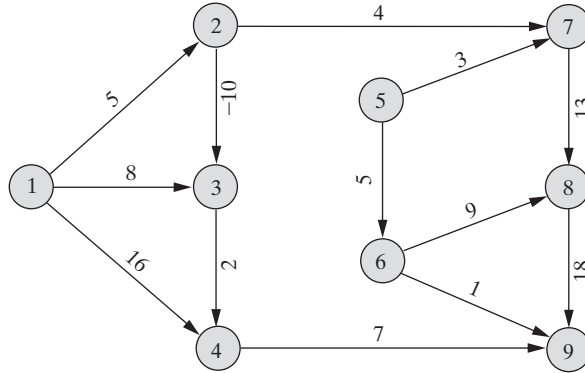


FIGURE 9.11 Acyclic Shortest Path Example

TABLE 9.4 Solution to Acyclic Shortest Path Example

p	$\nu[p]$	$d[p]$	p	$\nu[p]$	$d[p]$
1	0	—	6	∞	—
2	5	1	7	9	2
3	-5	2	8	22	7
4	7	3	9	14	4
5	∞	—			

Node processing now takes place in node number sequence. At node $p = 2$, values are updated as

$$\begin{aligned}\nu[2] &\leftarrow \min\{\nu[1] + c_{1,2}\} \\ &= \min\{0 + 5\} \\ &= 5\end{aligned}$$

with $d[2] \leftarrow 1$. Then at node $p = 3$, we compute

$$\begin{aligned}\nu[3] &\leftarrow \min\{\nu[1] + c_{1,3}, \nu[2] + c_{2,3}\} \\ &= \min\{0 + 8, 5 - 10\} \\ &= -5\end{aligned}$$

with $d[3] \leftarrow 2$. Computation now continues to $p = 4$, $p = 5$, and so on, until we complete at $p = 9$. If required, shortest paths can be retrieved from final $d[k]$ by applying principle [9.18](#).

Longest Path Problems and Acyclic Digraphs

Longest path problems are the maximize analogs of the shortest path problems solved very efficiently by Algorithms 9A through 9D. In principle, all the functional equation and algorithmic technology of Sections 9.2 to 9.7 can be adapted to treat longest path cases simply by substituting *min* by *max* and $+\infty$ by $-\infty$.

Notice, however, that the difficult case of negative dicycles, which none of the dynamic programming shortest path algorithms can handle (principle 9.9), becomes the case of **positive dicycles** in longest path models—dicycles of positive total length. In most instances these are so common that dynamic programming approaches to longest paths are effectively impossible.

The acyclic case, where there are no dicycles at all, does not suffer this limitation. We may directly apply Algorithm 9D after replacing *min* with *max* in Step 1, and $+\infty$ with $-\infty$ in Step 2.

Principle 9.29 Uniquely, among path optimization problems, longest paths in acyclic digraphs are as easy to compute as shortest paths.

To illustrate, return to the example of Figure 9.11, and consider applying the max version of Algorithm 9D to find longest paths from node 1 to all other nodes. The following table shows the results.

<i>p</i>	$\nu[p]$	$d[p]$	<i>p</i>	$\nu[p]$	$d[p]$
1	0	—	6	$-\infty$	—
2	5	1	7	9	2
3	8	1	8	22	7
4	16	1	9	40	8
5	$-\infty$	—			

Values for the first 3 nodes are easily determined because there is only one path to each. Node $p = 4$ presents the first real decision. Length $\nu[4] \leftarrow \max\{16, 8 + 2\} = 16$, coming from $d[4] \leftarrow 1$. At $p = 5$ we encounter a node with no inbound arcs at all. This implies $\nu[5] \leftarrow -\infty$, the worst possible value for a max problem. Continuing in the same way completes solution with values for all the remaining nodes.

9.7 CPM PROJECT SCHEDULING AND LONGEST PATHS

Section 9.1 introduced some of the more common shortest path model forms. However, many other models are closely related, even though they appear at first glance to have nothing to do with computing paths.

Project Management

One of the most commonly occurring such problems arises in the management of large work projects. For planning and control purposes, projects are usually subdivided into a collection of work **activities** that must all be completed to accomplish the project. Each activity has an estimated **duration**,

$$a_k \triangleq \text{time required to accomplish activity } k$$

and a list of predecessor activities. Activity j is a **predecessor** of activity k if activity j must be completed before activity k can begin.

The main issue is to compute an **early start schedule** for the project. That is, we want to know the earliest time that each activity can begin, subject to the requirement that all its predecessor activities have already been completed.

APPLICATION 9.4: WE BUILD CONSTRUCTION

For a contrived but suggestive application, consider We Build Construction's latest project. We Build is developing a 1-story medical office building on available land near a hospital.

Table 9.5 details the project's 9 work activities. For example, the table shows the estimated duration of the heating and air conditioning activity $k = 7$ is $a_7 = 13$ days, and that activity 7 cannot begin until predecessor activities 2 = rough plumbing and 4 = structural members have been completed.

TABLE 9.5 We Build Construction Application Tasks

k	Activity	Duration, a_k (days)	Predecessor Activities
1	Foundation	15	—
2	Rough plumbing	5	—
3	Concrete slab	4	1, 2
4	Structural members	3	3
5	Roof	7	4
6	Rough electrical	10	4
7	Heating and air conditioning	13	2, 4
8	Walls	18	4, 6, 7
9	Interior finish	20	5, 8

To plan materials deliveries and arrange subcontractors to do the various activities, We Build needs a schedule. In particular, they want to know the earliest time after project start that each activity can begin.

CPM Project Networks

To address project scheduling problems with shortest path technology, we require a network or graph. The **critical path method (CPM)** forms such project networks from precedence relationships.

Definition 9.30 **CPM project networks** have special *start* and *finish* nodes, plus one node for each activity. Arcs of zero length connect *start* to all activities without predecessors. Other arcs of length a_k connect each activity node k to all activities of which it is a predecessor, or to *finish* if there are no such nodes.

Figure 9.12 illustrates for the We Build application of Table 9.5. The construction begins by creating a node for each of the 9 activities, plus special *start* and *finish* nodes to represent the beginning and ending of project activity. Each precedence relationship generates an arc with length equal to the predecessor's duration. For example, arc (8,9) represents the fact that activity 8 is a predecessor of activity 9. Its length is $a_8 = 8$ days.

Special arcs from *start* and *finish* complete the digraph. Zero-length arcs lead from *start* to each activity having no predecessors. In the We Build application those are 1 = foundation and 2 = rough plumbing. Arcs of length a_k join activities that are the predecessors of no other activity to *finish*. Here only activity 9 = interior finish qualifies.

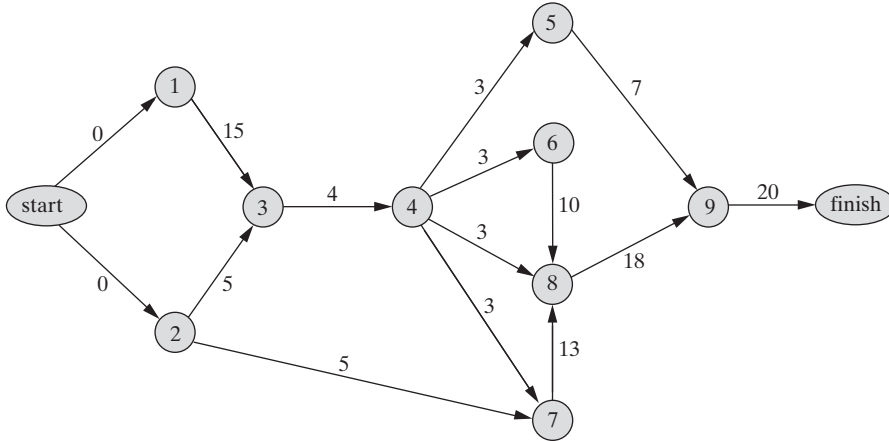


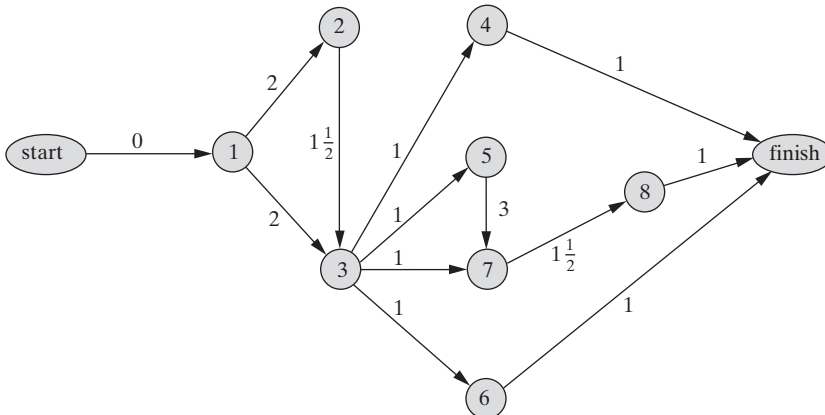
FIGURE 9.12 We Build Construction Project Network

EXAMPLE 9.15: CONSTRUCTING A CPM NETWORK

The following table lists the activities required of a political advance team in arranging a campaign rally. Construct the corresponding CPM project network.

k	Activity	Duration, a_k (days)	Predecessor Activities
1	Contact local party	2	—
2	Find location	$1\frac{1}{2}$	1
3	Arrange date and time	1	1, 2
4	Notify news media	1	3
5	Arrange sound system	3	3
6	Coordinate policy security	1	3
7	Install speaking platform	$1\frac{1}{2}$	3, 5
8	Decorate platform and site	1	7

Solution: Following construction [9.30], we obtain the following CPM project network:



CPM Schedules and Longest Paths

Focus on activity 3 in Figure 9.12. When is the earliest time it can begin? One limit comes from predecessor activity 1 with duration 15. Another derives from predecessor activity 2 with duration 5. Both must be completed before activity 3 can begin. Since neither has a predecessor, both can begin at time 0. Thus the earliest start time for activity 3 is day

$$\max\{a_1, a_2\} = \max\{15, 5\} = 15$$

Notice that the constraint imposed by predecessor 1 corresponds to path *start–1–3* in Figure 9.12, and that of predecessor 2 corresponds to path *start–2–3*. The longest such path defines the **early start time** for activity 3. This is true in general.

Principle 9.31 | The earliest start time of any activity k in a project equals the length of the longest path from *start* to node k in the corresponding project network.

The longest path to node k is the longest succession of activities that must complete before activity k can begin.

Critical Paths

Such longest paths are also called **critical paths**, from which the Critical Path Method derives its name, because delay in any activity along the critical path to node k will delay the start of activity k . The most important is the critical path from *start* to *finish*, which links activities that could delay completion of the entire project if they extend beyond their planned durations. Early start times and critical paths for all activities in the We Build project are shown in the following:

k	Activity	Early Start	Critical Path
1	Foundation	0	start–1
2	Rough plumbing	0	start–2
3	Concrete slab	15	start–1–3
4	Structural members	19	start–1–3–4
5	Roof	22	start–1–3–4–5
6	Rough electrical	22	start–1–3–4–6
7	Heating and air conditioning	22	start–1–3–4–7
8	Walls	35	start–1–3–4–7–8
9	Interior finish	53	start–1–3–4–7–8–9

In a similar way the least possible duration of a project corresponds to the longest *start-to-finish* path.

Principle 9.32 | The minimum time to complete a project equals the length of the longest or critical path from *start* to *finish* in the corresponding project network.

For We Build, this project critical path runs

$$\textit{start-1-3-4-7-8-9-finish}$$

with a total length of 73. Thus the minimum time to complete the building detailed in Table 9.5 is 73 days.

EXAMPLE 9.16: IDENTIFYING CRITICAL PATHS

Use inspection to determine the earliest start times and critical paths for all activities of the political advance project in Example 9.15. Also determine the minimum project duration and the activities which, if delayed, will delay completion of the whole project.

Solution: Inspecting for the longest paths in the network of Example 9.15, start times and critical paths are as follows:

<i>k</i>	Activity	Early Start	Critical Path
1	Contact local party	0	start-1
2	Find location	2	start-1-2
3	Arrange date and time	3½	start-1-2-3
4	Notify news media	4½	start-1-2-3-4
5	Arrange sound system	4½	start-1-2-3-5
6	Coordinate police security	4½	start-1-2-3-6
7	Install speaking platform	7½	start-1-2-3-5-7
8	Decorate platform and site	9	start-1-2-3-5-7-8

Minimum project duration will correspond to the longest *start-to-finish* path

$$start-1-2-3-5-7-8-finish$$

of length 10 days. Thus the activities that must complete as planned if the project is not to be delayed are activities 1, 2, 3, 5, 7, and 8.

Computing an Early Start Schedule for the We Build Construction Application

Algorithm 9E details a longest path scheme for computing early start times. Table 9.6 shows the results from application of Algorithm 9E to compute an early start schedule for our We Build application. Notation now signifies the following:

$$v[k] \triangleq \text{length of the longest path from source } start \text{ to node } k$$

$$d[k] \triangleq \text{immediate predecessor of node } k \text{ in a longest path}$$

As before, computation begins by initializing source node time $v[start] \leftarrow 0$. Then values for other nodes are established in increasing number order. First,

$$\begin{aligned} v[1] &\leftarrow \max\{v[start] + a_{start}\} \\ &= \max\{0 + 0\} = 0 \end{aligned}$$

ALGORITHM 9E: CPM EARLY START SCHEDULING

Step 0: Initialization. Number activity nodes so that each precedence arc (i, j) of the CPM project network has $i < j$. Then initialize the schedule time of the project *start* node as

$$\nu[start] \leftarrow 0$$

Step 1: Stopping. Terminate if the early start time of the project *finish* node has been fixed. Otherwise, let p be the lowest number of an unprocessed node.

Step 2: Processing. Compute the activity p early start schedule time

$$\nu[p] \leftarrow \max\{\nu[i] + a_i : i \text{ is a predecessor of } p\}$$

and let $d[p] \leftarrow$ the number of a node i achieving the maximum. Then return to Step 1.

TABLE 9.6 Early Start Schedule for the We Build Application

p	$\nu[p]$	$d[p]$	p	$\nu[p]$	$d[p]$
start	0	—	6	22	4
1	0	start	7	22	4
2	0	start	8	35	7
3	15	1	9	53	8
4	19	3	finish	73	0
5	22	4			

with $d[1] \leftarrow start$. In a similar way, $\nu[2] \leftarrow 0$ and $d[2] \leftarrow start$. Then, at $p = 3$

$$\begin{aligned} \nu[3] &\leftarrow \max\{\nu[1] + a_1, \nu[2] + a_2\} \\ &= \max\{0 + 15, 0 + 5\} = 15 \end{aligned}$$

making $d[3] \leftarrow 1$. The final results in Table 9.6 correspond exactly to the early start schedule detailed above.

Critical paths can be retrieved using optimal $d[k]$. For example, the entire project's critical path

$$start-1-3-4-7-8-9-finish$$

is recovered by backtracking (principle [9.18](#)) from *finish* as

$$\begin{aligned} d[finish] &= 9 \\ d[9] &= 8 \\ d[8] &= 7 \\ d[7] &= 4 \\ d[4] &= 3 \\ d[3] &= 1 \\ d[1] &= start \end{aligned}$$

EXAMPLE 9.17: COMPUTING A CPM EARLY START SCHEDULE

Return to the political advance project of Example 9.15, and apply Algorithm 9E to compute an early start schedule and identify the critical path for the entire project *finish*.

Solution: Following Algorithm 9E, $\nu[p]$ and $d[p]$ are set in increasing node number sequence. Results are summarized in the following table:

p	$\nu[p]$	$d[p]$	p	$\nu[p]$	$d[p]$
start	0	—	5	$4\frac{1}{2}$	3
1	0	start	6	$4\frac{1}{2}$	3
2	2	1	7	$7\frac{1}{2}$	5
3	$3\frac{1}{2}$	2	8	9	7
4	$4\frac{1}{2}$	3	finish	10	8

The critical path from *start* to *finish* is traced $d[\textit{finish}] = 8$, $d[8] = 7$, $d[7] = 5$, $d[5] = 3$, $d[3] = 2$, $d[2] = 1$, and $d[1] = \textit{start}$, yielding

$$\textit{start}-1-2-3-5-7-8-\textit{finish}$$

Late Start Schedules and Schedule Slack

Computations of early start Algorithm 9E implicitly assume that all activities can start at time 0. Early start schedules are then derived by finding the longest path of predecessor activity durations and summing (principle 9.21).

Suppose now that we have a **due date** for the project, that is, a time by which all activity must be completed. For example, our We Build project might be required to be complete on day 80. We can perform longest path computations in reverse to compute a **late start time** for each activity showing the latest it can begin if the due date is to be met.

Definition 9.33 The latest start time of any activity k in a project equals the due date less the length of a longest path from k to the *finish* node in the corresponding CPM network.

To find this late start schedule, we simply reverse the sequence of computations in Algorithm 9E. Table 9.7 illustrates for the We Build application with a due date of 80.

Computation begins by fixing the late start schedule for the *finish* node at the due date. Then we advance to the highest-numbered activity, 9. Its late start time is the minimum of those for successor activities less its own duration, or $80 - 20 = 60$. Continuing in this way, the late start time for activity 4 is computed:

$$\begin{aligned}\nu[4] &= \min\{\nu[5], \nu[6], \nu[7], \nu[8]\} - a_4 \\ &= \min\{53, 32, 29, 42\} - 3 \\ &= 26\end{aligned}$$

and so on, until we complete the *start* node.

TABLE 9.7 Late Start Schedule for the We Build application

p	$\nu[p]$	$d[p]$	p	$\nu[p]$	$d[p]$
finish	80	—	4	26	7
9	60	finish	3	22	4
8	42	9	2	17	3
7	29	8	1	7	3
6	32	8	start	7	1
5	53	9			

With both early and late schedules in hand, we can also subtract to determine the schedule slack for each activity.

Definition 9.34 | **Schedule slack** is the difference between the earliest and latest times that an activity can begin.

That is, schedule slack shows how much discretion we have in scheduling any single activity while meeting the overall project due date. Table 9.8 computes such slacks from the We Build application activities in Tables 9.6 and 9.7.

EXAMPLE 9.18: COMPUTING LATE START SCHEDULES AND SLACK

Return to the political advance project of Examples 9.15 and 9.17. Compute the corresponding late start schedule for each activity and its schedule slack assuming that all work must be completed by day 10.

Solution: As above, late start times are set in decreasing activity number sequence. Then differences are computed versus early start times to determine slack. The result is summarized in the following table:

k	Activity	Early Start	Late Start	Slack
1	Contact local party	0	0	0
2	Find location	2	2	0
3	Arrange date and time	$3\frac{1}{2}$	$3\frac{1}{2}$	0
4	Notify news media	$4\frac{1}{2}$	9	$4\frac{1}{2}$
5	Arrange sound system	$4\frac{1}{2}$	$4\frac{1}{2}$	0
6	Coordinate police security	$4\frac{1}{2}$	9	$4\frac{1}{2}$
7	Install speaking platform	$7\frac{1}{2}$	$7\frac{1}{2}$	0
8	Decorate platform and site	9	9	0

Acyclic Character of Project Networks

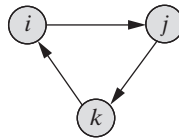
Positive dicycles cannot occur in acyclic digraphs, which have no dicycles at all. We can do CPM scheduling and the associated longest path computations because CPM networks fit this acyclic assumption.

TABLE 9.8 Schedule Slack for the We Build Application

<i>k</i>	Activity	Early Start	Late Start	Slack
1	Foundation	0	7	7
2	Rough plumbing	0	17	17
3	Concrete slab	15	22	7
4	Structural members	19	26	7
5	Roof	22	53	31
6	Rough electrical	22	32	10
7	Heating and air conditioning	22	29	7
8	Walls	35	42	7
9	Interior finish	53	60	7

Principle 9.35 Every well-formed project network is acyclic.

To see why, recall that *start* nodes in principle [9.30] have only outbound arcs, and *finish* nodes have only inbound arcs. Thus any dicycle would have to run through other nodes (i.e., those for activities). But arcs connecting activity nodes correspond to precedence relationships. Any dicycle $i-j-k-i$



would mean that activity *i* must precede activity *j*, activity *j* must precede activity *k*, and activity *k* must precede activity *i*. This is physically impossible. A project network with a dicycle admits no feasible schedule (if durations $a_k > 0$).

We have already seen in Section 9.6 (principle [9.29]) that either shortest or longest path computations on an acyclic digraph with m arcs require $O(m)$ effort. For the CPM case, arcs are precedence relationships. Thus in the light of [9.35], CPM computations are $O(p)$, where p is the number of precedence relationships that must be enforced.

9.8 DISCRETE DYNAMIC PROGRAMMING MODELS

Elementary discrete **dynamic programming** encompasses more than shortest paths, including a variety of problems that can still be modeled as shortest or longest path problems in a suitable digraph even though they have nothing to do with routes and distances. In this section we explore some of the possibilities.

Sequential Decision Problems

One common feature of problems amenable to dynamic programming is **sequential decision making**—decisions that can be arranged in a clear sequence. Then the decisions can be confronted one by one in sequence. That same sequence produces an acyclic digraph on which a shortest or longest path problem can be posed.

APPLICATION 9.5: WAGNER–WHITIN LOT SIZING

A classic application, due to H. Wagner and T. Whitin, concerns **lot sizing**, that is, planning production in an environment where there is a substantial **setup cost** incurred with each production run. Table 9.9 details an example. For each time period $k = 1, \dots, n$, we know

$r_k \triangleq$ quantity of product required at time k

$s_k \triangleq$ setup cost if production occurs at time k

$p_k \triangleq$ variable, per unit cost of production at time k

$h_k \triangleq$ unit cost of holding goods through period k

TABLE 9.9 Wagner–Whitin Lot Sizing Data

	Period, k					
	1	2	3	4	5	6
Requirement, r_k	10	40	20	5	5	15
Setup cost, s_k	50	50	50	50	50	50
Production cost, p_k	1	3	3	1	1	1
Holding cost, h_k	2	2	2	2	2	2

An optimal solution must carefully balance production and holding costs. Sometimes it will be better to pay the setup cost and produce as and when goods are required. Other times we will produce in one period for the next several, incurring a holding cost of storage and lost investment income on goods not required immediately.

States in Dynamic Programming

For our Wagner–Whitin inventory application the passage of time sequences the required decisions. To form a dynamic programming model, we must characterize states of incomplete solution that capture the history on which each new decision should be based.

Definition 9.36 **States** in dynamic programming characterize conditions of incomplete solution at which decisions should be considered.

If an optimal solution is known for any state, those of subsequent states can build upon it.

Development of a state description for Wagner–Whitin lot sizing begins with a simplifying insight: production needs to occur in an optimal solution only when no inventory is held through the preceding period. That is, we should produce or hold for any period k , but not both. Any solution that combined holding and production could at least be matched by shifting the full r_k to whichever of unit holding cost since the last production and variable production cost p_k was lower.

The result is a definition of states. We may say that the process has reached state k when requirements for periods $\ell < k$ have been fulfilled and there is no on-hand inventory.

Digraphs for Dynamic Programs

A suitable definition of states begins the construction of the digraph underlying any elementary dynamic program.

Principle 9.37 Nodes of the digraph associated with any dynamic program correspond to states of incomplete solution.

The remaining element is **decisions**. What options are available upon reaching any given state, and what state will we reach if a decision is adopted?

Principle 9.38 Arcs of the digraph associated with any dynamic program correspond to decisions. They link state nodes with a subsequent state to which the decision leads.

Figure 9.13 displays the digraph for our Wagner–Whitin lot-sizing application. Here decisions correspond to producing in some period to meet its requirements and, possibly, those of several subsequent periods. For example, the arc from state $k = 3$ to $\ell = 6$ reflects production at period 3 to meet requirements $r_3 = 20, r_4 = 5,$ and $r_5 = 5,$ arriving at period 6 with zero inventory. Its cost is

$$\begin{aligned}
 c_{3,6} &= \text{setup} + \text{production} + \text{holding} \\
 &= s_3 + (r_3 + r_4 + r_5)p_3 + [(r_4 + r_5)h_3 + (r_5)h_4] \\
 &= 50 + (20 + 5 + 5)3 + (5 + 5)2 + (5)2 \\
 &= 170
 \end{aligned}$$

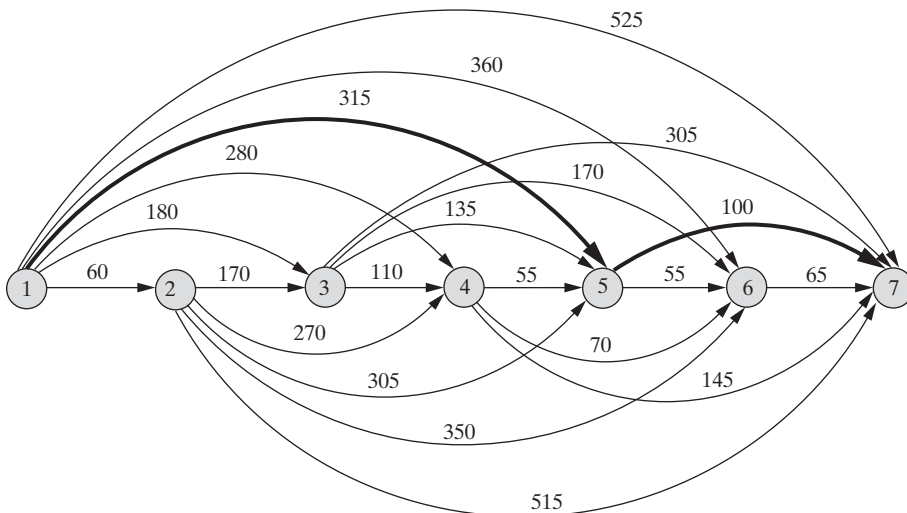


FIGURE 9.13 Digraph for Wagner–Whitin Application

Dynamic Programming Solutions as an Optimal Path

Notice now that each path from node 1 in the digraph of Figure 9.13 to node $(n+1) = 7$ constitutes a lot-sizing plan. For example, path 1–3–6–7 corresponds to production in period 1 for requirements r_1 and r_2 , then in period 3 for requirements r_3 , r_4 , and r_5 , then in period 6 for r_6 . The shortest such path identifies an optimal solution.

Principle 9.39 | Optimal solutions to elementary dynamic programs correspond to shortest or longest paths from a beginning to an ending state in the associated digraph.

They can be solved by applying the appropriate optimal path algorithm.

Like most dynamic programs, our lot-sizing problem is posed over an acyclic digraph because decision arcs at any state lead only to later states. Application of shortest acyclic Algorithm 9D produces the optimal solution highlighted in Figure 9.13. It corresponds to producing in periods 1 and 5 at total cost of 415.

The order of computation (see Section 14.2) for this dynamic programming solution depends on the number of time periods t , here 7. A best choice must be made for each of the $O(t)$ nodes, and option arcs need to be considered from each other t . This makes the Wagner–Whitin solution procedure $O(t^2)$.

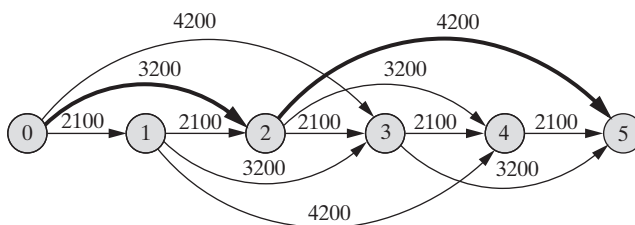
EXAMPLE 9.19: FORMULATING DYNAMIC PROGRAMS

The management team of a 5-year research project is planning a replacement policy for its personal computers. New models may be purchased for \$3000 each. If sold after 1 year, they retain a salvage value of \$1200. If sold after 2 years, the salvage value falls to \$500, and after 3 years the units are obsolete and have no value. Maintenance costs also increase with age. They are estimated at \$300 in the first year of service, \$400 in the second year, and \$500 in the third. Formulate a dynamic program and associated digraph to compute a minimum total cost replacement policy.

Solution: States in this problem correspond to years completed, and decisions relate to keeping computers for 1, 2, or 3 years. Cost of any decision is

$$(\text{purchase cost}) - (\text{salvage value}) + (\text{maintenance cost})$$

The result is the following digraph:



An optimal solution corresponds to a shortest path from time 0 through year 5. An optimum is highlighted in the figure with total cost \$7400.

Dynamic Programming Functional Equations

Although it is usually instructive to draw the digraph associated with a dynamic program, it is common to proceed directly to the underlying functional equation recursions.

Principle 9.40 Dynamic programming **functional equation** recursions encode connections among optimal values for different states of incomplete solution.

The process begins by interpreting the optimal value of any state. For example, with our Wagner–Whitin lot-sizing model,

$$\nu[k] \triangleq \text{minimum cost of fulfilling requirements for periods } \ell < k \\ \text{and arriving at } k \text{ with no inventory}$$

Dynamic programming solution of the problem is then encapsulated in functional equation recursions

$$\begin{aligned} \nu[1] &= 0 \\ \nu[k] &= \min\{\nu[\ell] + c_{\ell,k} : 1 \leq \ell < k\} \quad k = 2, \dots, n + 1 \end{aligned}$$

where

$$c_{\ell,k} \triangleq s_{\ell} + (r_{\ell} + \dots + r_{k-1})p_{\ell} + (r_{\ell+1} + \dots + r_{k-1})h_{\ell} + \dots + (r_{k-1})h_{k-2}$$

For example,

$$\begin{aligned} \nu[4] &= \min\{\nu[1] + c_{1,4}, \nu[2] + c_{2,4}, \nu[3] + c_{3,4}\} \\ &= \min\{\nu[1] + 280, \nu[2] + 270, \nu[3] + 110\} \end{aligned}$$

These are precisely the shortest path functional equations of definition [9.11](#) for the digraph of Figure 9.13.

EXAMPLE 9.20: FORMULATING DYNAMIC PROGRAM RECURSIONS

Formulate functional equations for the dynamic programming model of Example 9.19.

Solution: For this model

$$\nu[k] \triangleq \text{minimum cost of providing computers through year } k$$

This leads to functional equations

$$\begin{aligned} \nu[0] &= 0 \\ \nu[k] &= \min\{\nu[k-3] + c_3, \nu[k-2] + c_2, \nu[k-1] + c_1\} \quad k = 1, \dots, 5 \end{aligned}$$

where c_j is the total cost of keeping a computer for j years.

Dynamic Programming Models with Both Stages and States

A sequence of decisions based on states of incomplete solution lies at the heart of all discrete dynamic programs. However, it is often convenient to distinguish stages of decision making from states of solution.

Definition 9.41 In dynamic programs with both stages and states, **stages** delineate the sequence of required decisions and **states** encode the conditions within which decisions can be considered.

APPLICATION 9.6: PRESIDENT'S LIBRARY

We may illustrate with the design of shelving in the president's library.¹ The retiring president is collecting all his papers in a new presidential library. Materials for its archives are stored in covered cardboard boxes. All are 1.25 feet wide, but their heights vary. Table 9.10 shows the estimated number of boxes at each height.

TABLE 9.10 President's Library Storage Requirements

i	1	2	3	4	5	6	7
Height in feet, h_i	0.25	0.40	0.80	1.00	1.50	2.00	3.0
Thousands of boxes, b_i	10	2	12	30	8	6	4

Boxes in the archives will be placed on metal shelves, with no box stacked on top of another. The issue is how much shelving will be required. Figure 9.14 shows that if, for example, two shelf spacings are employed, the total face area of shelving is

$$\left(\begin{array}{l} \text{linear feet of} \\ \text{boxes with} \\ \text{height} \leq \text{smaller} \\ \text{shelf spacing} \end{array} \right) \left(\begin{array}{l} \text{smaller} \\ \text{shelf} \\ \text{spacing} \end{array} \right) + \left(\begin{array}{l} \text{linear feet of} \\ \text{boxes with} \\ \text{height} > \text{smaller} \\ \text{shelf spacing} \end{array} \right) \left(\begin{array}{l} \text{largest} \\ \text{box} \\ \text{height} \end{array} \right)$$

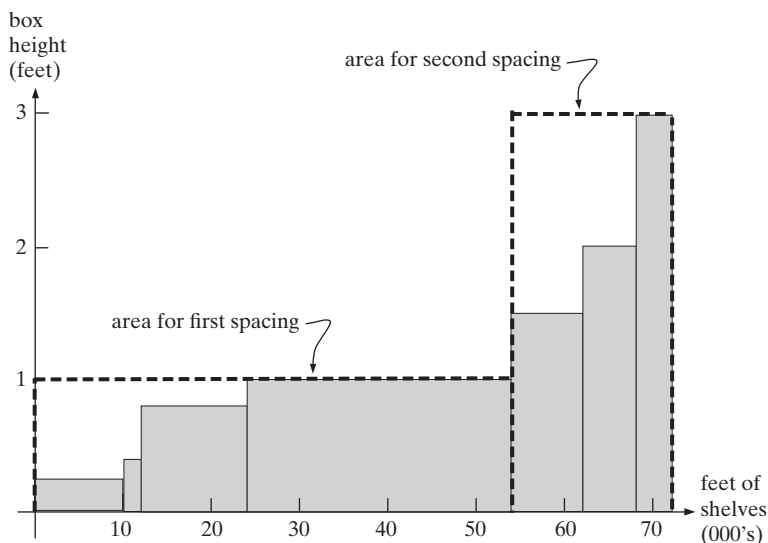


FIGURE 9.14 President's Library Shelving Face Areas

¹Based in part on F. F. Leimkuhler and J. G. Cox (1964), "Compact Book Storage in Libraries," *Operations Research*, 12, 419–427.

The most compact shelving obviously will result if different spacings are provided for each of the 7 sizes in Table 9.10. Still, designers would like to keep storage relatively uniform. Even if they choose to provide 1, 2, . . . , 7 spacings, they want to use the space as efficiently as possible.

Dynamic Programming Modeling of the President’s Library Application

To distinguish stages for a dynamic programming model of the president’s library application, we must consider the sequence of decisions. Here decisions correspond to choices of shelf spacing. Thus we will have one stage for each distinct spacing. More precisely,

stage $k \triangleq$ k th-to-last choice of a spacing

Next come states. On what preconditions should decisions be based? States are often the least intuitive part of forming a dynamic programming model. In our president’s library application, the decision to allocate a new shelf spacing depends on what box sizes have already been provided for. Thus we define states as

state $i \triangleq$ having provided shelves for box sizes 0, 1, . . . , i (0 indicates none)

Decisions, which constitute the third element of any dynamic programming model, are now easy to describe. The choice to provide a spacing for box sizes through j , given that those through i are already accommodated, means commitment of shelf space area

$$c_{i,j} \triangleq 1.25 \left(\sum_{i < s \leq j} b_s \right) h_j$$

For example, accommodating box sizes through $j = 6$, given that those up $i = 3$ have already been provided for, implies an area of

$$\begin{aligned} C_{3,6} &= 1.25(b_4 + b_5 + b_6)h_6 \\ &= 1.25(30 + 8 + 6)(2.0) = 110.0 \text{ thousand square feet} \end{aligned}$$

Figure 9.15 shows the digraph for this dynamic program. Notice that there is one group of nodes for each stage. In accord with principle 9.37, nodes distinguish states. Arcs of the digraph reflect decisions (principle 9.38). For example, the arc from node 3 of stage 2 to node 6 of stage 1 corresponds to making the second-to-last spacing for boxes accommodate sizes 4 to 6. All arcs from the last stage lead to an artificial finish node that corresponds to having provided for all box sizes.

Backward Solution of Dynamic Programs

Readers may wonder why we have chosen to number stages of the president’s library application in reverse or **backward** sequence. This is to highlight the sequence of solution we will apply.

Principle 9.42 | Elementary dynamic programs are often most easily conceptualized in backward sequence, that is, proceeding from final to initial conditions.

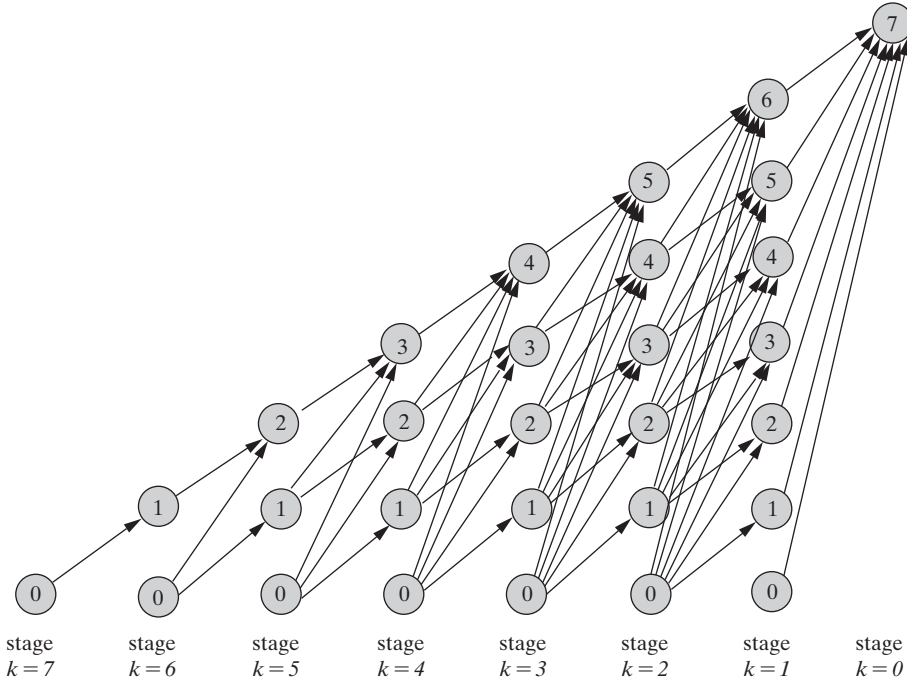


FIGURE 9.15 Digraph for President’s Library Application

Then optimal solutions correspond to shortest or longest paths to the final state in an associated digraph.

The president’s library example is no exception to principle [9.42](#). Its functional equations take the form

$$\left(\begin{array}{c} \text{optimal cost} \\ \text{to finish} \end{array} \right) = \min \left\{ \left(\begin{array}{c} \text{immediate} \\ \text{decision} \\ \text{cost} \end{array} \right) + \left(\begin{array}{c} \text{optimal cost} \\ \text{to finish} \\ \text{from the} \\ \text{resulting} \\ \text{state} \end{array} \right) \right\} \quad (9.1)$$

where the minimization is over the available decisions. More precisely, define

$$v_k[i] \triangleq \text{optimal cost to finish from stage } k, \text{ state } i$$

Then

$$\begin{aligned} v_7[7] &= 0 \\ v_k[i] &= \min \{ c_{i,j} + v_{k-1}[j] : j > i \} \quad k = 1, \dots, 6; \quad i \leq (7 - k) \end{aligned}$$

An optimal solution now corresponds to a shortest path to *finish* rather than the usual shortest path from a single source. However, Algorithm 9D is easily adapted. We need only evaluate the $v_k[\ell]$ in reverse acyclic sequence.

TABLE 9.11 Backward Solution of President’s Library Application

Stage, k	State, i	Optimal Area, $\nu_k[i]$	Next State, $d_k[i]$
finish	—	0.000	0
1	0	270.000	7
	1	232.500	7
	2	225.000	7
	3	180.000	7
	4	67.500	7
	5	37.500	7
2	6	15.000	7
	0	135.000	4
	1	122.500	4
	2	120.000	4
	3	105.000	4
3	4	50.000	6
	5	30.000	6
	0	117.500	4
	1	105.000	4
4	2	102.500	4
	3	87.500	4
	4	45.000	5
	0	108.125	1
5	1	100.000	4
	2	97.500	4
	3	82.500	4
	0	103.125	1
6	1	96.500	3
	2	94.500	3
	0	99.625	1
7	1	95.500	2
	0	98.625	1

Table 9.11 processing begins by setting the cost of the final state $\nu_{\text{finish}} \leftarrow 0$. Then all stage $k = 1$ states are solved in turn, each with only one out arc leading to *finish*.

A more typical step arises if fixing $\nu_2[3]$. This second-to-last shelf spacing (given boxes through size 3 are handled) could provide for size 4, or sizes 4 and 5, or sizes 4 to 6. Thus

$$\begin{aligned}
 \nu_2[3] &\leftarrow \min\{c_{3,4} + \nu_1[4], c_{3,5} + \nu_1[5], c_{3,6} + \nu_1[6]\} \\
 &= \min\{37.5 + 67.5, 71.25 + 37.5, 110.0 + 15.0\} \\
 &= \min\{105.0, 108.75, 125.0\} = 105.0 \text{ thousand square feet}
 \end{aligned}$$

with the optimal decision leading to state 4.

To establish the order of computation (see section 14.2) for dynamic programs like President's Library Application 9.6, we must consider both states and stages. Taking $n \triangleq$ the number of stages, and $m \triangleq$ the maximum number of states per stage, Figure 9.15 shows last stage 1 computation is $O(m)$. Then each subsequent stage k requires consideration of $O(m - k)$ states interacting with $O(m - k - 1)$ states of the next stage. This makes total computation equal to

$$O(m) + \sum_{k \geq 2} O(m - k)O(m - k - 1) = O(nm^2)$$

Multiple Problem Solutions Obtained Simultaneously

What solution is optimal in Table 9.11? There are several, depending on how many distinct shelf spacings are adopted. Optimal shelf areas are

$\nu_1[0] = 270.0$	for $k = 1$ spacing
$\nu_2[0] = 135.0$	for $k = 2$ spacings
$\nu_3[0] = 117.5$	for $k = 3$ spacings
$\nu_4[0] = 108.125$	for $k = 4$ spacings
$\nu_5[0] = 103.125$	for $k = 5$ spacings
$\nu_6[0] = 99.625$	for $k = 6$ spacings
$\nu_7[0] = 98.625$	for $k = 7$ spacings

with corresponding optimal solutions retrievable from corresponding $d_k[i]$ labels.

Solutions for multiple assumptions are a common benefit of dynamic programming solution.

Principle 9.43 | The dynamic programming strategy of exploiting connections among optimal solutions for a number of related problems often implies that optima for multiple problem scenarios are produced by a single shortest or longest path computation.

9.9 SOLVING INTEGER PROGRAMS WITH DYNAMIC PROGRAMMING

Earlier sections of this chapter showed how the dynamic programming approach can be used to efficiently solve a wide variety of shortest path and related problems. The Section 9.8 broadened the treatment with some other dynamic decision-making settings where DP applies. In this section we consider integer and combinatorial models that do not appear on the surface to be dynamic optimizations at all, but can be viewed that way to achieve dynamic programming solution. Every case is unique, but we can illustrate key ideas with the classic Binary Knapsack Problem of Section 11.2.

APPLICATION 9.7: ELECTORAL VOTE KNAPSACK APPLICATION

One classic combinatorial optimization problem where dynamic programming can be useful is the **Binary Knapsack Problem**, that is, a 1-row 0–1 Linear Program. For an example consider the task of a political campaign manager in allocating funds for the last 2 weeks of a presidential election. The Presidents of the United States are elected indirectly by accumulating a majority of **electoral votes** awarded in-block to the candidate in each state who gets the most popular votes there, regardless of how narrow the margin might be. Table 9.12 shows numbers of electoral votes v_j for $n = 7$ target states in which our consultant believes the election race is very close going into its final 2 weeks. The table also shows amounts a_j (in \$ million) the consultant believes would be required to be spent on advertising and activities like get-out-the-vote in each of the targets over the next 2 weeks to insure her candidate wins there. With a total budget of $b = \$10$ million, she wishes to choose the expenditure that will maximize the total resulting electoral vote for her candidate.

Using decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if target } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

the problem is easily modeled as the Integer Linear Program

$$\begin{aligned} \max \quad & \sum_{j=1}^n v_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_j x_j \leq b \\ & \text{all } x_j \text{ binary} \end{aligned}$$

We want to solve it by dynamic programming.

Dynamic Programming Modeling of Electoral Vote Knapsack

Stages in dynamic programs delineate the sequence of decisions to be made. In our knapsack application that is easily seen to be the sequence of targets for which spend-no-spend decisions must be taken. States in the dynamic program should capture the status of partial decision when a particular stage is taken up. Here is where some creativity is needed to fit the given ILP into dynamic decision-making format. A bit of thought will show that for knapsack problem states at a given stage relate to how much is left of the knapsack capacity—here the budget b —when the decision is made for stage (target) j . Denoting such states by i , possible choices for any stage/variable j will either keep the same i (i.e. $x_j = 0$) or be decreased by a_j if $x_j = 1$. Of course, using target j it is not even feasible if $i < a_j$. Table 9.12 shows data we can use to illustrate the approach.

TABLE 9.12 Electoral Vote Knapsack Data

target j	1	2	3	4	5	6	7
electoral votes v_j	9	29	6	10	4	18	13
\$ million cost to win a_j	2	5	1	2	1	4	3

The digraph in Figure 9.16 details the decisions to be considered in computing an optimum. Stages (targets) are arrayed horizontally, and incoming residual budgets (states) show vertically. A final stage at the end tracks ultimate solutions for each state. Arcs from any any node (i, j) show the state change implied by including or not including target j in the state i partial solution. For example, at node $(i, j) = (10, 2)$ one arc continues to node $(10, 3)$ to implement the option of not using variable 2. The other goes down to node $(5, 3)$ because using variable 2 will gain $v_2 = 29$ electoral votes at the cost of reducing the available budget by $a_2 = 5$.

Optimization begins at stage $j = 1$ with the full $i = 10$ million dollars available, and proceeds through each (i, j) pair in turn. We seek a longest path from state 10 of stage 1 to whichever of the final nodes yields the highest final value. An optimal solution is marked by darker arcs in Figure 9.16. Let

$v_j[i] \triangleq$ the value of the best partial solution when stage j is entered in state i

Then functional equations employed at each i and $j > 1$ to find optimal subpaths make

$$v_1[i] \leftarrow 0 \text{ for all } i$$

$$v_j[i] \leftarrow \max\{v_{j-1}[i], v_{j-1}[i + a_{j-1}] + v_{j-1}\} \text{ for all } i, j > 1$$

For example, at stage $j = 5$ the value for state $i = 5$ comes from the best of the value at the same state at stage $j - 1 = 4$ and the value at stage 4, state $5 + a_4 = 5 + 2 = 7$

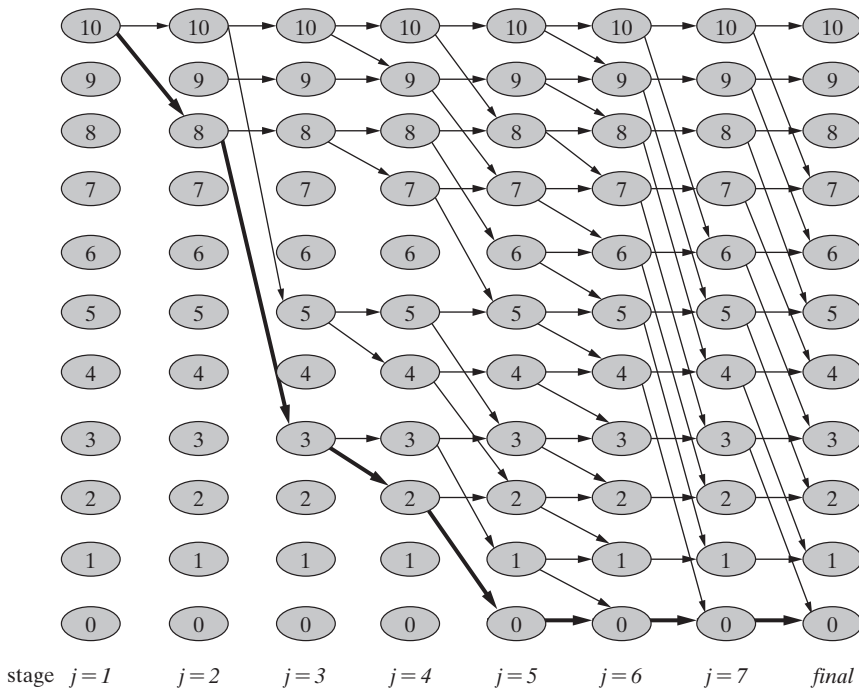


FIGURE 9.16 Electoral Vote Knapsack Dynamic Program

plus the value $v_4 = 10$ for stage 4. Prior calculations make $v_4[5] = 29$ and $v_4[7] = 15$, so that $v_5[5] = \max\{29, 15 + 10\} = 29$.

It is easy to see from the digraph in Figure 9.16 that computation across each of n stages and b states leads to an $O(nb)$ computational order for Binary Knapsack Problems. Still, it is important to highlight that this bounding function involves not just dimensions of instance size like the number of stages n , but also the magnitude of one of the instance constants b , which measures the number of states. As explained in Section 14.2 the standard way to count the size of a constant b is $\log b$, the number of digits needed to input it. This makes our dynamic programming computation above more properly characterized as $O(n \cdot 2^{\log b})$ (assuming binary encoding of b). Like many dynamic programs, the required computational effort will grow rapidly with the magnitude of instance constants. The difference will be important in our coming discussion of computational complexity theory in Chapter 14.

EXAMPLE 9.21: BINARY KNAPSACKS BY DYNAMIC PROGRAMMING

An entrepreneur with \$8 million to invest is considering acquisition of four companies. She estimates the cost of acquiring the four at \$5, \$1, \$2, and \$7 million, respectively, and the present value of owning them at \$8, \$3, \$4, and \$10. Investments are on an “all or nothing” basis (i.e., she must buy entire companies). Model as a dynamic program the problem of deciding which investments she should choose.

- (a) Formulate the entrepreneur’s problem as a Binary Knapsack model.
- (b) Define the states and stages of the corresponding discrete dynamic program to compute an optimum.
- (c) Draw the digraph corresponding to your model of (b).
- (d) Solve the model of (a) by identifying an optimal path in the figure of (c).

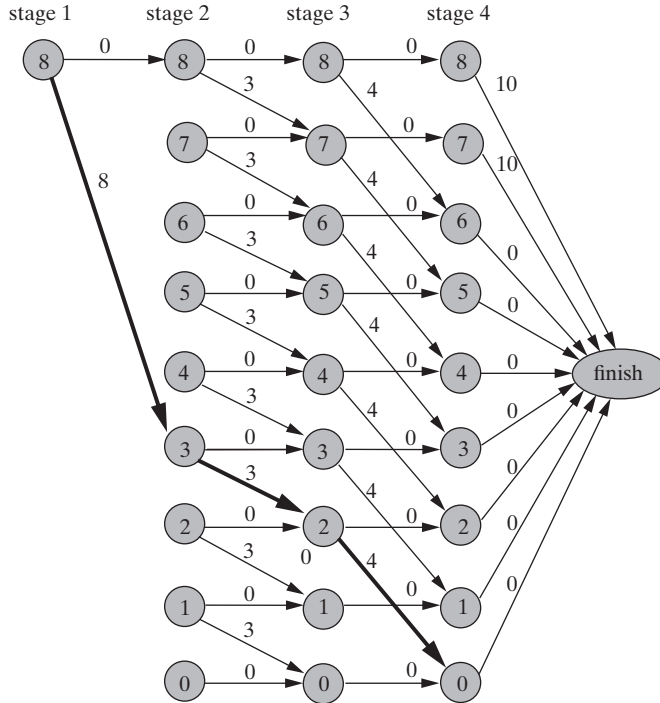
Solution:

(a) Using decision variable $x_j = 1$, if company j is acquired and $= 0$ otherwise, the required BKP formulation is

$$\begin{aligned} \max \quad & 8x_1 + 3x_2 + 4x_3 + 10x_4 \\ \text{s.t.} \quad & 5x_1 + 1x_2 + 2x_3 + 7x_4 \leq 8 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

(b) States i will be the remaining investment \$ million $= 0, 1, \dots, 8$. Stages j will reflect the decision sequence $1, \dots, 4$.

(c) The associated DDP digraph is shown below with one out-arc at each stage j to the same state of the next, indicating $x_j = 0$, and a second to the current state i minus investment for the case of $x_j = 1$.



(d) Solution starts a Stage 4 and proceeds backward to update values for each state and stage. An optimal solution is identified by heavy-weight arcs in the figure. It makes $x_1^* = x_2^* = x_3^* = 1, x_4^* = 0$

9.10 MARKOV DECISION PROCESSES

All the dynamic programming models considered so far—like almost all the other models of this book—have been fully **deterministic**; all parameters are assumed known with certainty at the time decisions must be taken. A wide range of other sequential decision processes involve uncertainty that cannot be ignored. Such **stochastic** problems have some parameters known only in probability. **Markov Decision Processes (MDPs)** are a very broad field of optimization models addressing such cases.

MDPs are named after the famous mathematician A. A. Markov (1856–1922) who pioneered some of the underlying probability theory. Treatment here will be a limited introduction. Much more is available in other references.

Elements of MDP Models

We begin by defining the elements of the MDP models.

Definition 9.44 Markov Decision Processes (MDPs) are discrete dynamic programs with the added feature that the impact of any action/decision cannot be predicted with certainty. Instead, at any state i , any chosen action $k \in K_i$ may lead to a range of possible transitions to next states $j \in J_{i,k}$ with known probabilities $p_{i,k,j} \geq 0$ summing to $= 1$ for each i and k . Parameters $r_{i,k,j}$ the transient reward/cost for decision k at state i when it leads to next state j .

Expected values are usually employed in functional equations to combine outcomes known only in probability in identifying optimal decisions.

Principle 9.45 In terms of states i , actions k , next states j , probabilities $p_{i,k,j}$, and transient rewards $r_{i,k,j}$ as defined in [9.44], expected value functional equations for a maximizing model combine appropriate values at boundary nodes $\{\nu(i)\}$ with main updates

$$\nu(i) \leftarrow \max_{k \in K_i} \{ \sum_{j \in J_{i,k}} [p_{i,k,j} r_{i,k,j} + \nu(j)] \}$$

A minimizing model would replace **max** by **min** in such functional equations.

EXAMPLE 9.22: IDENTIFYING ELEMENTS OF MARKOV DECISION MODELS

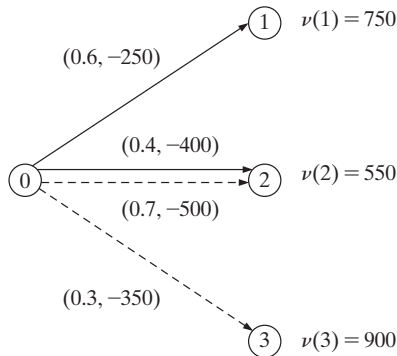
Treasure hunters are seeking to locate and retrieve pots of gold with value estimated at \$750 thousand, \$550 thousand, and \$900 thousand, for sites $i = 1, 2$ and 3 , respectively. In planning the one expedition for this summer, the hunters must choose between 2 never-explored jungle paths. One would require an estimated cost of \$250 thousand if it leads to site 1, and \$400 thousand if it reaches site 2, but site 3 is known to not be accessible. The other path would require an estimated cost of \$500 thousand if it leads to site 2, and \$350 thousand if it reaches site 3, but site 1 is known to not be accessible. The difficulty is that, never having explored either path, the explorers do not know which site they may ultimately visit. Path 1 is thought to lead to site 1 with probability 0.6 and to site 2 with probability 0.4. Similarly, path 2 will lead to site 2 with probability 0.7 and to site 3 with probability 0.3.

- Taking the current decision point as state $i = 0$, and states $i = 1, 2, 3$ for the treasure pots, identify all the elements (and subsets) of definition [9.44] to formulate this gold hunters' problem as an MDP.
- Sketch a digraph depicting your model of part (a) over nodes for 4 states and arcs for possible transitions. Use solid-line arcs for transitions with decision 1 and dashed-line arcs for transitions of decision 2. Also label all transition arcs with their cost and probability, and show the boundard value for the 3 gold-pot states.
- Form and solve the functional equation for state $i = 0$, and determine the optimal action.

Solution:

(a) States $i = 1, 2, 3$ are boundary ones with no decisions to make and (maximize problem) values $\nu(1) = 750$, $\nu(2) = 550$, and $\nu(3) = 900$. At state $i = 0$, decision set $K_0 = [1 \text{ for path 1 and } 2 \text{ for path 2}]$. Possible results for decision $k = 1$ in set $J_{0,1}$ are $j = 1$ with probability $p_{0,1,1} = 0.6$ and cost (negative reward) $r_{0,1,1} = -500$, versus $j = 2$ with probability $p_{0,1,2} = 0.4$ and cost $r_{0,1,2} = -350$. Results for decision $k = 2$ in set $J_{0,2}$ are $j = 2$ with probability $p_{0,2,2} = 0.7$ and cost $r_{0,2,2} = -500$, versus $j = 3$ with probability $p_{0,2,3} = 0.3$ and cost $r_{0,2,3} = -350$.

(b) The required digraph takes the form:



(c) Using boundary values at the 3 gold-pot states, and following [9.45], the functional equation for state $i = 0$ is

$$\begin{aligned} \nu(0) &= \max\{0.6(750 - 250) + 0.4(550 - 400), 0.7(550 - 500) + 0.3(900 - 350)\} \\ &= \max\{360, 200\} = 360 \end{aligned}$$

and decision/action 1 is optimal.

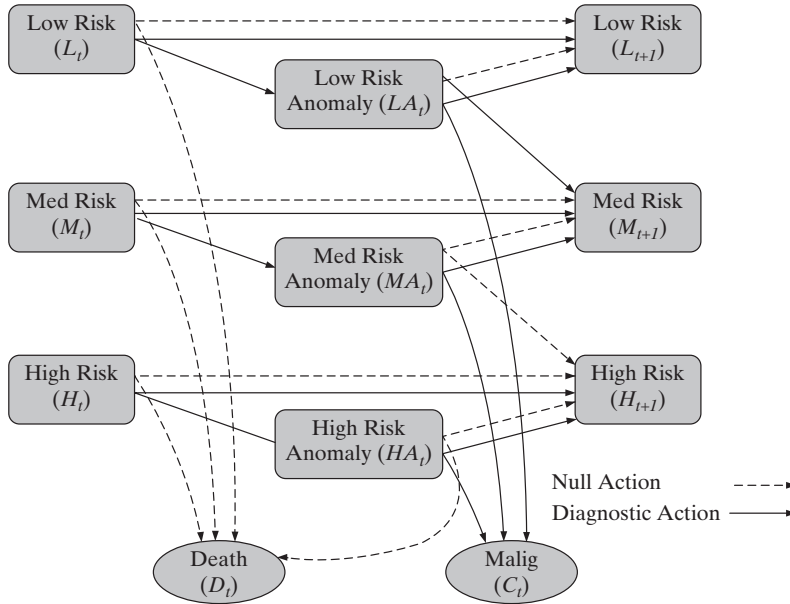
We can now illustrate with a more realistic example.

APPLICATION 9.8: RISK AND DETECTION OF BREAST CANCER

A major challenge for physicians aiding patients in prevention and detection of breast cancer is to decide what diagnostic steps to take at various states of patient risk and age in order to detect breast cancer early without undue diagnostic procedures. Real applications of MDP have added structure to these prevention and treatment decisions, and we will illustrate here with a simplified model posed over fictional parameter values.

Our model divides patient ages into 4 decision epochs $t = 1, \dots, 4$ corresponding to ages in the 40s, 50s, 60s, and 70s, plus a set of terminal states for ages 80

and over. Each of these decision epochs replicates the states, actions, and transitions depicted in the following diagram:



- States:** Underlying patient risk due to family history, co-morbidities, and the like in epoch t is divided among a Low-Risk state L_t , a Medium-Risk state M_t , and a High-Risk state H_t . If a decision at one of these states leads to discovery of an anomaly such as a lump or questionable structure on a mammogram, the implied transition may lead to corresponding anomaly states LA_t , MA_t , and HA_t . Finally, there are Death and Malignancy Care boundary states D_t and C_t in each t , where sick or dying patients exit this risk and detection analysis.
- Actions:** At each decision state there are two kinds of actions. One (indicated in the diagram by dashed lines) undertakes no clinical procedures, and allows nature to run its course. The other (indicated in the diagram by solid lines) proceeds to a diagnostic procedure. For basic states L_t , M_t , and H_t that action is sending the patient for mammography. Results may lead to discovery of an anomaly and transition to corresponding state LA_t , MA_t , or HA_t , or they may prove negative and produce transition to that same risk state at epoch $t + 1$. In Anomaly states LA_t , MA_t , or HA_t , the diagnostic action available is a Biopsy of the anomalous tissue. If the biopsy proves negative, transitions proceed to the same risk categories in epoch $t + 1$. If it tests positive for a malignancy, the transition is to Malignancy Care state C_t .
- Rewards:** The model estimates rewards for states and transitions in terms of **Quality-Adjusted Life Years (QALYs)** predicted for a patient. Generally, early detection of cancerous conditions will add QALYs. However, especially later in life, risks of complication from procedures like biopsies may deduct a penalty reducing the reward that would normally be expected.

Table 9.12 provides full details on the states, actions, transitions, and rewards assumed for all the 5 age epochs in our fictional data. For example, the physician of a low risk (L_1) patient in her 40s ($t = 1$), may choose to do nothing, or to take a mammogram. In the former case, there is a very low probability of death (0.05),

but the vast majority of cases will move on to state L_2 after living out an average of 7.50 QALYs. If a mammogram is ordered, it will detect an anomaly with probability 0.05 leading to state LA_1 , and otherwise it will continue to state L_2 with slightly reduced reward 6.50. The same patient in risk class M_1 will have the same future possibilities, but risk probabilities will be somewhat higher, and rewards lower.

TABLE 9.12 Parameters for Breast Cancer Application

State i	Action k	Transition j	Age 40s ($t = 1$)		Age 50s ($t = 2$)		Age 60s ($t = 3$)		Age 70s ($t = 4$)		Age 80 + ($t = 5$)	
			Prob	Rewrd	Prob	Rewrd	Prob	Rewrd	Prob	Rewrd	Rewrd	
L_t	None	Unchg	L_{t+1}	0.95	7.50	0.92	6.00	0.89	4.80	0.86	3.84	3.33
		Death	D_t	0.05	2.50	0.08	2.00	0.11	1.60	0.14	1.28	
	Mgram	Unchg	L_{t+1}	0.95	6.50	0.92	5.20	0.89	4.16	0.86	3.33	
		Anomly	LA_t	0.05	2.17	0.08	0.72	0.11	0.65	0.14	0.59	
M_t	None	Unchg	M_{t+1}	0.93	5.50	0.90	4.13	0.87	3.09	0.14	2.32	3.16
		Death	D_t	0.07	5.20	0.10	1.03	0.13	0.77	0.16	0.58	
	Mgram	Unchg	M_{t+1}	0.93	4.88	0.90	4.39	0.87	3.95	0.84	3.16	
		Anomly	MA_t	0.07	2.44	0.10	2.19	0.13	1.97	0.16	1.58	
H_t	None	Unchg	H_{t+1}	0.91	3.50	0.87	2.63	0.83	1.97	0.79	1.48	1.87
		Death	D_t	0.09	0.88	0.13	0.66	0.17	0.49	0.21	0.37	
	Mgram	Unchg	H_{t+1}	0.91	3.66	0.87	2.93	0.83	2.34	0.79	1.87	
		Anomly	HA_t	0.09	1.83	0.13	1.46	0.17	1.17	0.21	0.94	
LA_t	None	Unchg	L_{t+1}	0.70	3.47	0.60	3.58	0.50	2.81	0.40	2.19	
		Worsen	M_{t+1}	0.30	1.73	0.40	1.79	0.50	1.40	0.60	1.10	
	Biopsy	Benign	L_{t+1}	0.80	4.44	0.70	3.36	0.60	2.54	0.50	1.91	
		Malign	C_t	0.20	1.65	0.30	1.61	0.40	1.19	0.50	0.88	
MA_t	None	Unchg	M_{t+1}	0.60	1.71	0.50	1.54	0.40	1.38	0.30	1.11	
		Worsen	$H_t + 1$	0.40	1.28	0.50	1.15	0.60	1.04	0.70	0.83	
	Biopsy	Benign	M_{t+1}	0.60	3.07	0.50	2.50	0.40	2.34	0.30	1.82	
		Malign	C_t	0.40	1.22	0.50	1.04	0.60	0.88	0.70	0.66	
HA_t	None	Unchg	H_{t+1}	0.50	1.28	0.40	1.02	0.30	0.82	0.20	0.66	
		Worsen	D_t	0.50	0.96	0.60	0.77	0.70	0.61	0.80	0.49	
	Biopsy	Benign	H_{t+1}	0.60	2.32	0.50	1.70	0.40	1.24	0.30	0.97	
		Malign	C_t	0.40	0.91	0.50	0.69	0.60	0.52	0.70	0.39	
C_t	Absorb			1.00	22.00	1.00	15.00	1.00	8.00	1.00	1.00	
D_t	Absorb			1.00	11.00	1.00	7.50	1.00	4.00	1.00	0.50	

Solution of the Breast Cancer MDP

To compute optimal values and policies for each state of our model, we proceed backward starting with epoch $t = 5$. All 3 risk states there are absorbing with values 3.33, 3.16, and 1.87 QALYs, respectively.

Moving next to age 70s epoch $t = 4$, we first consider anomaly state LA_4 . There the functional equation balances no clinical intervention against a biopsy as

$$\begin{aligned}
 v(LA_4) &\leftarrow \max\{0.40(2.19 + 3.33) + 0.60(1.10 + 3.16), 0.50(2.19 + 3.33) \\
 &\quad + 0.50(0.88 + 1.00)\} \\
 &= \max\{4.76, 3.56\}
 \end{aligned}$$

leading to an optimal choice to take no intervention, with $v(LA_4) = 4.76$. Next, the functional equation for state L_4 is

$$\begin{aligned} v(L_4) &\leftarrow \max\{0.86(3.84 + 3.33) + 0.14(1.28 + 0.50), 0.86(3.33 + 3.33) \\ &\quad + 0.14(0.59 + 4.76)\} \\ &= \max\{6.41, 6.47\} \end{aligned}$$

Selection of second action, mammogram is preferred with value $v(L_4) = 6.47$. Table 9.13 shows the optimal values and actions for all states of the model.

TABLE 9.13 Solution of Breast Cancer Application

State	Age 40s ($t = 1$)		Age 50s ($t = 2$)		Age 60s ($t = 3$)		Age 70s ($t = 4$)		Age 80 + ($t = 5$)	
	Expect Value	Optimal Action	Expect Value	Optimal Action	Expect Value	Optimal Action	Expect Value	Optimal Action	Expect Value	Optimal Action
L_t	23.07	None	16.08	None	10.65	None	6.47	Mgram	3.33	Absorb
M_t	19.46	None	14.21	Mgram	9.95	Mgram	6.07	Mgram	3.16	Absorb
H_t	13.05	Mgram	9.34	Mgram	6.16	Mgram	3.54	Mgram	1.87	Absorb
LA_t	21.14	Biopsy	14.79	Biopsy	9.09	Biopsy	4.76	None		
MA_t	13.02	Biopsy	10.85	Biopsy	7.73	Biopsy	3.17	None		
HA_t	11.68	Biopsy	9.63	Biopsy	6.36	Biopsy	1.83	Biopsy		
C_t	22.00	Absorb	15.00	Absorb	8.00	Absorb	1.00	Absorb		
D_t	11.00	Absorb	7.50	Absorb	4.00	Absorb	0.50	Absorb		

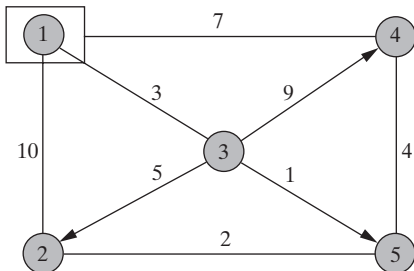
The purpose of constructing an MDP solution analysis like Table 9.13 is to gain broad policy insights. The synthetic numbers we have used in this example yield several. First, mammograms are not recommended for all ages and risk states. Low risk patients L_t do not apparently benefit enough to justify them until their 70s.

In contrast, a mammogram is advised for high risk patients H_t at all ages over 40. Similarly, biopsies are recommended for almost all age groups if an anomaly has placed the patient in states LA_t , MA_t , or HA_t . The exception is ($t = 4$), low and medium risk patients in their 70s. There the complication risk of a biopsy outweighs the likely gain from information it could yield.

None of these insights should be taken as real medical advice. Still, they do demonstrate the enormous potential value of MDP analysis in healthcare settings.

EXERCISES

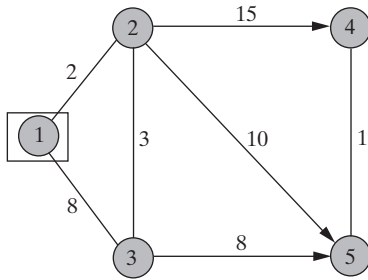
9-1 Consider the following graph.



Numbers on arcs and edges represent lengths.

- ✓ (a) Identify the nodes, arcs, and edges of the graph.
- ✓ (b) Determine whether each of the following sequences is a path of the graph: 1-3-4-5, 2-5-3-4, 1-3-2-5-4, 1-3-4-1-2.
- ✓ (c) Direct the graph (i.e., exhibit a digraph with the same paths as those of the given graph).

9-2 Do Exercise 9-1 for the graph



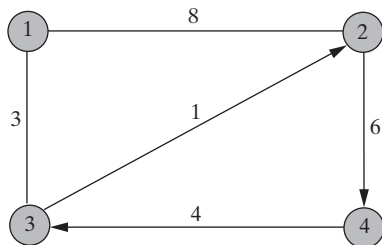
and sequences 3-2-4-5, 3-2-1-4, 1-3-5-2-4, 1-3-2-5.

9-3 Return to the problem of Exercise 9-1.

- ✓ (a) Find (by inspection) shortest paths from node 1 to all other nodes.
- ✓ (b) Verify that every subpath of the optimal 1 to 2 path in part (a) is itself optimal.
- ✓ (c) Detail your optimal solutions of part (a) in functional notation $\nu[k]$ and $x_{i,j}[k]$.
- ✓ (d) Write functional equations for the shortest path problems of part (a).
 - (e) Verify that the $\nu[k]$ of part (c) satisfy functional equations of part (d).
- ✓ (f) Explain why functional equations are sufficient to characterize optimal values $\nu[k]$ for this instance.

9-4 Do Exercise 9-3 for the problem of Exercise 9-2, verifying the optimal 1 to 3 path in part (b).

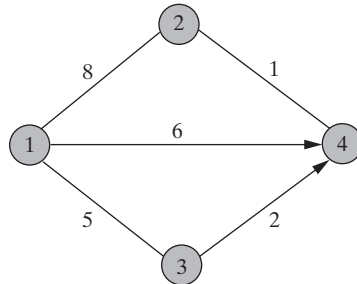
9-5 Consider the following graph.



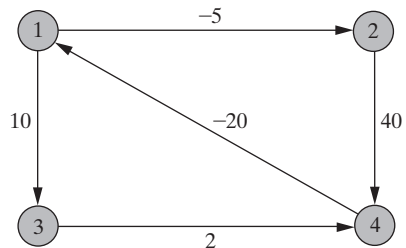
- ✓ (a) Find (by inspection) shortest paths from all nodes to all other nodes.
- ✓ (b) Verify that every subpath of the optimal 1 to 4 path in part (a) is itself optimal.
- ✓ (c) Detail your optimal solutions of part (a) in functional notation $\nu[k][\ell]$ and $x_{i,j}[k][\ell]$.
- ✓ (d) Write functional equations for the shortest path problems of part (a).
 - (e) Verify that the $\nu[k][\ell]$ of part (c) satisfy functional equations of part (d).

- ✓ (f) Explain why functional equations are sufficient to characterize optimal values $\nu[k][\ell]$ for this instance.

9-6 Do Exercise 9-5 for the following graph, verifying the optimal 4 to 2 path in part (b).



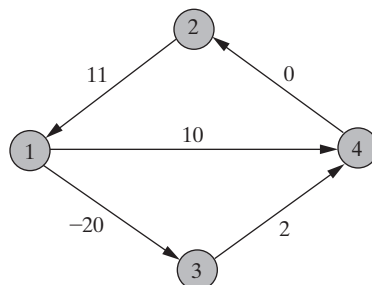
9-7 Consider the following digraph:



Numbers on arcs represent costs.

- ✓ (a) Find (by inspection) shortest paths from node 1 to all other nodes.
- ✓ (b) Enumerate the dicycles of the graph.
- ✓ (c) Determine whether each dicycle is a negative dicycle.
- ✓ (d) How does the presence of negative dicycles in this instance make it more difficult for an algorithm to compute shortest paths?

9-8 Do Exercise 9-7 for the digraph



9-9 Return to the graph of Exercise 9-1, and suppose that we seek shortest paths from node 1 to all other nodes.

- ✔ (a) Explain why Bellman–Ford Algorithm 9A can be employed to compute the required shortest paths.
- ✔ (b) Apply Algorithm 9A to compute the lengths of shortest paths from node 1 to all other nodes.
- ✔ (c) Use $d[k]$ labels of your computations in part (b) to recover all optimal paths.
- ✔ (d) Verify the interpretation of interim labels $\nu^{(t)}[k]$ as lengths of shortest paths using at most t arcs/edges by showing that values at the end of iteration $t = 2$ in your computations of part (b) correspond to lengths of shortest paths using 1 or 2 arcs/edges.
- ✔ (e) Determine the maximum number of arcs/edges in any path of this graph, and explain how this bounds the computation required for Algorithm 9A.

9-10 Do Exercise 9-9 on the graph of Exercise 9-2.

9-11 Use Bellman–Ford Algorithm 9A to identify a negative dicycle in each of the following graphs.

- ✔ (a) The digraph of Exercise 9-7
- ✔ (b) The digraph of Exercise 9-8

9-12 Return to the graph of Exercise 9-5, and suppose that we seek shortest paths from all nodes to all other nodes.

- ✔ (a) Explain why Floyd–Warshall Algorithm 9B can be employed to compute the required shortest paths.
- ✔ (b) Apply Algorithm 9B to compute the length of shortest paths from all nodes to all other nodes.
- ✔ (c) Use $d[k][\ell]$ labels of your computations in part (b) to recover all optimal paths.
- ✔ (d) Verify the interpretation of interim labels $\nu^{(t)}[k][\ell]$ as lengths of shortest paths using only intermediate nodes up to t by showing that values at the end of iteration $t = 2$ in your computations of part (b) correspond to lengths of shortest paths using only nodes 1 and 2 as intermediates.

9-13 Do Exercise 9-12 on the graph of Exercise 9-6.

9-14 Use Floyd–Warshall Algorithm 9B to identify a negative dicycle in each of the following graphs.

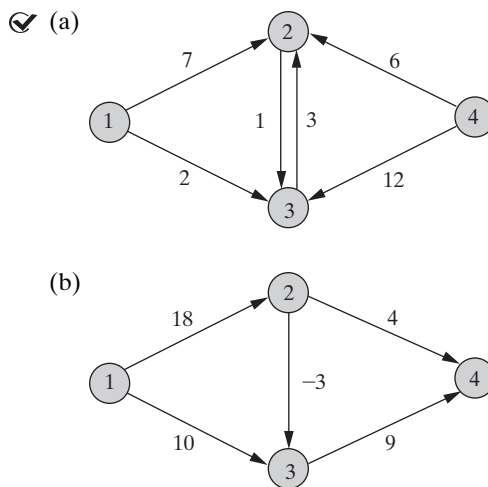
- ✔ (a) The digraph of Exercise 9-7
- ✔ (b) The digraph of Exercise 9-8

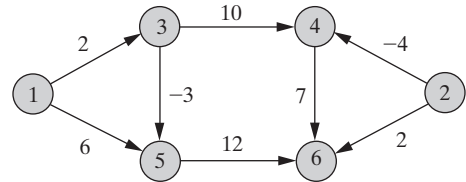
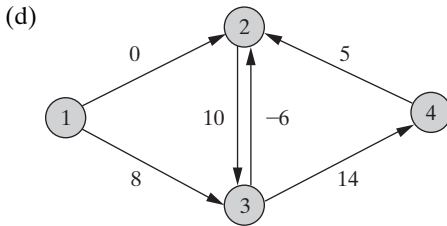
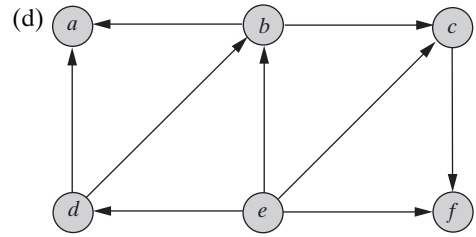
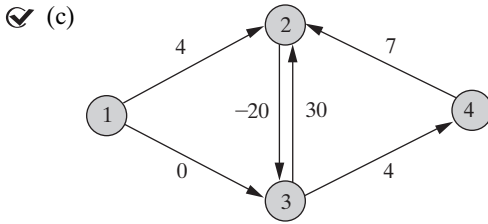
9-15 Return to the graph of Exercise 9-1, and suppose that we seek shortest paths from node 1 to all other nodes.

- ✔ (a) Explain why Dijkstra Algorithm 9C can be employed to compute the required shortest paths.
- ✔ (b) Apply Algorithm 9C to compute the length of shortest paths from node 1 to all other nodes.
- ✔ (c) Use $d[k]$ labels of your computations in part (b) to recover all optimal paths.
- ✔ (d) Verify the interpretation of interim labels $\nu[k]$ as lengths of shortest paths using only permanently labeled nodes by showing that values after two permanent nodes have been processed in your computations of part (b) correspond to lengths of shortest paths using only those nodes (and the destination).

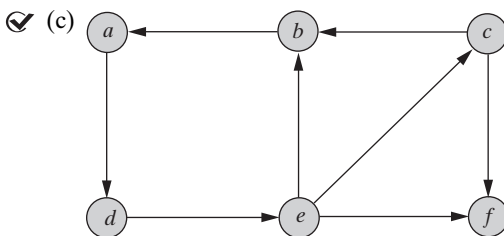
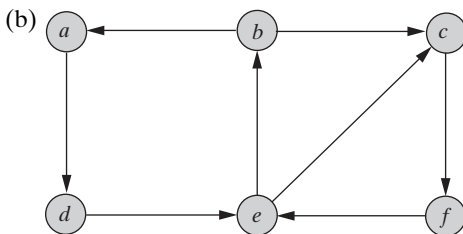
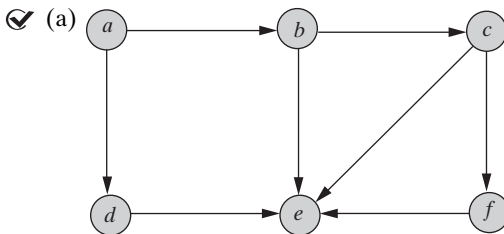
9-16 Do Exercise 9-15 on the graph of Exercise 9-2.

9-17 For each of the following digraphs, determine which of Bellman–Ford Algorithms 9A, Floyd–Warshall Algorithm 9B, and Dijkstra Algorithm 9C could be applied to compute shortest paths from node 1 to all others. Then, if any is applicable, choose the most efficient and apply it to compute the required paths.





9-18 Demonstrate whether each of the following digraphs is acyclic by exhibiting a dicycle or numbering nodes so that every arc (i, j) has $i < j$.

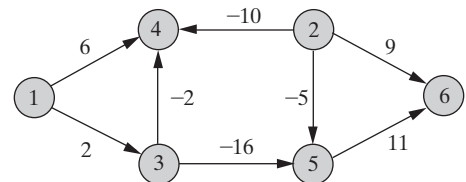


9-19 Consider the following digraph:

Suppose that labels on arcs are cost and that we seek a minimum total cost path from node 1 to all other nodes.

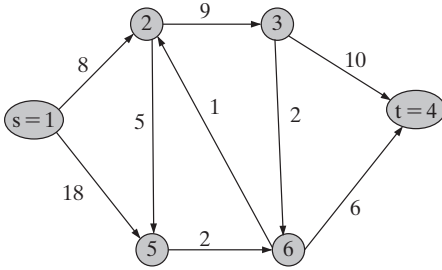
- ✓ (a) Demonstrate that acyclic shortest path Algorithm 9D can be applied to compute the required paths.
- ✓ (b) Use Algorithm 9D to compute the lengths of shortest paths from node 1 to all other nodes.
- ✓ (c) Use $d[k]$ labels from your computations in part (b) to recover all optimal paths.

9-20 Do Exercise 9-19 for the digraph



9-21 Determine whether acyclic shortest path Algorithm 9D could be applied to compute shortest paths from node 1 to all other nodes in each digraph of Exercise 9-17. If so, explain whether it would be more efficient than the Bellman–Ford, Floyd–Warshall, and Dijkstra alternatives, and why.

9-22 The digraph below shows an instance of the shortest path problem. Numbers on the arcs are lengths. We wish to compute a shortest path from the indicated node s to the indicated node t .



- (a) Which of this chapter’s four shortest path Algorithms 9A–9D (Bellman–Ford, Dijkstra, Floyd–Warshall, and Acyclic) should be preferred for this task? Explain why.
- (b) Apply the algorithm recommended in (a) to compute the lengths of shortest paths from node s to all other nodes. Show all details of labeling and updates as the algorithm proceeds. Then follow that algorithm’s backtrack labels to recover (only) an optimal path from s to t .

9-23 The table that follows lists the tasks that must be performed in preparing a simple breakfast. The table also shows the number of minutes each requires and the tasks that must be completed before each can begin.

k	Task	Time	Predecessors
1	Boil water	5	None
2	Get dishes	1	None
3	Make tea	3	1, 2
4	Pour cereal	1	2
5	Fruit on cereal	2	4
6	Milk on cereal	1	4
7	Make toast	4	None
8	Butter toast	3	7

- ✓ (a) Construct the corresponding CPM project network [9.30].
- ✓ (b) Use given activity numbers to verify that your project network is acyclic.
- ✓ (c) Apply CPM scheduling Algorithm 9E to compute early start times for each activity and an early finish time for the entire project.
- ✓ (d) Use $d[k]$ labels of your scheduling computations to identify the activities along a critical path from project start to finish.

- ✓ (e) Compute late start times for each activity assuming that the breakfast must be complete in 10 minutes, and combine with part (c) to determine schedule slacks.

9-24 The table that follows lists the activities that must be performed in organizing a soccer tournament. The table also shows the estimated number of days that each will require and the activities that must be completed before each can begin.

k	Activity	Time	Predecessors
1	Select dates	1	None
2	Recruit sponsors	4	1
3	Set fee	1	2
4	Buy souvenirs	5	1, 2
5	Mail invitations	1	2, 3
6	Wait responses	4	5
7	Plan pairings	1	6
8	Team packets	1	4, 7

Perform (a) through (e) of Exercise 9-23, assuming that the organization must be completed in 13 days.

9-25 Officers of Industrial Engineering student organizations are planning the annual awards banquet. The following table shows the major activities of the project, along with their durations (in days) and predecessors.

No.	Activity	Duration	Predecessors
1	Distribute award ballots	2	None
2	Receive completed ballots	7	1
3	Determine award winners	1	2
4	Reserve banquet room	2	None
5	Send invitations	2	4
6	Receive RSVPs	10	5
7	Choose menu	4	4
8	Hold banquet	1	3, 6, 7

Perform (a) through (e) of Exercise 9-23, assuming that the project must be completed in 17 days.

9-26 Construction of a small two-story house involves the tasks listed in the following table. The table also shows the estimated duration of each task in days and the tasks that must be completed before it can begin.

Task	Time	Predecessors
Foundation (FD)	8	None
Concrete slab (CS)	5	FD
First bearing walls (1B)	3	CS
First internal walls (1I)	4	CS
First finishing (1F)	12	1B, 1I, FL
Second floor (FL)	3	1B
Second bearing walls (2B)	4	2FL
Second internal walls (2I)	5	2FL
Second finishing (2F)	10	2B, 2I, R
Roof (RF)	2	2B

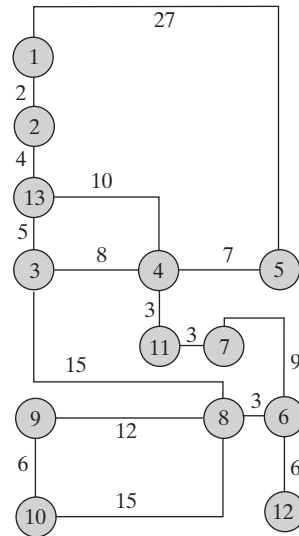
- ✓ (a) Construct the corresponding CPM project network [9.30].
- ✓ (b) Number activity nodes in your project network so that every arc (i, j) has $i < j$.
- ✓ (c) Use CPM scheduling Algorithm 9E with your activity numbering to compute early start times for each activity and an early finish time for the entire project.
- ✓ (d) Use $d[k]$ labels of your scheduling computations to identify the activities along a critical path from project start to finish.
- (e) Compute late start times for each activity assuming that construction must be complete in 35 days, and combine with part (c) to determine schedule slacks.

9-27 The table that follows shows the activities required to construct a new computer laboratory, along with their estimated durations (in weeks) and predecessor activities.

Activity	Time	Predecessors
Order furniture (OF)	1	None
Order computers (OC)	1	None
Order software (OS)	1	OC
Furniture delivery (FD)	6	OF, P
Computer delivery (CD)	3	OC, AF
Software delivery (SD)	2	OS
Assemble furniture (AF)	1	FD
Install computers (IC)	1	CD
Install software (IS)	1	IC, SD
Wire room (W)	2	None
Paint room (P)	1	W

Perform (a) through (e) of Exercise 9-26, assuming that the lab must be completed in 17 weeks.

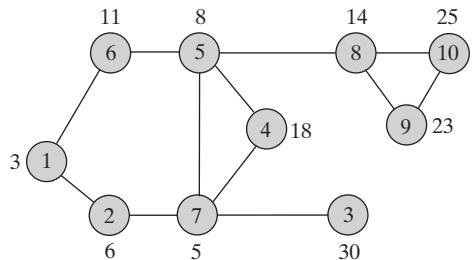
9-28 The figure that follows shows a partially complete layout for a circuit board of a large computer peripheral. Lines show channels along which circuits can be placed, together with the lengths of the channels in centimeters. Several circuits can be placed in a single channel (on different layers).



The last step in the design is to choose the routing from a component to be installed at point 1 to connections at points 8, 10, 11, and 12.

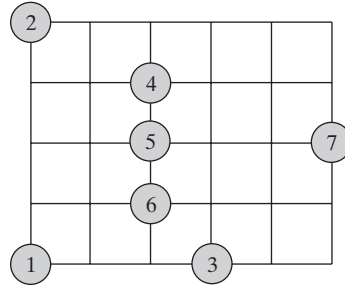
- (a) Explain why this problem can be modeled as a shortest path problem.
- (b) Explain why the most efficient procedure available from this chapter for computing optimal circuit routes is Dijkstra Algorithm 9C.
- ✓ (c) Apply Algorithm 9C to compute optimal circuit routes to the 4 specified points.

9-29 The figure that follows shows the links of a proposed campus computer network. Each node is a computer, and links are fiber-optic cable.



Designers now want to decide how E-mail, which will be broken into standard-length packets, should be routed from Internet gateway node 1 to/from all other nodes. For example, E-mail for node 4 might be transmitted by 1 to 6, then repeated by 6 to 5, then repeated by 5 to 4. Numbers on nodes in the figure indicate minimum times (in nanoseconds) required by the corresponding computer to transmit or receive a message packet. The time to send a packet along any link of the network is the maximum of the times for the associated sending and receiving computers.

points 4, 5, and 6, and a grinding machine at 7. Each grid square indicated is the same size.



- (a) Explain why this problem can be modeled as a shortest path problem, and sketch the graph and edge weights over which optimal paths are to be computed.
- (b) Explain why the most efficient procedure available from this chapter for computing optimal E-mail routes is Dijkstra Algorithm 9C.
- ✓ (c) Apply Algorithm 9C to compute optimal routes to all computers.

9-30 The campus shuttle bus begins running at 7:00 P.M. and continues until 2 A.M. Several drivers will be engaged, but only one should be on duty at any time. If a shift starts at or before 9:00 P.M., a regular driver can be engaged for a 4-hour shift at cost \$50. Otherwise, part-time drivers will be engaged. Some would work 3-hour shifts at \$40 and the rest are limited to 2-hour shifts at \$30.

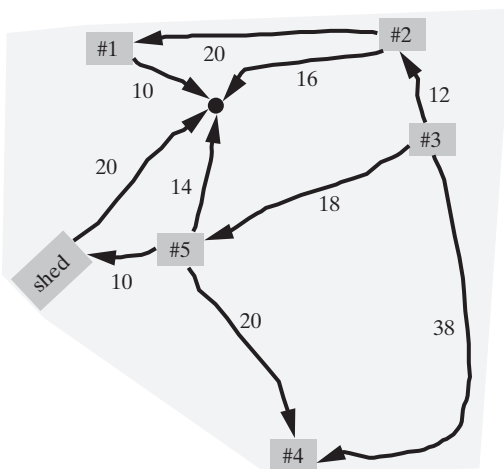
- (a) Show that the problem of computing a minimum total cost for the night shift driver schedule can be modeled as a shortest path problem on a graph with nodes equal to the hours of the night from 7:00 P.M. through 2 A.M. Also sketch the corresponding digraph, and label it with arc lengths.
- ✓ (b) Is your digraph acyclic? Explain.
- (c) Determine which of the algorithms of this chapter would solve your shortest path problem most efficiently, and justify your choice.
- ✓ (d) Apply your chosen algorithm to compute an optimal night shift schedule.

9-31 The machine shop depicted in the figure that follows has a heat treatment workstation at point 1, forges at points 2 and 3, machining centers at

Processing of a camshaft product begins with heat treatment, then goes to any forge, then moves to any machining center, and finishes at the grinding machine. Movement between workstations is rectilinear (i.e., north/south displacement plus east/west).

- (a) Show that the problem of computing a minimum total movement camshaft routing can be modeled as a shortest path problem. Also sketch the corresponding digraph, and label it with arc lengths.
- (b) Is your digraph acyclic? Explain.
- (c) Determine which of the algorithms of this chapter would solve your shortest path problem most efficiently, and justify your choice.
- ✓ (d) Apply your chosen algorithm to compute an optimal movement.

9-32 The figure that follows shows the trails of Littleville’s Memorial Park.



Whenever there is a major event there, heavy hot dog and soft-drink carts are stored in the shed and rolled out along the trails to the five marked locations. Numbers on trail links indicate their length, and arrows show which way is uphill. The work to push a cart uphill on any link is proportional to that length. Carts can also be moved in the opposite direction, but the downhill effort is only half as much.

- (a) Show how the problem of finding the least total-effort-push routes for all carts can be modeled as a shortest path problem. Also sketch the corresponding digraph, and label it with arc lengths.
- (b) Is your digraph acyclic? Explain.
- (c) Determine which of the algorithms of this chapter would solve your shortest path problem most efficiently, and justify your choice.
- ✓ (d) Apply your chosen algorithm to compute optimal move routes.

9-33 A pharmaceutical manufacturer must supply 30 batches of its new medication in the next quarter, then 25, 10, and 35 in successive quarters. Each quarter in which the company makes product requires a \$100,000 setup, plus \$3000 per batch produced. There is no limit on production capacity. Batches can be held in inventory, but the cost is a high \$5000 per batch per quarter. The company seeks a minimum total cost production plan.

- (a) Explain why this problem can be approached by discrete dynamic programming, with states $k = 1, \dots, 5$ representing the reaching of quarter k with all earlier demand fulfilled and no inventory on hand.
- ✓ (b) Sketch the digraph corresponding to the dynamic program structure of part (a). Include costs on all arcs.
- (c) Explain why the feasible production plans correspond exactly to the paths from node $k = 1$ to node $k = 5$ in your digraph.
- ✓ (d) Solve a shortest path problem on your digraph to compute an optimal production plan.
- ✓ (e) Use your computations in part (d) to compute an optimal production plan for the first two quarters.

9-34 Do Exercise 9-33 with all parameters the same except a holding cost of \$1000.

9-35 A copy machine repairman has four pieces of test equipment for which he estimates 25%, 30%, 55%, and 15% chances of using them at his next stop. However, the devices weigh 20, 30, 40, and 20 pounds, respectively, and he can carry no more than 60 pounds. The repairman seeks a maximum utility feasible collection of devices to carry with him.

- (a) Explain why this problem can be approached by discrete dynamic programming, with stages $k = 1, \dots, 4$ representing the four devices and states $w = 0, 10, 20, 30, 40, 50, 60$ in each stage indicating reaching that decision stage with w units of weight limit remaining.
- ✓ (b) Sketch the digraph corresponding to the dynamic program structure of part (a). Include objective function contributions on all arcs.
- (c) Explain why the feasible production plans correspond exactly to the paths from stage $k = 1$, state $w = 60$ to the last stage in your digraph.
- ✓ (d) Solve a longest path problem on your digraph to compute an optimal toolkit.

9-36 Do Exercise 9-35 with all parameters the same except a weight limit of 40 pounds.

9-37 Consider solving the following Knapsack ILP by Discrete Dynamic Programming (DDP):

$$\begin{aligned} \min \quad & 18x_1 + 13x_2 + 20x_3 + 12x_4 \\ \text{s.t.} \quad & 2x_1 + 6x_2 + 4x_3 + 3x_4 \geq 14 \\ & x_1, x_2, x_3, x_4 \in [0, 2] \text{ and integer} \end{aligned}$$

- (a) Define the states and stages of the DDP to compute an optimum.
- (b) Draw a digraph from which an optimum can be computed over the states and stages of (a). How does it differ from similar digraphs for 0-1 Knapsack Problems like Figure 9.16?
- (c) Solve the DDP depicted in (b), as it starts at the Finish node and proceeds backward to update values for each state and stage. Then identify the optimal solution found.

9-38 Repeat Exercise 9-37, increasing the KP instance on the right-hand side to 19.

9-39 Two promising drugs—X14Alpha and X14Beta—are ready for testing to combat rare,

but dangerous virus X14. If one of the drugs proves to have high effectiveness, 500 lives could be saved in the coming winter, but only 200 lives would be saved if the a drug proved to have just moderate effectiveness. Also, the tests may prove inconclusive, which would mean no drugs could be released.

Unfortunately, only 10 patients are available to serve as test subjects. Researchers can test X14Alpha on all 10, or X14Beta on all 10, or split the subjects 5 and 5 between the drugs, with the most successful being considered for release. The following table shows the predicted probabilities of different kinds of test results depending on which subjects are tested.

Test Design	Probability of Proven Effectiveness		
	High	Moderate	Inconclusive
All X14Alpha	0.30	0.40	0.30
All X14Beta	0.40	0.20	0.40
Split	0.15	0.60	0.25

- ✔ (a) Taking states $i = 1, 2, 3$ for the high, moderate, and inconclusive outcomes, identify all the elements of definition [9.44] to formulate this drug testing challenge as a MDP maximizing the expected number of saved lives.
- ✔ (b) Sketch a digraph depicting your model of part (a) over nodes for 3 states and arcs for possible transitions. Clearly label all transition arcs with the decision to which they are attached, the associated probability, and the implied reward.
- ✔ (c) Form the functional equations for all states i .
- ✔ (d) Solve the equations of (c) to compute an the optimal testing action.

9-40 Mindy is playing a gambling game of 3 rounds. She will start with 4 chips, and she can

wager any number of chips she has on hand at each round. With probability 0.45, she will win the bet and receive a number of additional chips equal to her wager. Otherwise, with probability 0.55 she will lose all the chips wagered, and of course, the game is over if she runs out of chips. Mindy wants to choose the betting strategy that will lead to the highest expected number of chips at the end of the 4 rounds.

- (a) Formulate Mindy’s task as a MDP with multiple states and stages, including identifying all the elements of definition [9.44].
- (b) Sketch a digraph depicting your model of part (a). You need not insert all parameter details, but do show exemplars of all transition arcs with the decision to which they are attached, the reward they would realize, and the associated probability.
- (c) Form the functional equations for all states and stages.
- (d) Establish that the optimal (expected value) bet at every state is simply to wager just 1 chip, which produces the expected value overall of 3.70 chips.

9-41 Saw King (SK) is the region’s leading seller of the highest rated model of chain saws. Their challenge is that their present store site has space for only an inventory of 18 of the very large saws. SK is trying to develop a plan for how to make inventory replenishment decisions in that constrained environment. The company can place an order at the end of each week for goods to arrive at the beginning of the next, and the plan they are developing should cover the historically volatile coming 5 weeks. Placing an order costs \$500 plus \$900 per unit purchased, and inventory holding costs \$95 per unit per week. If stock is available, sale of a saw produces revenue of \$2000. Demand is uncertain, but the following table estimates the probability for different numbers of weekly sales assuming stock is available. If stock is not on hand, demand is lost.

Dem	Prob	Dem	Prob	Dem	Prob	Dem	Prob	Dem	Prob
0	.020	4	.033	8	.046	12	.100	16	.041
1	.023	5	.036	9	.060	13	.060	17	.038
2	.027	6	.039	10	.100	14	.045	18	.036
3	.030	7	.043	11	.130	15	.042	over 18	.051

- (a) Taking states $i = 0, \dots, 18$ for inventory on hand as each week begins, identify all the elements of definition 9.44 to formulate this inventory management challenge as a MDP maximizing the total expected undiscounted net profit of gross income from sales less ordering and holding costs. Assume initial inventory in week 1 equals 18, and that the order the end of week 5 will restore that max inventory level.
- (b) Sketch a digraph depicting your model of part (a) over nodes for 5 stages and 18 states with arcs for possible transitions. You need not insert all parameter details, but do show exemplars of all transition arcs with the decision to which they are attached, the reward they would realize, and the associated probability.
- (c) Form the functional equations for all states and stages for your model of (a).
- (d) Solve the equations of (c) to determine an optimal inventory policy for SK.

9-42 Mini Job (MJ) is a small job shop manufacturer with a contract to stamp 200 copies per day for the next 5 days of a metal door panel needed by an automobile manufacturer. MJ's machining center that does the stamping can meet that demand with regular 40-hour shifts, costing a total of \$3000 in labor per day if the machine is full "working" state $i = 1$ in which it begins. But if the device has "minor defects" (state $i = 2$), productivity falls to 160 copies within regular hours, leaving the remaining 40 to be done on overtime at a total of \$4500 per day. If the the device has "major defects" (state $i = 3$), only 75 units can be produced in regular hours, 25 more on the overtime shift, and the other 100 must be purchased from a competitor at a total of \$10,000 per day. Finally, the machine may be completely "inoperable" (state $i = 4$). On such days NJ must contract the full 200 units at a total cost of \$20,000.

MJ's planning challenge is to decide when to do repair on the work center. On any day, MJ can leave the machine in its current condition, or undertake repairs. In full working state $i = 1$, the machine

will continue in that state with probability 0.60 and decline to state $i = 2$ with probability 0.40. Without repair of minor defects, the machine may continue in state $i = 2$ for another day at probability 0.50, or decline to state $i = 3$ or $i = 4$ with probabilities 0.30 and 0.20, respectively. Similarly, without repair of major defects the machine will continue state $i = 3$ with probability 0.40 and become inoperable with probability 0.60. Assume repair happens instantly once a decision is made.

If repairs are made to the machine in minor defects state $i = 2$ at cost \$9000, there is a 0.90 chance it will return to full working state, and 0.10 that it continues to have minor defects. Similarly, if repairs are made to the machine in major defects state $i = 3$ at cost \$14,000, there is a 0.70 chance of restoring full working state, 0.20 of moving up to minor defects state $i = 2$, and 0.10 of continuing to have major defects. Doing repairs on the center when it is completely inoperable, would cost \$17,000, but have a 0.60 chance of making it fully operable, a 0.20 chance of leaving it with minor defects, a 0.15 chance of moving it only to major defects state, and 0.05 of leaving the machine still inoperable.

MJ seeks a policy for when to repair that minimizes the total expected cost of its operations over the 5-day contract period.

- (a) Formulate MJ's challenge as a MDP with multiple states and stages, including identifying all the elements of definition 9.44.
- (b) Sketch a digraph depicting your model of part (a) over nodes for the states and stages, plus arcs for possible transitions. You need not insert all parameter details, but do show exemplars of all transition arcs with the decision to which they are attached, the reward they would realize, and the associated probability.
- (c) Form the functional equations for all states and stages of your model in (a).
- (d) Solve the equations of (c) to determine an optimal repair policy for MJ.

9-43 Elite Air (EA)² is a business-class only airline advertising complete meals for all its passengers. EA must choose, then update the number to

²Based on J. H. Goto, M. E. Lewis, and M. L. Puterman (2002), "Coffee, Tea, or ...?: A Markov Decision Process Model for Airline Meal Provisioning," ORIE, Cornell University.

order q_e , at epochs $e = 4, \dots, 0$, which is initially set to the number of booked and standby passengers b_4 known at that time. Values of both q_e and b_e can range between $0, 1, \dots, B \triangleq$ the seating capacity of the flight. As flight time approaches, meal requirements and passenger loads are re-evaluated at every 4-hour epoch.

The predicted number of passengers who must be served varies stochastically over the epochs as bookings are clarified, but values are independent of meal planning decisions, specifically $p[b_e, b_{e-1}] \triangleq$ the probability that estimate b_e is updated to b_{e-1} during epoch $e = 4, \dots, 1$.

Order quantities q_e are also reviewed at each e . In epochs $e = 4$ and 3 , meals can be ordered at standard cost c dollars each. Thereafter, extra meals may be added at late-order cost $\$1.8c$. Meals already on order may be cancelled, but the savings will only be $\$0.4c$, not the full original amount.

EA wants to develop a meal ordering policy that assures $q_0 = b_0$ at departure time, while

minimizing total expected cost of meal purchase and return.

- (a) Formulate EA's challenge as a MDP with multiple states and stages, including identifying all the elements of definition [9.44](#). Use bivariate states (q_e, b_e) with $q_e \triangleq$ the current number of meals ordered, and $b_e \triangleq$ the corresponding estimate of passengers expected. Round any fractional value arising in computations to the nearest integer.
- (b) Sketch a digraph depicting your model of part (a) over nodes for the states and stages, plus arcs for possible transitions. You need not insert all parameter details, but do show exemplars of all transition arcs with the decision to which they are attached, the reward they would realize, and the associated probability.
- (c) Form the functional equations over all states and stages for your model of (a).

REFERENCES

- Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin (1993), *Network Flows*, Prentice-Hall, Upper Saddle River, New Jersey.
- Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.
- Bertsekas, Dimitri P. (1987), *Dyanmic Programming: Deterministic and Stochastic Models*, Prentice-Hall, Englewood Cliffs, New Jersey.
- Denardo, Eric V. (2003), *Dynamic Programming: Models and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey.
- Lawler, Eugene (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehardt and Winston, New York, New York.
- Puterman, Martin L. (2005), *Markov Decision Processes-Discrete Stochastic Dynamic Programming*, Wiley, Hoboken, New Jersey.

Network Flows and Graphs

We have seen in Chapters 4 to 7 that linear program models admit some very elegant analysis. Global optima can be computed efficiently, and all sorts of “what if” sensitivity analyses can be performed on results.

Network flow problems are special, yet widely applicable cases of linear programs which prove even more tractable. Much larger models can be solved, because specialized algorithms apply. Most important, discrete cases, which we know are usually more difficult, can often be managed with no extra effort at all.

10.1 GRAPHS, NETWORKS, AND FLOWS

One of the things that make network flow models particularly tractable is that decisions and constraints have a form that we can easily represent in a diagram. More precisely, network flow models arise on structures called **directed graphs** or **digraphs**.

Digraphs, Nodes, and Arcs

We encountered digraphs in Section 9.1. The digraphs begin with a collection of **nodes** (or **vertices**), which we denote throughout this chapter by

$$V \triangleq \{\text{nodes or vertices of the network}\}$$

These indicate the facilities, or intersections, or transfer points of the network. Nodes are joined in digraphs by a collection of **arcs**, which we denote throughout this chapter by

$$A \triangleq \{\text{arcs of the network}\}$$

Arcs show possible flows or movements from one node to another. We indicate individual arcs simply by listing the pair of nodes they connect. For example, arc (4, 7) would go from node 4 to node 7.

Digraphs are termed **directed** because the direction of flow matters. For example, an arc leading from node 7 to node 4, which would be denoted (7, 4), is not the same as (4, 7). They represent traffic in different directions.

APPLICATION 10.1: OPTIMAL OVENS (OOI)

As usual, it will be much easier to absorb key notions with a small model in mind. Consider the (entirely fictitious) case of Optimal Ovens, Incorporated (OOI).

OOI makes home toaster ovens at plants in Wisconsin and Alabama. Completed ovens are shipped by rail to one of OOI’s two warehouses in Memphis and Pittsburgh, and then distributed to customer facilities in Fresno, Peoria, and Newark. The two warehouses can also transfer small quantities of ovens between themselves, using company trucks.

Our task is to plan OOI’s distribution of new model E27 ovens over the next month. Each plant can ship up to 1000 units during this period, and none are presently stored at warehouses. Fresno, Peoria, and Newark customers require 450, 500, and 610 ovens, respectively. Transfers between the warehouses are limited to 25 ovens, but no cost is charged. Unit costs (in dollars) of other possible flows are detailed in the following tables.

From/To	3: Memphis	4: Pittsburgh
1: Wisconsin	7	8
2: Alabama	4	7

From/To	5: Fresno	6: Peoria	7: Newark
3: Memphis	25	5	17
4: Pittsburgh	29	8	5

OOI Application Network

Figure 10.1 depicts OOI’s network. The 2 plants, 2 warehouses, and 3 customer sites make up the 7 nodes of this digraph. Arcs show the possible oven flows, with arrowheads indicating direction. For example, arc (3, 7) denotes ovens shipped from the Memphis warehouse to the Newark customer site. The absence of an arc (7, 3) means that ovens are not allowed to make the opposite move from Newark

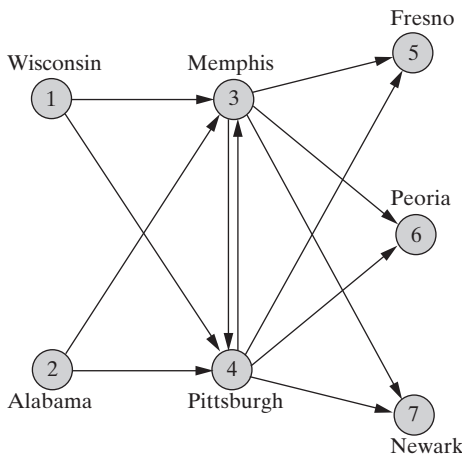


FIGURE 10.1 Network for Optimal Ovens, Incorporated (OOI) Application

to Memphis. Two opposed arcs joining the warehouse nodes indicate that traffic between them can flow in either direction.

Minimum Cost Flow Models

What linear program does a digraph like Figure 10.1 represent? We want it to describe flows, here of ovens starting at plants, passing through warehouses and terminating at customers.

As usual, we begin with decision variables.

Principle 10.1 Decision variables $x_{i,j}$ in network flow models reflect the amount of flow in arcs (i, j) .

Letting $c_{i,j}$ denote the unit cost of flow on arc (i, j) , the total cost to be minimized is simply

$$\sum_{(i,j) \in A} c_{i,j} x_{i,j}$$

Some constraints are equally easy. Flows must be nonnegative to make sense, and **capacities** or upper bounds $u_{i,j}$ may apply. These requirements lead to constraints:

$$0 \leq x_{i,j} \leq u_{i,j} \quad \text{for all } (i, j) \in A \quad (10.1)$$

The defining characteristic of network flow problems is the form of their main constraints.

Principle 10.2 Main constraints of network flow problems enforce **balance** (or **conservation**) of flow at nodes.

More precisely, we want

$$(\text{total flow in}) - (\text{total flow out}) = \text{specified net demand}$$

at every node. In symbols,

$$\sum_{(i,k) \in A} x_{i,k} - \sum_{(k,j) \in A} x_{k,j} = b_k \quad \text{for all } k \in V \quad (10.2)$$

where b_k denotes the specified net demand (required flow imbalance) at node k .

We are now ready to state the full minimum cost network flow model form.

Definition 10.3 The **minimum cost network flow model** for a digraph on nodes $k \in V$ with net demands b_k , and arcs $(i, j) \in A$ with capacity $u_{i,j}$ and unit cost $c_{i,j}$ is

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_{(i,k) \in A} x_{i,k} - \sum_{(k,j) \in A} x_{k,j} = b_k \quad \text{for all } k \in V \\ & 0 \leq x_{i,j} \leq u_{i,j} \quad \text{for all } (i, j) \in A \end{aligned}$$

Sources, Sinks, and Transshipment Nodes

Nodes come in three types. **Sink** or **demand** nodes such as the customer sites in the OOI application consume flow. **Source** or **supply** nodes such as the OOI plants create flow. **Transshipment** nodes such as OOI warehouses merely pass along flow. Net demands b_k have corresponding sign.

Principle 10.4 Net demand b_k is positive at sink (demand) nodes, negative at source (supply) nodes, and zero at transshipment nodes.

OOI Application Model

Figure 10.2 includes net demands, costs, and capacities on the digraph for our OOI application (plus an extra node 8 explained below). The corresponding minimum cost network flow model is

$$\begin{aligned}
 \min \quad & 7x_{1,3} + 8x_{1,4} + 4x_{2,3} + 7x_{2,4} + 25x_{3,5} + 5x_{3,6} \\
 & + 17x_{3,7} + 29x_{4,5} + 8x_{4,6} + 5x_{4,7} && \text{(total cost)} \\
 \text{s.t.} \quad & -x_{1,3} - x_{1,4} - x_{1,8} && = -1000 \quad \text{(node 1)} \\
 & -x_{2,3} - x_{2,4} - x_{2,8} && = -1000 \quad \text{(node 2)} \\
 & +x_{1,3} + x_{2,3} + x_{4,3} - x_{3,4} - x_{3,5} - x_{3,6} - x_{3,7} && = 0 \quad \text{(node 3)} \\
 & +x_{1,4} + x_{2,4} + x_{3,4} - x_{4,3} - x_{4,5} - x_{4,6} - x_{4,7} && = 0 \quad \text{(node 4)} \quad (10.3) \\
 & +x_{3,5} + x_{4,5} && = 450 \quad \text{(node 5)} \\
 & +x_{3,6} + x_{4,6} && = 500 \quad \text{(node 6)} \\
 & +x_{3,7} + x_{4,7} && = 610 \quad \text{(node 7)} \\
 & +x_{1,8} + x_{2,8} && = 440 \quad \text{(node 8)} \\
 & x_{3,4} \leq 25, x_{4,3} \leq 25 && \text{(capacities)} \\
 & x_{i,j} \geq 0 \quad \text{for all } (i,j) \in A
 \end{aligned}$$

Heavy lines in Figure 10.3 show an optimal solution.

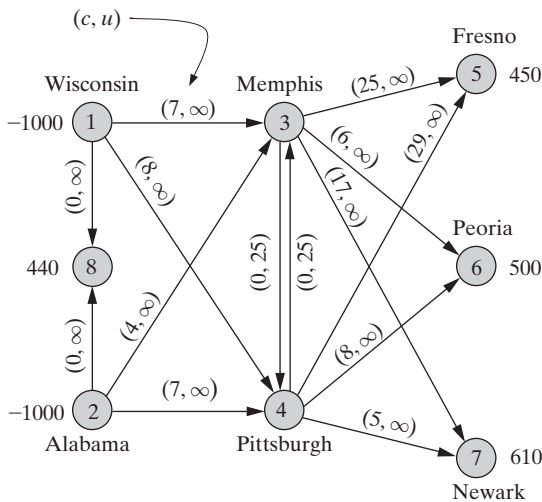


FIGURE 10.2 Minimum Cost Network Flow Problem for OOI Application

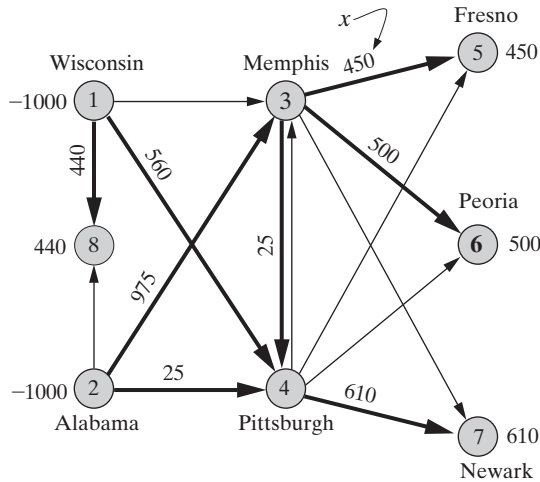


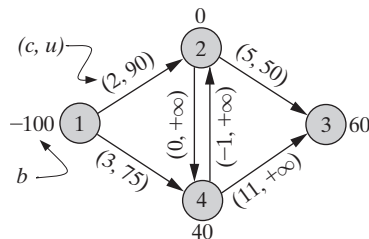
FIGURE 10.3 Optimal Flows for OOI Application

Plants are the source nodes of the OOI network. Thus their flow balance constraints (the first two) in model (10.3) have negative net demands. Warehouse nodes merely transship flow, so net demand is zero. The last four balance constraints of model (10.3) detail the sink nodes consuming flow; corresponding right-hand sides are positive.

The OOI application has capacities only on arcs connecting the two warehouses. No more than 25 ovens may be transferred in either direction. Bound constraints in model (10.3) reflect these two capacity limits and nonnegativity on all arcs.

EXAMPLE 10.1: FORMULATING MINIMUM COST NETWORK FLOW MODELS

The figure that follows shows a network on four nodes. Numbers next to nodes are net demands b_k , and those on arcs are cost and capacity $(c_{i,j}, u_{i,j})$.



- (a) Formulate the corresponding minimum cost network flow problem.
- (b) Classify nodes of the problem as source, sink, or transshipment.

Solution: Here

$$V = \{1, 2, 3, 4\}$$

$$A = \{(1, 2), (1, 4), (2, 3), (2, 4), (4, 2), (4, 3)\}$$

(a) Using variables $x_{1,2}$, $x_{1,4}$, $x_{2,3}$, $x_{2,4}$, $x_{4,2}$, and $x_{4,3}$ to represent flows on the six members of A , the formulation of principle [10.3](#) is

$$\begin{aligned}
 \min \quad & 2x_{1,2} + 3x_{1,4} + 5x_{2,3} - 1x_{4,2} + 11x_{4,3} \\
 \text{s.t.} \quad & -x_{1,2} - x_{1,4} = -100 \\
 & x_{1,2} + x_{4,2} - x_{2,3} - x_{2,4} = 0 \\
 & x_{2,3} + x_{4,3} = 60 \\
 & x_{1,4} + x_{2,4} - x_{4,2} - x_{4,3} = 40 \\
 & x_{1,2} \leq 90, x_{1,4} \leq 75, x_{2,3} \leq 50 \\
 & x_{i,j} \geq 0 \quad \text{for all } (i,j) \in A
 \end{aligned}$$

(b) The only node with a negative net demand or supply is node 1. Thus it is the only source node. Nodes 3 and 4 have positive net demand, making them sinks. Remaining node 2, which has neither a demand nor a supply, is a transshipment node.

Total Supply = Total Demand

One element appears in Figure 10.2 and model (10.3) that was absent in original Figure 10.1. An extra sink node 8 has been added.

To see why, look again at flow balance constraints of [10.3](#). Flow is created only at source nodes and consumed only at sink nodes. Thus there is no hope of a feasible flow unless

$$\text{total supply} = \text{total demand}$$

which means $\sum_k b_k = 0$.

When the given network of a minimum cost network flow problem does not come to us with total supply equal total demand, we must make some adjustments before proceeding.

Principle 10.5 If total supply is less than total demand in a given network flow problem, the problem is infeasible. If total supply exceeds total demand, a new sink node should be added to consume excess demand via zero-cost arcs from all sources.

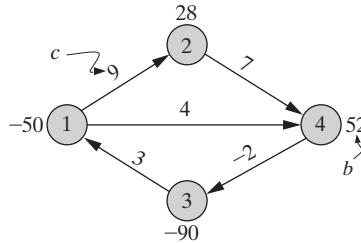
Node 8 in Figure 10.2 was added because

$$\text{total supply} = 1000 + 1000 > 450 + 500 + 610 = \text{total demand}$$

in the OOI application. The excess of 440 determines node 8’s demand. We want this excess supply to be able to reach node 8 without affecting any other part of the optimization. Zero-cost arcs from both source nodes do the job.

EXAMPLE 10.2: BALANCING TOTAL SUPPLY AND TOTAL DEMAND

The digraph below shows net demand b_k next to each of its nodes and cost $c_{i,j}$ on its arcs.

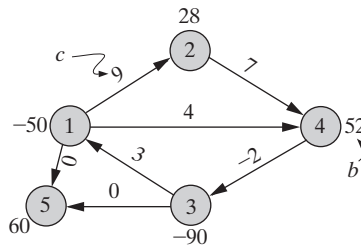


Modify the network as required to produce an equivalent one with total supply equal to total demand.

Solution: Here total supply is $50 + 90 = 140$ and total demand is $28 + 52 = 80$. Thus total supply exceeds total demand by

$$140 - 80 = 60$$

To produce an equivalent model with total supply equal to total demand, we apply principle [10.5](#) and add a “dummy” sink 5 to consume the excess.



Zero-cost arcs $(1, 5)$ and $(3, 5)$ from the two sources assure that the extra supply can reach node 5 without affecting other costs.

Starting Feasible Solutions

All search methods for network flows proceed by assuming a starting feasible solution is at hand, then systematically improving it until an optimum is obtained or unboundedness is established. When a starting solution is not known, or there is uncertainty about whether one exists, the standard network flow model can be adapted to investigate using two-phase or big- M methods of Section 3.5.

The key to both those methods is construction of an artificial model from which computation can begin. New artificial variables are introduced to force feasibility, and their sum is minimized. If that sum can be driven to $= 0$, what remains is a feasible flow for the original model. If not, the original model is infeasible.

Artificial Network Flow Model

The only new element in the network flow context is that we wish to create an artificial model that is itself a minimum cost network flow problem so that usual algorithms apply. In particular, we want to be able to interpret artificial variables as flows in arcs.

To obtain such a network flow artificial model, we simply put a zero flow on all arcs of the original model and add one artificial node. Supply and demand requirements are fulfilled by artificial arcs joining this special node to all others having net demand $b_k \neq 0$.

Principle 10.6 An **artificial network model** and starting point for computing initial feasible solutions in minimum cost network flow problems can be constructed by (i) assigning 0 flow to all arcs of the original model, (ii) introducing an artificial node, (iii) creating artificial arcs from each supply node $k (b_k < 0)$ to the artificial node with flow equal to the specified supply $|b_k|$, and (iv) adding artificial arcs from the artificial node to each demand node $k (b_k > 0)$ with flow equal to the required demand $|b_k|$.

Figure 10.4 illustrates for our OOI model. Artificial node 0 has been added to anchor artificial arcs. Then artificial arcs (1, 0) and (2, 0) balance flow at supply nodes 1 and 2 by carrying exactly the supply specified at those nodes. Similarly, artificial arcs (0, 5), (0, 6), (0, 7), and (0, 8) satisfy demand requirements by bringing the needed flow to each demand node. All original arcs have flow = 0.

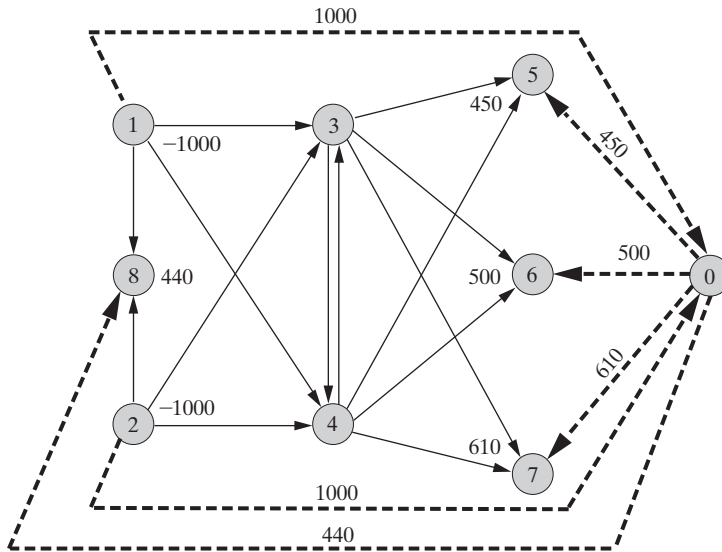
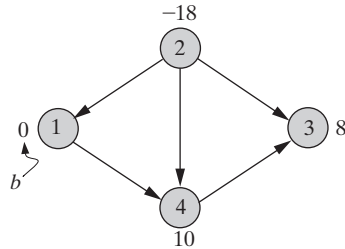


FIGURE 10.4 Starting Phase I Flow for the OOI Application

Zero flows on all original arcs satisfy their upper- and lower-bound constraints, and all specified artificial flows are nonnegative. Also, setting artificial flows to exactly the needed supply or demand has assured flow balance at all the original nodes. Finally, under total supply equals total demand requirement [10.5], flow will balance at the artificial node as well.

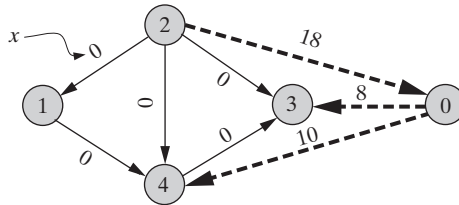
EXAMPLE 10.3: CONSTRUCTING AN ARTIFICIAL NETWORK MODEL

Consider a network flow problem with net demands as depicted in the following figure:



Construct both the corresponding artificial network model for phase I or big- M computation of a starting feasible solution and the associated artificial solution.

Solution: Following construction [10.6](#), we introduce an artificial node 0, along with artificial arcs from supply node 2 and to demand nodes 3 and 4. Flows on original arcs are all zero, and those on artificials equal the specified supply or demand. The result is the following artificial model and artificial starting solution:



Time-Expanded Flow Models and Networks

As with the more general linear programs of Section 4.5, many applications of network flows involve time-expanded formulations to account for flows over time. This is especially true for those involving inventory management.

In the specially structured network flow case, such problems lead to time-expanded networks.

Definition 10.7 | **Time-expanded networks** model each node of a flow system as a series of nodes, one for each time interval. Arcs then reflect either flows between points in a particular time, or flows across time in a particular location.

APPLICATION 10.2: AGRICO CHEMICAL TIME-EXPANDED NETWORK FLOW

To see the idea, consider the case of Agrico Chemical,¹ which is a large chemical fertilizer company. Figure 10.5 sketches our fictitious version of the network model used to plan Agrico’s production, distribution, and inventory. Much in the spirit of OOI Application 10.1, products for this real company originate at 4 plants and are transshipped through 20 regional distribution centers before reaching customers in any of 500 service areas.

Highly seasonal demand makes the Agrico case more complex than OOI. Agrico produces fertilizers throughout the year, but much of the demand comes in the spring quarter. Production capacity cannot accommodate spring demand in just

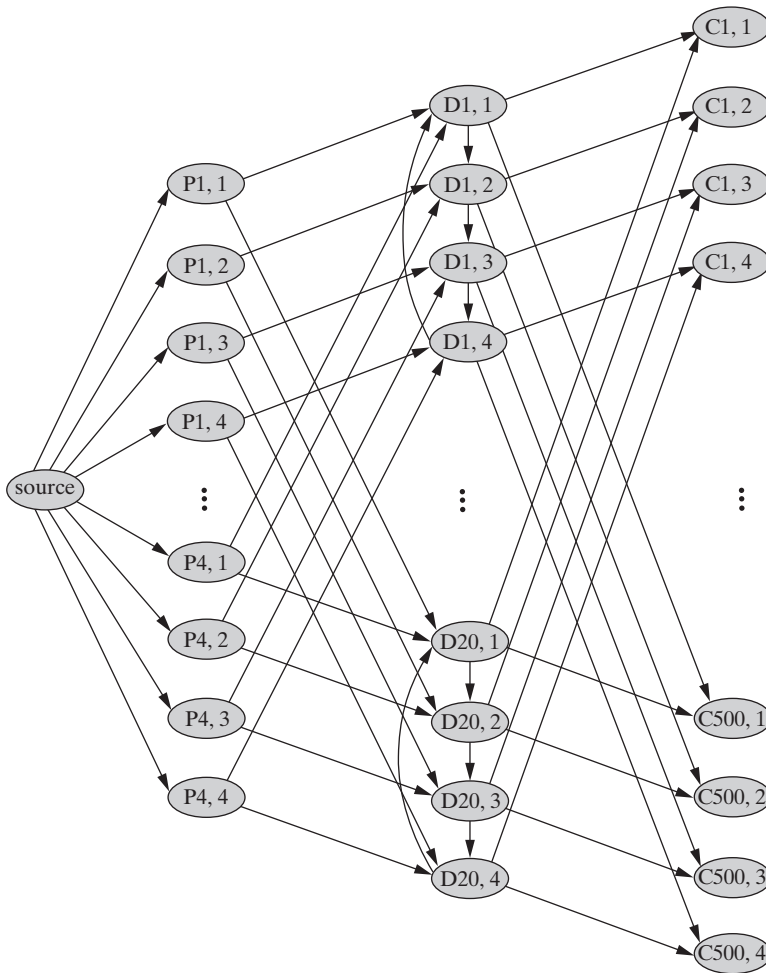


FIGURE 10.5 Agrico Time-Expanded Network Format

¹Based on F. Glover, G. Jones, D. Karney, D. Klingman, and J. Mote (1979), “An Integrated Production, Distribution, and Inventory Planning System,” *Interfaces*, 9:5, 21–35.

one quarter. Thus the company builds up inventories at distribution sites during off-seasons to ship in the spring. Of course, storage is not unlimited, and inventories result in holding costs.

Agrico's decision problem requires choosing the amount to produce in each of the four quarters of the year, the pattern of shipping and storing it at distribution sites, and a scheme for sending it on to customers. We want to do all this at minimum total cost.

Time-Expanded Modeling of Agrico Application

The Agrico digraph of Figure 10.5 illustrates a time-expanded network. Total annual flow originates at the source node, but separate arcs representing production connect it to the 8 nodes (P_i, t) modeling plants $P1$ through $P4$ in quarters $t = 1, \dots, 4$. Capacities on those arcs would enforce the quarterly capacities of the corresponding plants, and costs would reflect units costs of production.

Plants in any quarter, t , are connected by transportation arcs to distribution center nodes (D_j, t) in the same quarter. Distribution centers, in turn, are linked to customer demands of the corresponding quarter. Costs on these arcs would reflect the unit cost of transportation, perhaps differentiated by the quarter in which the shipment takes place.

Holding arcs between nodes for the same distribution center comprise the main new feature representable with time-expanded modeling. For example, the arc from $(D1, 2)$ to $(D1, 3)$ models holding of the product at distribution site number 1 from the second to the third quarter. Its cost would be the unit holding cost at distribution site 1, and its capacity the size of the available storage. It would be impossible to model both flows in time and flow among facilities without distinguishing nodes by both place and time.

EXAMPLE 10.4: MODELING IN TIME-EXPANDED NETWORKS

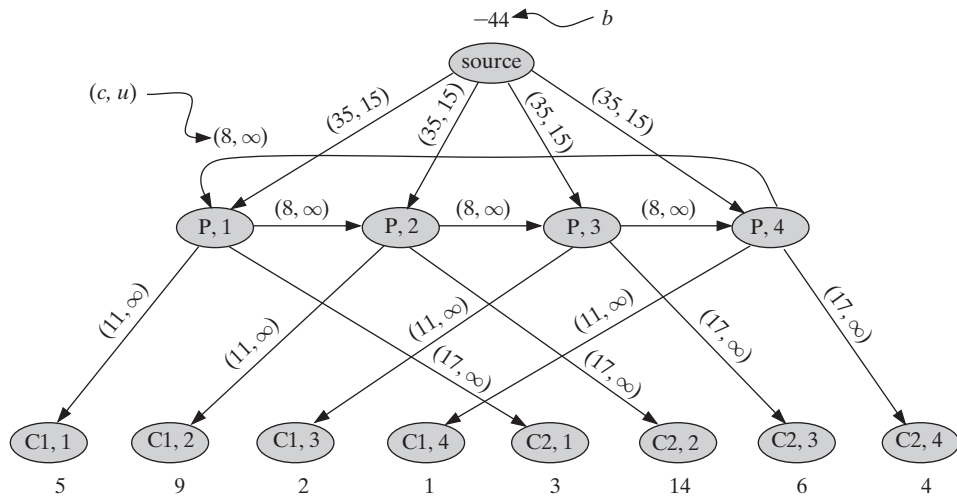
A certain company can manufacture up to 15 thousand units of its product in any calendar quarter at a cost of \$35 per thousand. The following table shows cost per thousand units shipped to each of the company's two customers, and the number (thousands) of units demanded by each in various quarters.

Customer	Shipping Cost	Demand by Quarter			
		1	2	3	4
1	11	5	9	2	1
2	17	3	14	6	4

Assuming that inventory can be maintained at the plant for \$8 per thousand per quarter, develop a time-expanded network flow model to determine the company's best production, distribution, and inventory plan.

Solution: Following [10.7], we create 4 nodes for the plant in different quarters and 4 nodes for each of the 2 customers in different quarters. Commodities for all quarters and customers arises at a common source node. Production arcs link commodities to plant nodes by quarter, and holding arcs connect the plant nodes. Transportation

arcs join the plant in each quarter to customer demands for that quarter. The result is the following time-expanded minimum cost network flow model:



Node–Arc Incidence Matrices and Matrix Standard Form

As in all our work with linear programs, we will often wish to think of minimum cost network flow model [10.3] in matrix terms. To do so, we will abuse notation to treat the flow variables $x_{i,j}$ as a vector \mathbf{x} even though components have two subscripts. Then, collecting costs $c_{i,j}$ and capacities $u_{i,j}$ in corresponding vectors \mathbf{c} and \mathbf{u} , and arraying net demands b_k in vector \mathbf{b} reduces the minimum cost network flow model [10.3] to the familiar LP standard form

$$\begin{aligned}
 \min \quad & \mathbf{c} \cdot \mathbf{x} \\
 \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\
 & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}
 \end{aligned} \tag{10.4}$$

Main constraint matrix \mathbf{A} has a very special structure. Such matrices are called node–arc incidence matrices because they both encode the flow balance requirements and provide an algebraic description of the underlying digraph.

Definition 10.8 Node–arc incidence matrices represent both the flow balance requirements and the graph structure of a network flow model with a row for every node and a column for every arc. The only nonzero entries in each column are a -1 in the row for the node the corresponding arc leaves and a $+1$ in the row for the node the arc enters.

Table 10.1 illustrates for the OOI application. The 8 rows correspond to the 8 nodes of the digraph in Figure 10.2. One column is present for each of the 14 arcs. The column for arc (3, 6) has a -1 in row 3 and a $+1$ in row 6 because arc (3, 6) leaves 3 and enters 6.

TABLE 10.1 Node–Arc Incidence Matrix of the OOI Application

Node	Arc													
	(1, 3)	(1, 4)	(1, 8)	(2, 3)	(2, 4)	(2, 8)	(3, 4)	(3, 5)	(3, 6)	(3, 7)	(4, 3)	(4, 5)	(4, 6)	(4, 7)
1	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0
3	+1	0	0	+1	0	0	-1	-1	-1	-1	+1	0	0	0
4	0	+1	0	0	+1	0	+1	0	0	0	-1	-1	-1	-1
5	0	0	0	0	0	0	0	+1	0	0	0	+1	0	0
6	0	0	0	0	0	0	0	0	+1	0	0	0	+1	0
7	0	0	0	0	0	0	0	0	0	+1	0	0	0	+1
8	0	0	+1	0	0	+1	0	0	0	0	0	0	0	0

One of the conveniences of network flow problems is that we can depict them in network diagrams. Notice, however, that we could just as well start from the node–arc incidence matrix. If we had been given Table 10.1, it would be easy to sketch the corresponding digraph.

EXAMPLE 10.5: CONSTRUCTING NODE–ARC INCIDENCE MATRICES

Construct the node–arc incidence matrix for the original digraph of Example 10.2.

Solution: Consistent with principle [10.8](#), the node–arc incidence matrix will have a row for each of the 4 nodes and a column for each of the 5 arcs. The full matrix is as follows:

Node	Arc				
	(1, 2)	(1, 4)	(2, 4)	(3, 1)	(4, 3)
1	-1	-1	0	+1	0
2	+1	0	-1	0	0
3	0	0	0	-1	+1
4	0	+1	+1	0	-1

EXAMPLE 10.6: INTERPRETING NODE–ARC INCIDENCE MATRICES

Consider the following matrix:

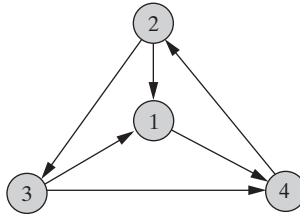
$$\begin{pmatrix} -1 & +1 & 0 & +1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & +1 \\ 0 & 0 & +1 & -1 & -1 & 0 \\ +1 & 0 & 0 & 0 & +1 & -1 \end{pmatrix}$$

- (a) Explain why it is a node–arc incidence matrix.
- (b) Draw the corresponding digraph.

Solution: Again we apply principle 10.8.

(a) This is a node–arc incidence matrix because each column has only two nonzero entries, one -1 and one $+1$.

(b) Associating nodes 1 to 4 with the 4 rows of the matrix, we construct the digraph by inserting an arc for each column of the matrix. The arc leaves the node of the row where the column has a -1 and enters the node of the row where it has a $+1$.



10.2 CYCLE DIRECTIONS FOR NETWORK FLOW SEARCH

Linear programming algorithms of Chapters 5–7 center on constructing improving feasible directions, directions of solution change that preserve feasibility and improve the objective function for suitably small steps. Knowing that network flow problems are linear programs with special properties, it should not surprise you that unusually simple improving feasible directions are at the heart of their tractability.

Chains, Paths, Cycles, and Dicycles

Sections 9.1 introduced the notions of **paths** and **dicycles** in a graph. To derive directions for network flow problems, we need the additional concepts of chains and cycles.

Definition 10.9 | A **chain** is a sequence of arcs connecting two nodes. Each arc has exactly one node in common with its predecessor in the sequence, and no node is visited more than once.

Definition 10.10 | A **cycle** is a chain with the same beginning and ending node.

We describe a chain or cycle merely by listing its arcs in sequence. When no confusion will result, we may also use the corresponding sequence of nodes.

Figure 10.6(a) illustrates on the OOI digraph of Figure 10.2. The first example shows a chain $(1, 3), (3, 6), (4, 6), (2, 4)$ connecting nodes 1 and 2. It could just as well be called by its node sequence $1-3-6-4-2$ because there is only one arc that could provide each of the implied node-to-node connections. Figure 10.6(a)'s second example is chain $(1, 3), (3, 4), (4, 7)$. Here the node sequence would not be definitive because both this chain and $(1, 3), (4, 3), (4, 7)$ have node sequence $1-3-4-7$.

Part (b) of Figure 10.6 shows some sequences that are not chains. The first is not connected, and the second repeats node 3.

Notice that chains need not observe direction on the arcs. This is how a chain differs from a path.

Definition 10.11 Paths are chains that transmit all arcs in the forward direction.

Thus highlighted sequence 1–3–6–4–2 in Figure 10.6(a) is a chain but not a path; it violates direction on arcs (4, 6) and (2, 4). Second sequence 1–3–4–7 is both a chain and a path.

The only additional element with cycles is beginning and ending at the same node. For example, Figure 10.6(c) includes cycle 1–3–6–4–1 starting and ending at node 1 and cycle (3, 4), (4, 3) beginning and ending at node 3. Part (d) of the figure confirms that disconnected arc sequences or ones repeating a node cannot constitute cycles.

As with chains and paths, the distinction between cycles and dicycles involves direction.

Definition 10.12 Dicycles are cycles that have all arcs oriented in the same direction.

Thus the first cycle 1–3–6–4–1 of Figure 10.6(c) is not a dicycle because it violates direction on arcs (4, 6) and (1, 4). Second cycle (3, 4), (4, 3) does meet the definition of a dicycle because it passes both arcs in the forward direction.

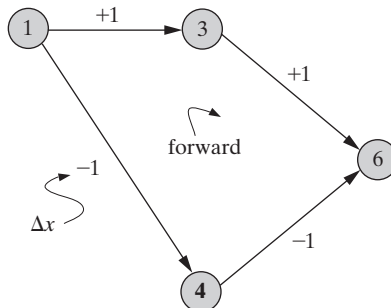
Cycle Directions

Cycles can pass arcs in either a **forward** (with direction) or the **reverse** (against direction) manner. Cycle directions derive from this visitation pattern.

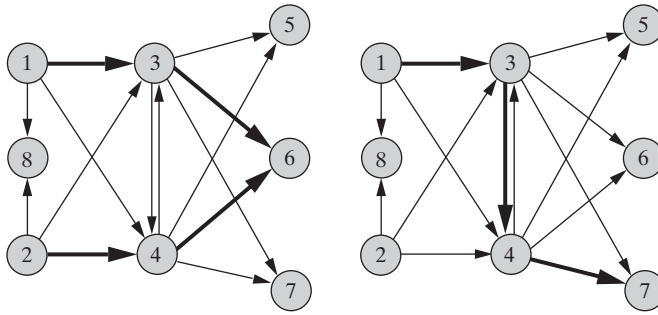
Definition 10.13 A **cycle direction** of a minimum cost network flow model increases flow on forward arcs and decreases flow on reverse arcs of a cycle in the given digraph; that is,

$$\Delta x_{i,j} \triangleq \begin{cases} +1 & \text{if arc } (i, j) \text{ is forward in the cycle} \\ -1 & \text{if arc } (i, j) \text{ is reverse in the cycle} \\ 0 & \text{if arc } (i, j) \text{ is not part of the cycle} \end{cases}$$

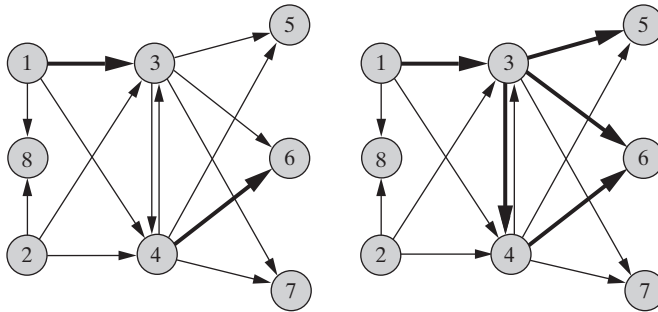
For example, consider the first cycle of Figure 10.6(c): 1–3–6–4–1. With the first two arcs forward in the cycle and the last two reverse, we obtain the cycle direction sketched below.



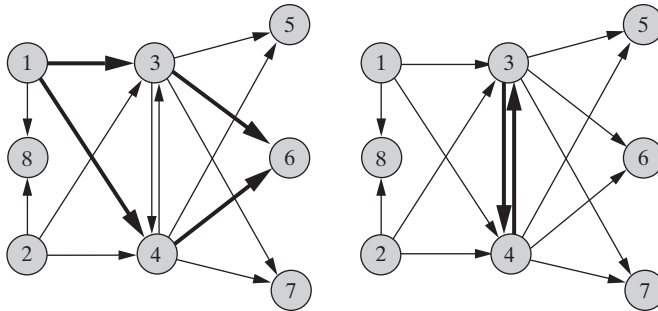
Forward arcs in the cycle have +1 components, reverse arcs have -1, and all other arcs have 0.



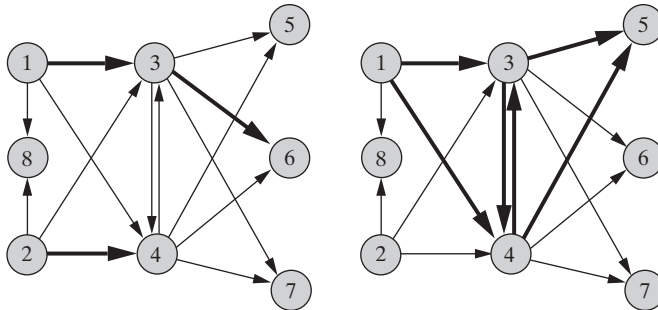
(a) Chains



(b) Not chains



(c) Cycles



(d) Not cycles

FIGURE 10.6 Chains and Cycles of the OOI Network

Notice that it matters which way we orient the cycle. If we used the same cycle as above but thought of visiting it in sequence 1-4-6-3-1, arcs (1,4) and (4,6) would be forward arcs with $\Delta x = +1$, and arcs (3,6) and (1,3) would be reverse with $\Delta x = -1$.

Maintaining Flow Balance with Cycle Directions

The main, flow conservation constraints of minimum cost network flow problems in matrix format (10.4) are equality constraints $\mathbf{Ax} = \mathbf{b}$ where \mathbf{A} is the node-arc incidence matrix of the given digraph and \mathbf{b} is the net demand vector. We know from our earliest investigation of improving search (principle [3.29], Section 3.3) that any direction $\Delta \mathbf{x}$ preserving feasibility in such equality constraints must satisfy the condition $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$. We are interested in cycle directions because they satisfy this net-change-zero condition for node-arc incidence matrices \mathbf{A} .

Principle 10.14 Adjusting a feasible flow along a cycle direction of a network flow model leaves flow balance constraints satisfied.

To see that this is true, recall that each node of the cycle is touched by exactly two arcs, and think about the four ways two arcs can visit a node. Figure 10.7 illustrates how node-arc incidence matrix signs of the direction [10.8] combine with those of the direction [10.13] to produce a net change = 0 at each node. In the forward-reverse case, for example, we have

$$+1 \Delta x_{i,j} + 1 \Delta x_{k,j} = +1(+1) + 1(-1) = 0$$

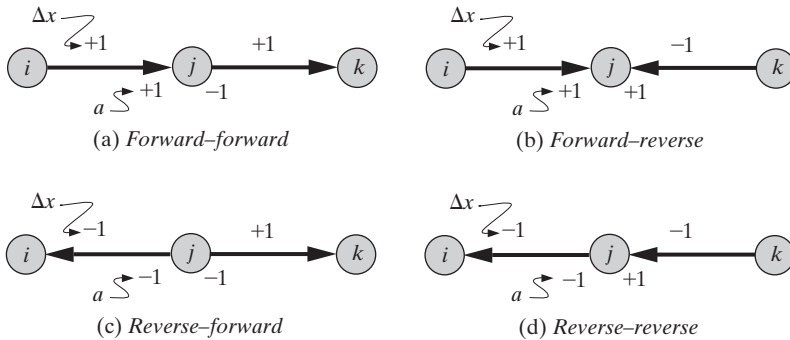
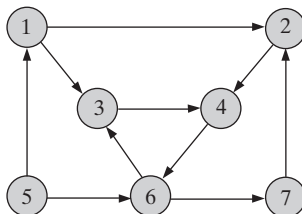


FIGURE 10.7 Possible Ways a Cycle Can Visit a Node

EXAMPLE 10.7: CONSTRUCTING CYCLE DIRECTIONS

Consider the following digraph:



Construct the cycle direction for each of the following cycles, and verify that each retains flow balance at node 6.

(a) 1–2–7–6–5–1

(b) 3–4–6–3

Solution:

(a) Applying definition [10.13](#), the cycle direction has

$$\Delta x_{1,2} = \Delta x_{5,1} = +1$$

$$\Delta x_{7,2} = \Delta x_{6,7} = \Delta x_{5,6} = -1$$

$$\Delta x_{1,3} = \Delta x_{3,4} = \Delta x_{4,2} = \Delta x_{6,3} = \Delta x_{4,6} = 0$$

Checking principle [10.14](#) at node 6,

$$-1 \Delta x_{6,7} + 1 \Delta x_{5,6} = -1(-1) + 1(-1) = 0$$

(b) Again applying definition [10.13](#), the cycle direction has

$$\Delta x_{3,4} = \Delta x_{4,6} = \Delta x_{6,3} = 1$$

$$\Delta x_{1,2} = \Delta x_{1,3} = \Delta x_{2,4} = \Delta x_{7,2} = \Delta x_{6,7} = \Delta x_{5,6} = \Delta x_{5,1} = 0$$

At node 6

$$+1 \Delta x_{4,6} - 1 \Delta x_{6,3} = +1(+1) - 1(+1) = 0$$

Feasible Cycle Directions

Flow balance equalities are not the only constraints of a minimum cost network flow model [10.3](#). We must also be concerned with nonnegativity constraints and arc capacities $u_{i,j}$. Section 3.3 principles [3.27](#) and [3.28](#) tell us the requirements that must be added for upper and lower bounds. No arc with zero flow can decrease, and no capacitated arc can increase if bound constraints are also to be maintained in a move from a current feasible solution \mathbf{x} .

Principle 10.15 A cycle direction $\Delta \mathbf{x}$ is a feasible direction at current solution \mathbf{x} if and only if $x_{i,j} > 0$ on all reverse arcs of the cycle, and $x_{i,j} < u_{i,j}$ on all forward arcs.

To illustrate, we need a feasible flow. Consider the OOI flow $\mathbf{x}^{(0)}$ depicted in Figure 10.8. That $\mathbf{x}^{(0)}$ satisfies net demand requirements at all nodes and conforms to all bounds.

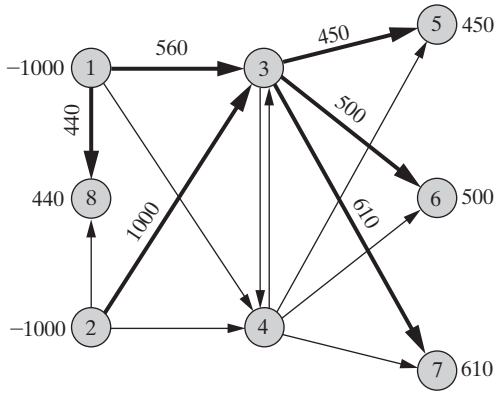


FIGURE 10.8 Initial Flow $\mathbf{x}^{(0)}$ in OOI Application

Figure 10.9 shows a cycle 2-4-7-3-2 that does satisfy conditions [10.15](#). Both reverse arcs have $x_{i,j}^{(0)} > 0$, and neither forward arc has a capacity.

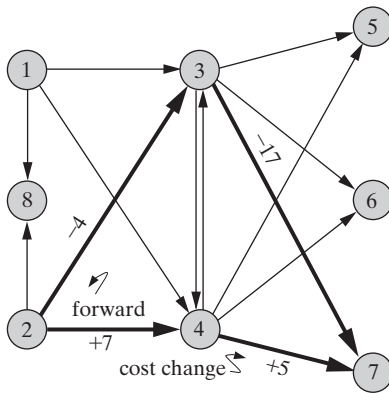
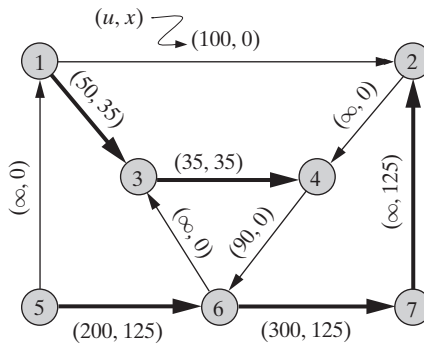


FIGURE 10.9 An Improving Feasible Cycle Direction at $\mathbf{x}^{(0)}$ of Figure 10.8

EXAMPLE 10.8: IDENTIFYING FEASIBLE CYCLE DIRECTIONS

The figure that follows shows a network flow problem with a feasible flow \mathbf{x} on highlighted arcs. Labels on arcs indicate capacities and flows $(u_{i,j}, x_{i,j})$.



Determine whether the direction for each of the following cycles is a feasible cycle direction.

- (a) 1–2–7–6–5–1
- (b) 3–4–6–3
- (c) 1–3–6–5–1

Solution: We apply conditions [10.15](#).

- (a) This cycle direction is feasible because all reverse arcs currently have positive flow and no forward arc is at capacity.
- (b) This cycle direction is not feasible. Forward arc (3, 4) is at capacity and cannot increase.
- (c) This cycle direction is not feasible. One of the reverse arcs, (3, 6), has current flow 0 and cannot decrease.

Improving Cycle Directions

Feasible cycle directions aid a search for an optimal flow only if they improve the objective function. We know from principle [3.18](#) (Section 3.3) that this will be true if $\bar{c} \triangleq \mathbf{c} \cdot \Delta \mathbf{x} < 0$ for our minimizing model. The simple +1, -1, 0 coefficient structure of cycle directions (principle [10.13](#)) make this test particularly easy to apply.

$$\begin{aligned}
 \bar{c} &\triangleq \mathbf{c} \cdot \Delta \mathbf{x} \\
 &= \sum_{(i,j) \text{ in cycle}} c_{i,j} \Delta x_{i,j} \\
 &= \sum_{\text{forward } (i,j)} c_{i,j}(+1) + \sum_{\text{reverse } (i,j)} c_{i,j}(-1) \\
 &= (\text{total forward arc cost}) - (\text{total reverse arc cost})
 \end{aligned}$$

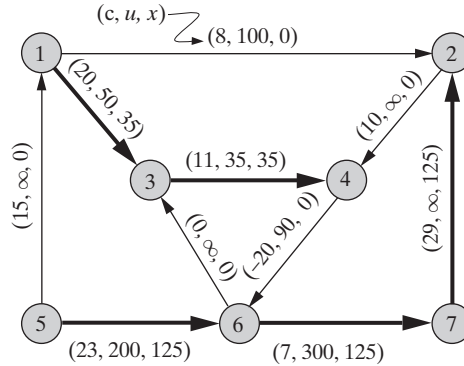
Principle 10.16 | A cycle direction improves for a minimum cost network flow model if the difference of total forward arc cost and total reverse arc cost < 0 .

Cycle 2–4–7–3–2 of Figure 10.9 illustrates a direction that is both feasible and improving. Applying [10.16](#) to check the latter gives

$$(\text{total forward}) - (\text{total reverse}) = (7 + 5) - (17 + 4) = -9 < 0$$

EXAMPLE 10.9: IDENTIFYING IMPROVING CYCLE DIRECTIONS

The figure that follows adds costs to the network of Example 10.8. Labels on arcs now show costs, capacities, and current flows $(c_{i,j}, u_{i,j}, x_{i,j})$.



Determine whether each of the following cycles yields an improving cycle direction.

- (a) 1–2–7–6–5–1
- (b) 2–4–6–7–2
- (c) 6–3–1–2–4–6

Solution: We apply condition [10.16](#).

(a) This cycle direction improves because

$$(\text{total forward}) - (\text{total reverse}) = (8 + 15) - (29 + 7 + 23) = -36$$

(b) This cycle direction does not improve because

$$(\text{total forward}) - (\text{total reverse}) = (10 - 20 + 7 + 29) - (0) = 26$$

(c) This cycle direction improves because

$$(\text{total forward}) - (\text{total reverse}) = (0 + 8 + 10 - 20) - (20) = -22$$

Step Size with Cycle Directions

If we can take an arbitrary step λ in improving feasible direction $\Delta \mathbf{x}$, we know that our model is unbounded. The objective can be improved forever without losing feasibility.

More often, upper and lower bounds on arc flows impose limits. Each unit step in a cycle direction increases forward arc flows by a unit and decreases reverse flows by a unit. Feasibility will be lost the first time these changes encounter a nonnegativity or capacity constraint. More specifically,

Principle 10.17 Steps from feasible flow \mathbf{x} in cycle direction $\Delta \mathbf{x}$ retain feasibility for step sizes up to $\lambda = \min \{ \lambda^+, \lambda^- \}$, where

$$\lambda^+ \triangleq \min \{ (u_{i,j} - x_{i,j}) : (i,j) \text{ forward} \} (= +\infty \text{ if there are no forward arcs})$$

$$\lambda^- \triangleq \min \{ x_{i,j} : (i,j) \text{ reverse} \} (= +\infty \text{ if there are no reverse arcs})$$

Return to the cycle direction 2–4–7–3–7 of Figure 10.9 and the flow $\mathbf{x}^{(0)}$ of Figure 10.8. For that example,

$$\begin{aligned}\lambda^+ &= \min\{(\infty - 0), (\infty - 0)\} = \infty \\ \lambda^- &= \min\{610, 1000\} = 610\end{aligned}$$

Thus $\lambda = \min\{\infty, 610\} = 610$ is the largest step we can take in that cycle direction without losing feasibility.

EXAMPLE 10.10: COMPUTING STEPS IN CYCLE DIRECTIONS

Return to the example of Example 10.9. Whether or not directions corresponding to the three specified cycles improve the objective function, determine the maximum step λ in each direction that preserves feasibility.

Solution: We apply principle [10.17](#).

(a) For cycle 1–2–7–6–5–1,

$$\begin{aligned}\lambda^+ &= \min\{(100 - 0), (\infty - 0)\} = 100 \\ \lambda^- &= \min\{125, 125, 125\} = 125 \\ \lambda &= \min\{\lambda^+, \lambda^-\} = \min\{100, 125\} = 100\end{aligned}$$

(b) For cycle 2–4–6–7–2,

$$\begin{aligned}\lambda^+ &= \min\{(\infty - 0), (90 - 0), (300 - 125), (\infty - 125)\} = 90 \\ \lambda^- &= +\infty \\ \lambda &= \min\{\lambda^+, \lambda^-\} = \min\{90, \infty\} = 90\end{aligned}$$

(c) For cycle 6–3–1–2–4–6,

$$\begin{aligned}\lambda^+ &= \min\{(\infty - 0), (100 - 0), (\infty - 0), (90 - 0)\} = 90 \\ \lambda^- &= \min\{35\} = 35 \\ \lambda &= \min\{\lambda^+, \lambda^-\} = \min\{90, 35\} = 35\end{aligned}$$

Sufficiency of Cycle Directions

Properties we long ago encountered for linear programs of any form, including network flow models of present interest, make a solution globally optimal if and only if it admits no improving feasible directions (principle [5.1](#)). We have seen how cycle directions can be improving and feasible. But what if none exist that satisfy both requirements? Certainly there are more complicated improving feasible directions for network flow models. Could any improve a solution when cycle directions fail? Happily, no.

Principle 10.18 | A feasible flow in a minimum cost network flow problem is (globally) optimal if and only if it admits no improving feasible cycle direction.

To formally justify property [10.18](#), we will demonstrate how every more complex improving feasible direction can be decomposed.

Principle 10.19 Every direction $\Delta \mathbf{x}$ satisfying $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$ for node–arc incidence matrix \mathbf{A} of a minimum cost network flow model can be decomposed into a weighted sum of cycle directions. Furthermore, if $\Delta \mathbf{x}$ is feasible and improving, at least one of the cycle directions will also be feasible and improving.

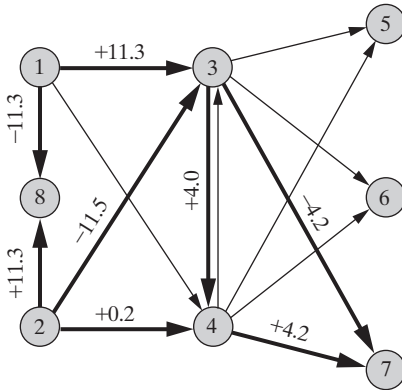
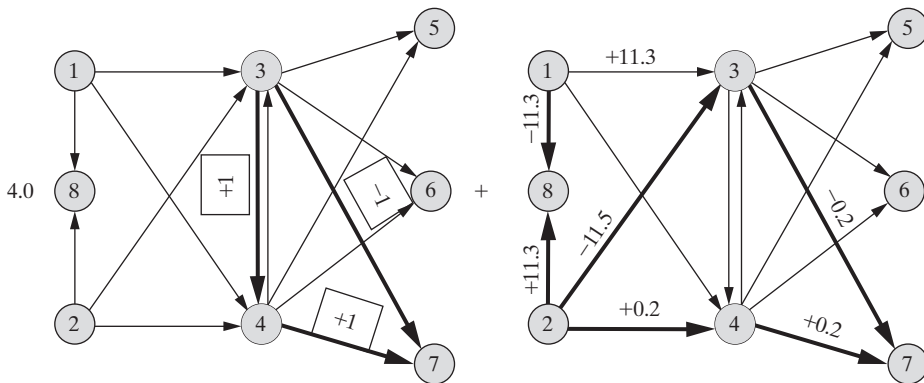


FIGURE 10.10 A Non-Cycle Improving Feasible Direction at $\mathbf{x}^{(0)}$ of Figure 10.8

Consider the direction shown in Figure 10.10. It certainly does not look like a cycle direction, but the reader can check that it satisfies conditions to be both feasible and improving. The objective changes by $\bar{c} = -\$15.9$ for every unit step.

To produce the decomposition of [10.19](#), pick any cycle within the nonzero arcs of the complex direction, and let $\alpha =$ the minimum absolute value of its components. One of several cycles in Figure 10.10 is 3–4–7–3, with $\alpha = 4.0$. Now express the complex direction as the sum of α times the (necessarily feasible) cycle direction for the chosen cycle, plus what remains of the complex one after forward components along the cycle are reduced by α , and reverse increased by the same amount. For the example of Figure 10.10 this gives



Notice several things. First, the remaining direction must be feasible because if its sum at each node were not $= 0$, the weighted sum, which is the original

direction, could not have been feasible. Furthermore, our choice of α assures no signs on arcs in the original direction have changed in the residual one. Importantly, however, the coefficient on at least one arc has dropped to $= 0$. Thus we can repeat the process, finding a cycle in each residual with weight $\alpha_1 \dots$, until no nonzero components remain. The resulting decomposition is a positive-weighted sum of the cycle directions, so its reduced cost must be the same sum of the reduced costs for the included cycle directions. If the sum has the right sign to improve, at least one of the cycle directions must have that sign as well.

Rudimentary Cycle Direction Search for Network Flows

Most known procedures for minimum cost network flow problems can be viewed as special forms of improving search using cycle directions. With all the properties of cycle directions in hand, we are now ready to specify Rudimentary Algorithm 10A below. Details of refinements are provided in upcoming Sections 10.3 and 10.4.

Rudimentary Cycle Direction Search of the OOI Application

Figure 10.11 summarizes costs and capacities of the OOI application, along with the starting feasible flows of Figure 10.8, which have cost \$32,540. Figure 10.12 then details an application of Algorithm 10A to compute an optimal flow.

We first apply the improving feasible cycle direction of Figure 10.9. For that cycle $\lambda^+ = +\infty$ and $\lambda^- = \min\{610, 1000\} = 610$. A maximum feasible step of $\lambda = 610$ in that direction yields flow $\mathbf{x}^{(1)}$ at cost \$27,050.

Next we employ cycle 2–3–1–4–2 at $\bar{c} = -2$ and $\lambda = 560$ to produce flow $\mathbf{x}^{(2)}$ at cost \$25,930. A final step in the direction for the indicated cycle 2–3–4–2 completes recovery of the optimal flow in Figure 10.3 with value \$25,855.

The reader can verify that each of the three directions employed was both improving and feasible for the solution to which it was applied. However, many other choices were available. Pending the development of Sections 10.3 and 10.4,

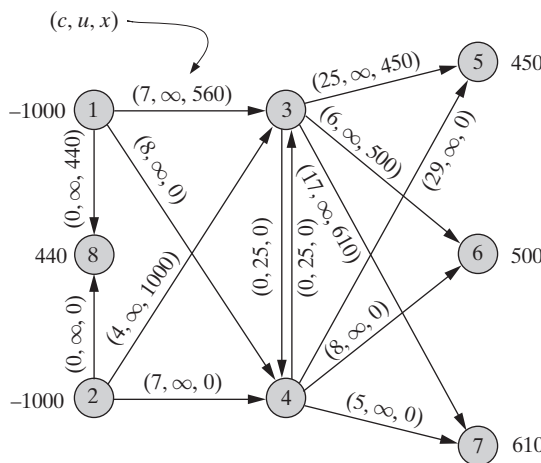


FIGURE 10.11 Data and Initial Flow for the OOI Application

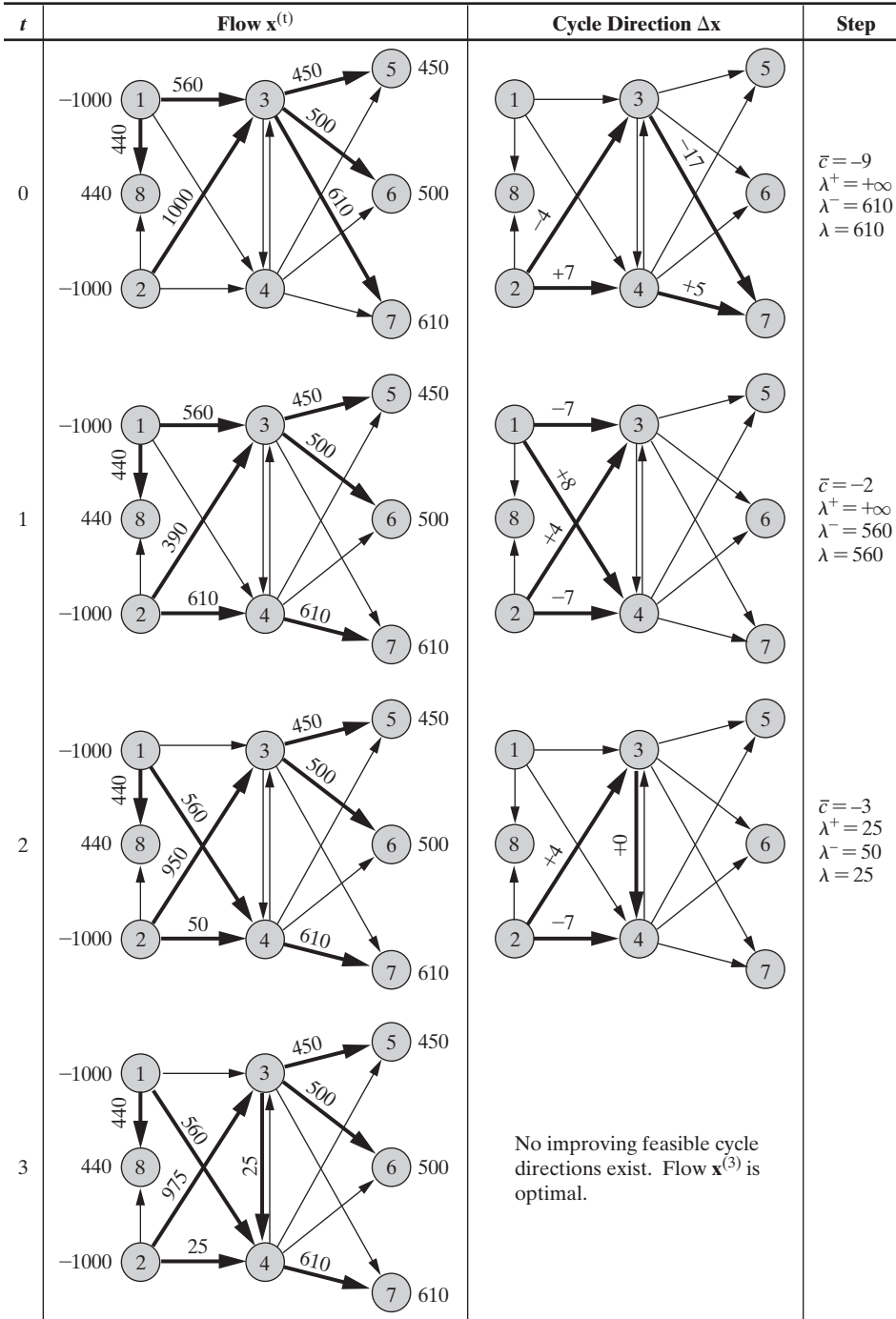


FIGURE 10.12 Rudimentary Cycle Direction Solution of OOI Application

ALGORITHM 10A: RUDIMENTARY CYCLE DIRECTION SEARCH

Step 0: Initialization. Choose any starting feasible flow $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Optimal. If no improving feasible cycle direction $\Delta\mathbf{x}$ exists at current solution $\mathbf{x}^{(t)}$ (principles [10.15] and [10.16]), then stop. Flow $\mathbf{x}^{(t)}$ is globally optimal.

Step 2: Cycle Direction. Choose an improving feasible cycle direction $\Delta\mathbf{x}$ at $\mathbf{x}^{(t)}$.

Step 3: Step Size. Compute the maximum feasible step λ in direction $\Delta\mathbf{x}$ (principle [10.17]):

$$\begin{aligned} \lambda^+ &\leftarrow \min\{ (u_{ij} - x_{ij}^{(t)}) : (i, j) \text{ forward} \} \text{ (+}\infty \text{ if none)} \\ \lambda^- &\leftarrow \min\{ x_{ij}^{(t)} : (i, j) \text{ reverse} \} \text{ (+}\infty \text{ if none)} \\ \lambda &\leftarrow \min\{ \lambda^+, \lambda^- \} \end{aligned}$$

If $\lambda = \infty$, stop; the model is unbounded.

Step 4: Advance. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda\Delta\mathbf{x}$$

by increasing flows on forward arcs of the cycle direction and decreasing those on reverse arcs by the amount λ . Then increment $t \leftarrow t + 1$ and return to Step 1.

readers should assume that choices were made arbitrarily, with appropriate cycles discovered by trial and error.

Many cycle directions remain in the optimal flow $\mathbf{x}^{(3)}$. Some are either improving or feasible. Still, it can be shown that no cycle direction is both improving and feasible. Algorithm 10A terminates with a conclusion of optimality.

10.3 CYCLE CANCELLING ALGORITHMS FOR OPTIMAL FLOWS

Algorithm 10A in Section 10.2 outlines the common computational logic of many network flow algorithms. The big question to be answered is, “How do we identify improving feasible cycle directions at each iteration, or prove that none exists?” This section presents the **cycle cancelling** approach often considered the most efficient.

Residual Digraphs

The cycle cancelling method (and many other network flow procedures) begins each iteration by constructing a residual digraph that details available options for improving feasible cycle directions. Each arc in the original digraph yields up to two

in the residual one, depending on whether its present flow can feasibly increase or decrease or both.

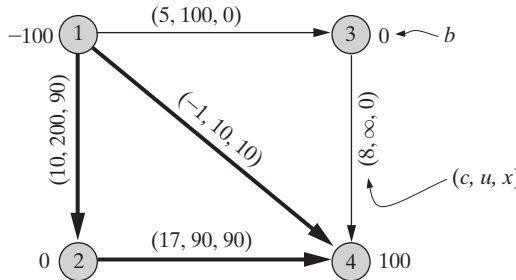
Definition 10.20 The **residual digraph** associated with current feasible flow $\mathbf{x}^{(t)}$ has the same nodes as the given network. It has one “increase arc” (i, j) , with cost $c_{i,j}$, for each increasable arc flow $x_{i,j}^{(t)} < u_{i,j}$, and one (backward) “decrease arc” (j, i) , with cost $-c_{i,j}$, for each decreasable arc flow $x_{i,j}^{(t)} > 0$.

Figure 10.13 illustrates for the OOI starting conditions depicted in Figure 10.11. Every flow that can both increase and decrease yields two opposed arcs in the residual digraph. For example, arc $(2, 3)$ with current flow $x_{2,3}^{(0)} = 1000$ produces both residual graph arc $(2, 3)$ and arc $(3, 2)$. The first, increase arc represents the fact that $x_{2,3}$ can feasibly become larger. Its cost is $c_{2,3} = 4$, the unit cost of such a flow increase. Decrease arc $(3, 2)$ shows that the flow can also become smaller without losing feasibility. It has cost $-c_{2,3} = -4$, the unit savings from a flow decrease.

Flows that cannot both increase and decrease produce just one arc in the residual digraph. For example, flow $x_{1,4}^{(0)} = 0$. It can only increase, so the residual graph has only arc $(1, 4)$ at cost $c_{1,4} = 8$.

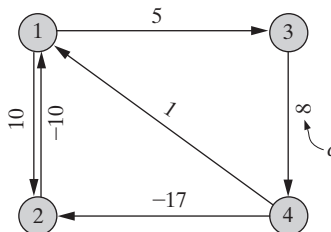
EXAMPLE 10.11: CONSTRUCTING RESIDUAL DIGRAPHS

Consider the network flow problem sketched below. Numbers on nodes show net demand b_k , and those on arcs show costs, capacities, and current flows $(c_{i,j}, u_{i,j}, x_{i,j})$.



Construct the corresponding residual digraph.

Solution: Applying principle 10.20, the residual digraph is



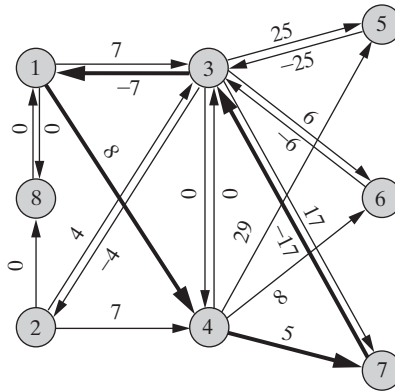


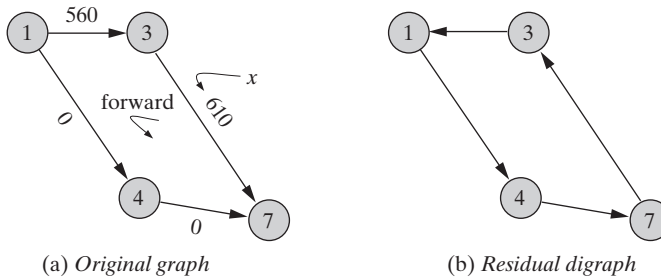
FIGURE 10.13 Residual Digraph for OOI Application Flow of Figure 10.12

Feasible Cycle Directions and Dicycles of Residual Digraphs

The point of constructing a residual digraph is to make it easier to identify improving feasible cycle directions in the original graph. Notice first that any cycle direction meeting principle 10.15’s requirements for feasibility—forward flows below capacity, reverse flows positive—now corresponds to a dicycle (definition 10.12) in the residual graph. A backward, decrease arc has been provided for each reverse arc of the cycle.

Principle 10.21 Cycle direction $\Delta \mathbf{x}$ is feasible for a current flow $\mathbf{x}^{(t)}$ if and only if the residual graph for $\mathbf{x}^{(t)}$ contains a dicycle with forward arcs of the $\Delta \mathbf{x}$ cycle corresponding to increase arcs in the residual dicycle, and reverse arcs of the cycle corresponding to decrease arcs.

One example is cycle 7–3–1–4–7 of the original digraph.



The corresponding cycle direction is feasible because forward arcs (1, 4) and (4, 7) are below capacity, and reverse arcs (3, 7) and (1, 3) have positive flow. It follows that each forward arc of (a) corresponds to an increase arc in (b), and each reverse arc of (a) yields a (backward) decrease arc in (b).

Improving Feasible Cycle Directions and Negative Dicycles of Residual Digraphs

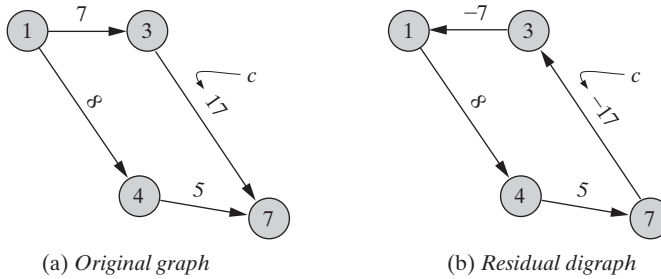
We seek improving feasible cycle directions, those with negative reduced costs:

$$\bar{c} = (\text{total forward arc cost}) - (\text{total reverse arc cost})$$

But under construction [10.20], forward arcs appear in the residual digraph as increase arcs with the same cost, and reverse arcs appear as backward, decrease arcs with the sign of their costs switched. It follows that the \bar{c} of a feasible cycle direction is exactly the length of the corresponding dicycle in the residual digraph. Dicycles of negative length yield what we require.

Principle 10.22 Feasible cycle direction $\Delta \mathbf{x}$ is improving if and only if the corresponding residual graph dicycle is a negative dicycle.

Again we can illustrate with the direction for OOI application cycle 7–3–1–4–7.



The original cycle is improving because the corresponding dicycle has negative total length.

EXAMPLE 10.12: CONNECTING ORIGINAL AND RESIDUAL DIGRAPHS

Return to the example of Example 10.11.

- (a) Show that the direction for cycle 1–3–4–2–1 is both feasible and improving.
- (b) Identify the corresponding dicycle in the residual digraph, and verify that it is a negative dicycle.

Solution:

(a) The direction for cycle 1–3–4–2–1 is feasible because forward flows (1, 3) and (3, 4) are below capacity, and reverse flows (2, 4) and (1, 2) are positive. It is improving because

$$\bar{c} = (5 + 8) - (17 + 10) = -14$$

(b) The residual digraph of Example 10.11 contains dicycle 1–3–4–2–1 with increase arcs (1, 3) and (3, 4), plus decrease arcs (4, 2) and (2, 1). Its total length is

$$5 + 8 - 17 - 10 = -14 = \bar{c}$$

Using Shortest Path Algorithms to Find Cycle Directions

Principles 10.21 and 10.22 reduce our search for improving feasible cycle directions to a hunt for negative dicycles in the residual digraph. The advantage is that we already know algorithms for identifying negative dicycles or proving that none exists.

The Floyd–Warshall shortest path Algorithm 9B, which we derived in Section 9.4, is one. That procedure is designed primarily to compute shortest paths from all nodes of a graph to all other nodes. If the given graph contains a negative dicycle, shortest path computation fails, but a negative dicycle is returned. Interchanging roles to make negative dicycles the normal outcome and completion of shortest path calculations the exception produces exactly the improving feasible cycle direction subroutine that we require.

Principle 10.23 Application of Floyd–Warshall Algorithm 9B to the residual digraph for a current feasible flow either yields a negative dicycle, indicating that the direction for the corresponding cycle of the original graph is both improving and feasible, or completes a shortest path computation, proving that improving feasible cycle directions do not exist.

EXAMPLE 10.13: FINDING DIRECTIONS WITH FLOYD–WARSHALL

Apply Floyd–Warshall Algorithm 9B to the residual digraph of Example 10.11 either to produce an improving feasible cycle direction or to prove that none exists.

Solution: After its third main iteration, Floyd–Warshall computation produces the following shortest path and inbound node results:

<i>k</i>	$v^{(0)}[k, \ell]$				$d[k, \ell]$			
	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
1	0	10	5	13	–	1	1	3
2	–10	0	–5	3	2	–	1	3
3	∞	∞	0	8	–	–	–	3
4	–27	–17	–22	–14	2	4	1	3

Computation halts with the conclusion that there is a negative dicycle because diagonal distance $v^{(3)}[4, 4] = -14$ is negative. Using $d[k, \ell]$ labels to recover the dicycle, $d[4, 4] = 3$, $d[4, 3] = 1$, $d[4, 1] = 2$, $d[4, 2] = 4$. Thus 4–2–1–3–4 is a negative dicycle of the residual graph. We already know from Example 10.12 that the corresponding cycle direction is indeed improving and feasible.

Cycle Cancelling Solution of the OOI Application

Algorithm 10B formalizes principle 10.23 in the Cycle Cancelling method for network flow optimization. Figure 10.14 details its application to our OOI Application of Figure 10.11.

ALGORITHM 10B: CYCLE CANCELLING FOR NETWORK FLOWS

Step 0: Initialization. Choose any starting feasible flow $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Residual Digraph. Construct the residual digraph corresponding to the current flow $\mathbf{x}^{(t)}$ (principle [10.20]).

Step 2: Floyd–Warshall. Execute Algorithm 9B on the current residual graph. If Floyd–Warshall computation terminates with no indication of a negative dicycle, stop. No improving feasible cycle directions exist, and current flow $\mathbf{x}^{(t)}$ is globally optimal.

Step 3: Cycle Direction. Use Floyd–Warshall decision labels $d[k, \ell]$ to trace a negative dicycle in the residual graph, and construct the cycle direction $\Delta \mathbf{x}$ for the corresponding cycle in the original graph.

Step 4: Step Size. Compute the maximum feasible step λ in direction $\Delta \mathbf{x}$ (rule [10.17]):

$$\lambda^+ \leftarrow \min\{(u_{ij} - x_{ij}^{(t)}) : (i, j) \text{ forward}\} \text{ (+}\infty \text{ if none)}$$

$$\lambda^- \leftarrow \min\{x_{ij}^{(t)} : (i, j) \text{ reverse}\} \text{ (+}\infty \text{ if none)}$$

$$\lambda \leftarrow \min\{\lambda^+, \lambda^-\}$$

If $\lambda = \infty$, stop; the model is unbounded.

Step 5: Advance. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$$

by increasing flows on forward arcs of the cycle direction and decreasing those on reverse arcs by the amount λ . Then increment $t \leftarrow t + 1$ and return to Step 1.

As before, graphs on the left of Figure 10.14 show flows in the real network. Notice that the sequence of solutions differs from that of Figure 10.12 because different improving feasible cycle directions were used. Still, both finished at the same optimal flow.

Graphs at the right in Figure 10.14 depict the residual digraphs for each flow encountered. Details of Floyd–Warshall computation have been omitted, but the negative dicycles obtained are highlighted. For example, the first application of Algorithm 9B identified negative dicycle 7–3–1–4–7 in the residual digraph for initial flow $\mathbf{x}^{(0)}$. Its total length is $-17 - 7 + 8 + 5 = -11$. Application of rule [10.17] around the corresponding cycle in the real OOI network yields $\lambda = 560$. Then increasing flows by 560 on forward arcs (1, 4) and (4, 7), while decreasing by the same amount on reverse arcs (3, 7) and (1, 3) produces the indicated flow $\mathbf{x}^{(1)}$.

Computation continues until $t = 4$. There Floyd–Warshall terminates on the residual digraph without discovering a negative dicycle. We may conclude (principle [10.23])

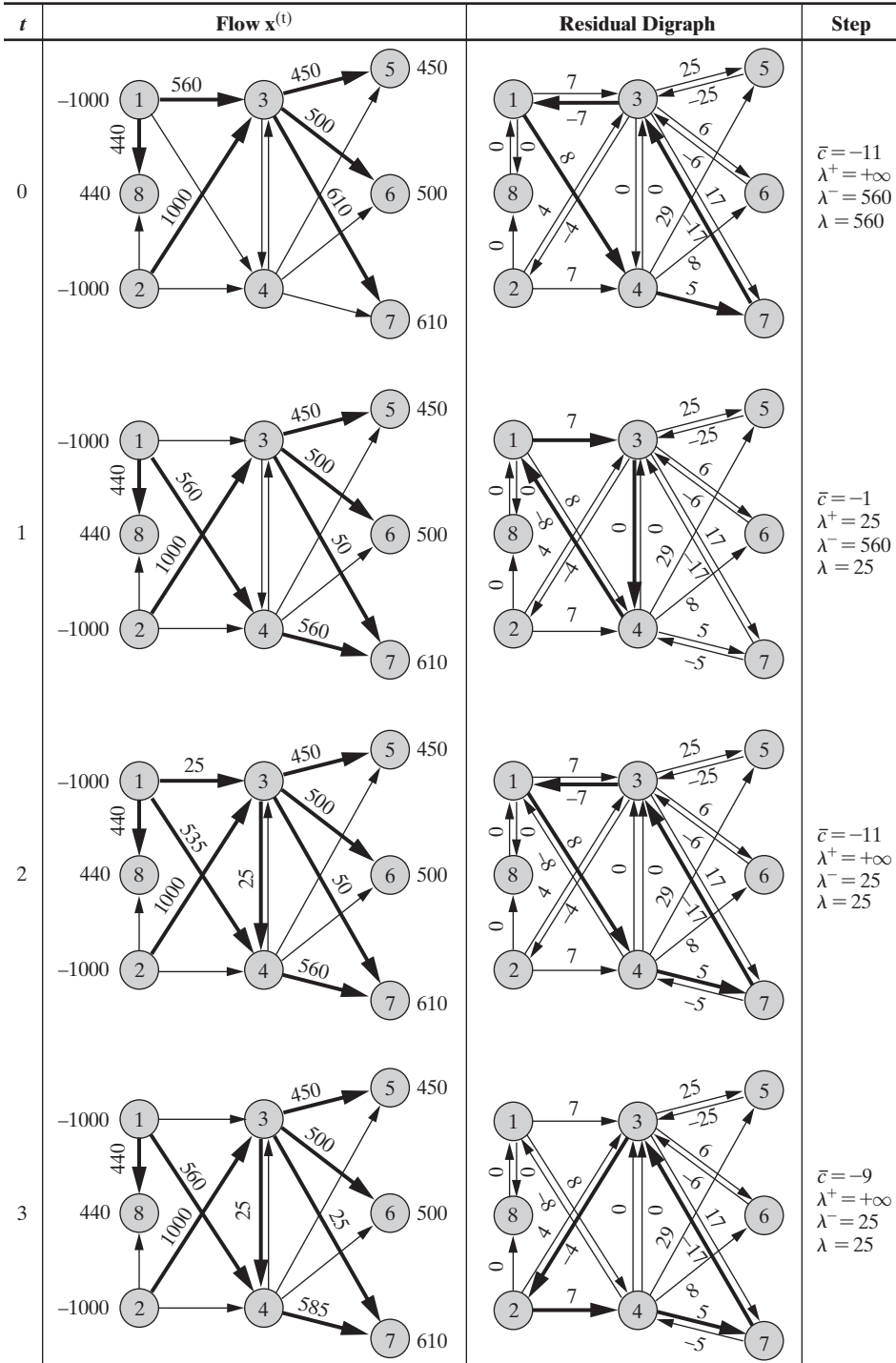


FIGURE 10.14 Cycle Cancelling Solution of the OOI Application

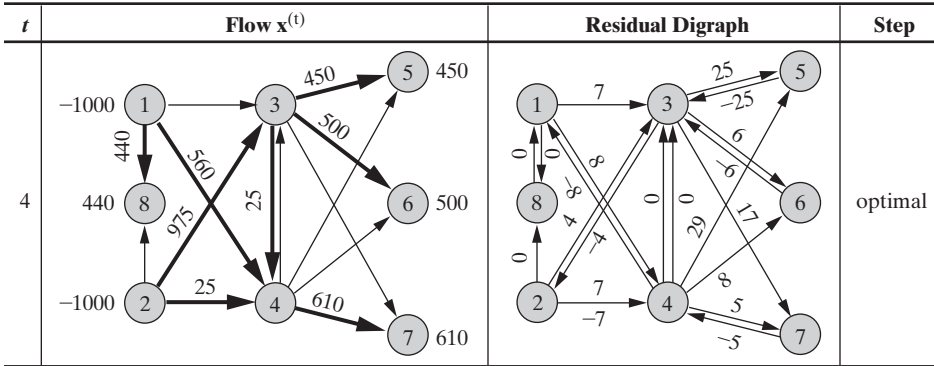


FIGURE 10.14 Cycle Cancelling Solution of the OOI Application (*Continued*)

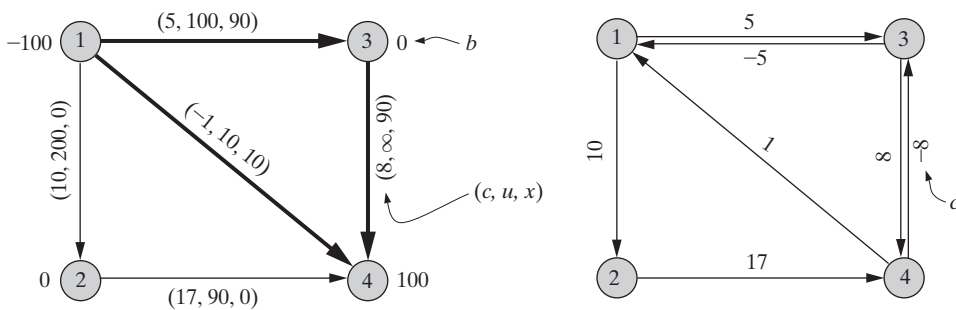
that no negative dicycle exists in the residual digraph, so that no improving feasible cycle direction exists for our current flow. The flow must be optimal.

EXAMPLE 10.14: APPLYING THE CYCLE CANCELLING ALGORITHM

Apply Cycle Cancelling Algorithm 10B to the network flow problem of Example 10.11.

Solution: We have already seen in Example 10.13 that Floyd–Warshall computation on the initial residual digraph yields negative dicycle 4–2–1–3–4. Forward arcs in the corresponding cycle imply that $\lambda^+ = \min\{(100 - 0), (\infty - 0)\} = 100$. Reverse arcs give $\lambda^- = \min\{90, 90\} = 90$. Thus step size $\lambda = \min\{100, 90\} = 90$.

Adjustment by this amount around cycle 4–2–1–3–4 produces the following new flow and residual digraph:



Floyd–Warshall computation on the new residual digraph will show that there is no negative dicycle. Thus the revised flow is optimal.

Polynomial Computational Order of Cycle Cancelling

Many of the details are beyond the scope of this book, (for more, see for example Ahuja, Magnanti, and Orlin, *Network Flows: Theory, Algorithms and Applications*, 1993). Still, we can outline how the general approach of Algorithm 10B can be

made rigorous enough to develop a bound on the number of steps needed to solve instances of network flows problems as described in Section 14.2.

Given a network flow problem on digraph $G(V, A)$ and a current flow \bar{x} and costs \mathbf{c} , Algorithm 10B first constructs a Residual Digraph \bar{G} of flow-change options available at the current solution, with costs c_{ij} on residual forward/increase arcs (i, j) and $-c_{ij}$ on opposed reverse/decrease arcs (j, i) . Then Algorithm 10B applies shortest-path methods at each iteration t to search for a negative total length dicycle in \bar{G} . If one can be found, it becomes the source of the next improving feasible cycle direction of the search. If no negative dicycle exists in \bar{G} , the current flow is optimal.

Let D^t denote the collection of arcs in the negative dicycle employed at iteration t . The first key to a computational bound on cycle cancelling is to use not just any negative dicycle at each iteration t , but the one of **minimum mean length (MML)** $\mu^t \triangleq \sum_{(i,j) \in D^t} c_{ij} / |D^t|$. That is, we wish to use directions derived from a dicycle in \bar{G} with least (most negative) average arc length. Fortunately, methods are available to find an MML dicycle efficiently.

Principle 10.24 | Shortest-path-style discrete dynamic programming methods like those of Chapter 9 can find an Minimum Mean Length dicycle D in any residual digraph \bar{G} of a given $G(V, A)$ in $O(|V||A|)$ time.

Next we need to normalize the search for MML dicycles with a set of (dual) node multipliers \mathbf{w} with components w_i for all $i \in V$ to obtain reduced \bar{A} costs $\bar{c}_{ij} \triangleq c_{ij} - w_i + w_j$. The interesting thing about switching to \bar{c}_{ij} is that the length of any dicycle D in \bar{G} is unchanged; each w_i subtracted on one arc of the dicycle is added back on the next to leave the reduced cost total and its mean unchanged. A clever choice can achieve more. Node multipliers \mathbf{w} can be found that (i) make $\bar{c}_{ij} \geq \mu^t |D^t|$ for all arcs in \bar{G} , and (ii) $\bar{c}_{ij} = \mu^t |D^t|$ for members of D .

Now consider how things change after a maximum step λ is taken along the direction for an MML dicycle D^t of the \bar{G} at iteration t . The next \bar{G} will have all the same arcs and reduced costs except around D^t . There, unless optimality has been reached, reduced costs induced by carefully chosen w_i will remain $\geq \mu^t$ except on one or more arcs (i, j) that established stepsize λ . Those arcs will be dropped in the next \bar{G} , and replaced by their opposites with $\bar{c}_{j,i} = -\bar{c}_{ij} \geq -\mu^t$. The consequence is that the length of the next MML dicycle cannot be less in \bar{c} terms than the last. Furthermore, since both original and reduced costs yield the same MML cycle length, the pattern must also hold for original costs.

Principle 10.25 | The sequence of minimum mean dicycle lengths $\{\mu^t\}$ computed at iterations t of the MML-cycle-cancelling Algorithm 10B is monotone non-decreasing.

How long can this go on? Consider first, sequences of iterations where all used MML dicycles have $\bar{c}_{ij} < 0$ on every member arc. Each iteration drops at least one such negative arc from \bar{G} , and replaces it by one with $\bar{c}_{ij} > 0$, so such all-negative steps can happen at most $O(|A|)$ times in a row.

Now consider the first t for which the chosen MML dicycle D^t has at least one member arc with a nonnegative reduced cost, yet optimality is not established, so $\mu^t < 0$. Then

$$\sum_{(i,j) \in D^t} c_{ij} = |D^t| \mu^t \geq (|D^t| - 1) \mu^t \geq (|D^t| - 1) \mu^{t-1}$$

The first inequality comes from dropping the nonnegative member in the sum and the second from property [10.25]. Dividing through by $|D^t|$ gives

$$\mu^t \geq (1 - 1/|D^t|) \mu^{t-1} \text{ or in the worst case } \mu^t \geq (1 - 1/|V|) \mu^{t-1}$$

Results from geometric series establish that any series reducing values by a factor of $(1 - 1/|V|)$ in each epoch will half the series value after no more than $|V|$ epochs. Thus we may conclude that after at most $O(|V|)$ epochs of at most $O(|A|)$ all negative MML dicycles, the magnitude of μ^t reduces by a factor of 1/2. In the worst case, this progress begins with a $\mu^0 \geq -c^{max}$ where c^{max} is the largest cost in the instance. Furthermore, if a $\mu^t > -1/|V|$ is reached in an integer-cost instance, the total cost of dicycle D^t cannot be any negative integer; $\mu^t \geq 0$ and the current flow is optimal. The consequence is that after at most $\log(|V|c^{max})$ reductions by 1/2, optimality must be reached. Combining these insights with principle [10.24] gives a bound on overall computation over integer data which is polynomial in instance size (see Chapter 14).

Principle 10.26 | Given a network flow problem over $G(V, A)$ and integer data including $c_{ij} \leq c^{max}$, the Minimum Mean Length implementation of Cycle Cancelling Algorithm 10B computes an optimal solution in at most $O(|V||A| \log(|V|c^{max}))$ iterations or $O(|V|^2|A|^2 \log(|V|c^{max}))$ total time.

10.4 NETWORK SIMPLEX ALGORITHM FOR OPTIMAL FLOWS

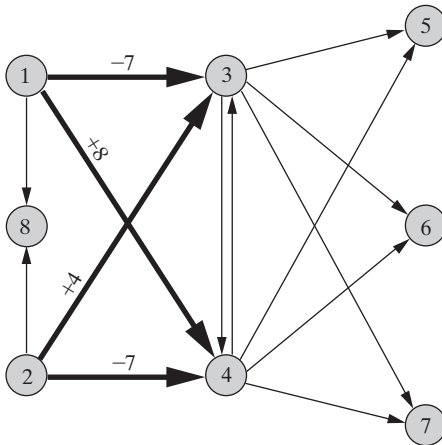
All the minimum cost network flow models of Sections 10.1–10.6 are linear programs, so the LP algorithms of Chapters 5 and 7 could be applied. Still, we have already seen in Section 10.3 how the elegant structure of cycle directions leads to much more efficient computations in the network setting.

In this section we fill in the details of rudimentary cycle direction Algorithm 10A to derive one such procedure. It is known as the **network simplex** because it specializes the usual simplex computations of Chapter 5, taking advantage of the fact that simplex directions turn out to be cycle directions in the network flow case.

Linear Dependence in Node–Arc Matrices and Cycles

Simplex search centers on **bases**—maximal collections of linearly independent columns drawn from the constraint matrix of main (standard form) constraints $\mathbf{Ax} = \mathbf{b}$. Columns of the main constraints in network flow models are columns of the node-arc incidence matrix [10.8]. They correspond to arcs. Thus, to understand simplex in the network context, we must first understand what collections of arcs correspond to linearly independent collections of columns.

Begin with cycles. For example, consider the OOI cycle (2, 3), (1, 3), (1, 4), (2, 4) illustrated in Figure 10.15. The figure displays both the cycle and the corresponding columns of the node–arc incidence matrix shown in Table 10.1.



Node	Arc			
	(2, 3)	(1, 3)	(1, 4)	(2, 4)
1	0	-1	-1	0
2	-1	0	0	-1
3	+1	+1	0	0
4	0	0	+1	+1
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

FIGURE 10.15 Cycle and the Corresponding Node–Arc Incidence Columns of OOI Application

Suppose that we apply weights of +1 on all column vectors for all forward arcs in the cycle, and -1 on columns of reverse arcs. Then

$$+ 1 \begin{pmatrix} 0 \\ -1 \\ +1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - 1 \begin{pmatrix} -1 \\ 0 \\ +1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} -1 \\ 0 \\ 0 \\ +1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - 1 \begin{pmatrix} 0 \\ -1 \\ 0 \\ +1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Our ± 1 weighted sum of the columns has produced the zero vector. This implies that the columns are linearly dependent because any one of the vectors can be expressed (by transposing) as a nonzero linear combination of the others.

The ± 1 weights we have chosen are exactly those of the corresponding cycle direction [10.13]. Thus for the same (Figure 10.7) reasons that cycle directions satisfy

$\mathbf{A}\Delta\mathbf{x} = \mathbf{0}$, such a ± 1 linear combination of node–arc incidence columns for the arcs of a cycle will always yield the zero vector.

Principle 10.27 | Node–arc incidence matrix columns for arcs of a cycle form a linearly dependent set.

Since column vectors of a basis must be linearly independent, we have an immediate consequence.

Principle 10.28 | Basic sets of arcs for minimum cost network flow models can contain no cycles.

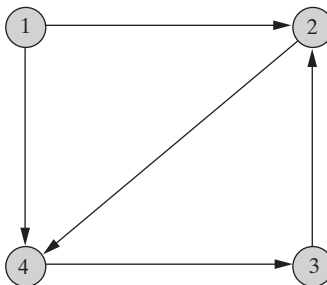
Cycles are not the only way to produce linearly dependent sets in network flow models. For example, return to the complex direction of Figure 10.10. That weighting of node–arc incidence columns also produces the zero vector because the direction satisfies $\mathbf{A}\Delta\mathbf{x} = \mathbf{0}$. Thus the collection of arcs with nonzero weights is linearly dependent.

Still, we know from principle 10.19 that complex directions like that of Figure 10.10 decompose into weighted sums of cycle directions. This implies that each includes at least one cycle.

Principle 10.29 | Every linearly dependent arc set in a minimum cost network flow model contains a cycle.

EXAMPLE 10.15: CHECKING LINEAR INDEPENDENCE OF ARCS

Consider the following digraph:



Determine which of the following arc sets could be part of a basis for the corresponding node–arc incidence matrix.

- (a) $\{(1, 2), (4, 3)\}$
- (b) $\{(2, 4), (4, 3), (3, 2)\}$
- (c) $\{(1, 2), (1, 4), (4, 3), (2, 4)\}$
- (d) $\{(1, 2), (2, 4), (3, 2)\}$

Solution:

- (a) This set is linearly independent under principle [10.29] because it contains no cycle. It could be part of a basis.
- (b) These arcs form a cycle. Thus, under principle [10.28], they cannot be part of a basis.
- (c) This set is linearly dependent because it contains cycle 1–2–4–1. Under principle [10.28], it cannot be part of a basis.
- (d) This set is linearly independent under principle [10.29] because it contains no cycle. It could be part of a basis.

Spanning Trees of Networks

We say that a graph is **connected** if there is a chain between every pair of nodes. A graph is a **tree** if it is connected and contains no cycles. A tree is a **spanning tree** if it touches every node of a graph.

Figure 10.16 illustrates these definitions. Part (a) shows a spanning tree. It is connected, it contains no cycles, and it touches all nodes of the OOI network. Parts (b) to (d) demonstrate the various ways that graphs can fail the definition of a spanning tree.

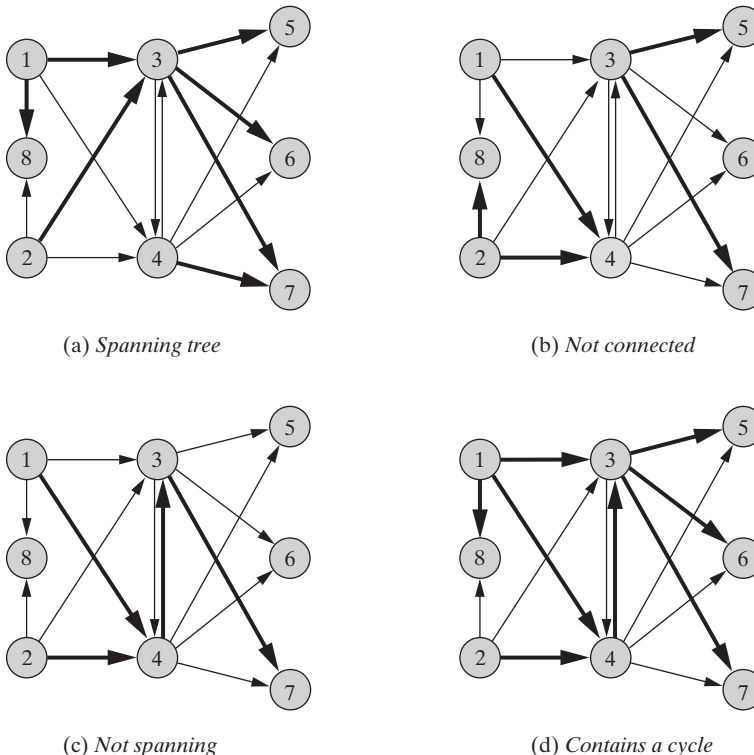


FIGURE 10.16 Notion of a Spanning Tree

The connected but cycle-free nature of trees leads to an important property:

Principle 10.30 | Every pair of nodes in a tree is connected by a unique chain of the tree. If the tree spans a graph, every pair of nodes in the graph is connected by a unique chain of the tree.

For example, nodes 8 and 4 are connected in the spanning tree of Figure 10.16(a) by the chain 8–1–3–7–4 and by no other. If there were two such chains, they would form a cycle, and there must be one if the tree is connected.

Spanning Tree Bases for Network Flow Models

Returning to the issue of bases for node–arc incidence matrices, the example of Figure 10.16(d) has no hope of being a basis because it contains a cycle. Corresponding columns of the node–arc incidence matrix will be linearly dependent (principle 10.27). The graphs in parts (a) to (c) have no such cycles. Thus principle 10.29 assures us that corresponding columns are linearly independent. But are they a basis?

A basis must be a maximal linearly independent set (i.e., it should be impossible to enlarge it without creating a dependency). Notice that the examples in Figure 10.16(b) and (c) fail this test. We know from principle 10.29 that a dependency will be created only if we form a cycle. That cannot occur when we insert an arc between previously unconnected nodes. For example, adding arc (4, 3) in Figure 10.16(b) preserves linear independence. The same is true for adding arc (3, 5) in Figure 10.16(c).

The spanning tree of Figure 10.16(a) is different. Because there is already a unique chain between every pair of nodes (principle 10.30), inserting any other arc closes a cycle. For example, adding arc (2, 4) closes cycle 2–4–7–3–2.

Principle 10.31 | Adding an arc to a spanning tree produces a unique cycle.

It follows immediately that spanning trees correspond to maximal linearly independent sets and no bases. We have reached the key to understanding simplex computations in minimum cost network flow models.

Principle 10.32 | A collection of columns in the node–arc incidence matrix of a minimum cost network flow problem forms a basis if and only if corresponding arcs form a spanning tree of the associated digraph.

EXAMPLE 10.16: IDENTIFYING NETWORK FLOW BASES

Determine which of the four arcs sets in Example 10.15 forms a basis for the corresponding network flow problem.

Solution: We have already seen in Example 10.15 that sets (b) and (c) are linearly dependent. Set (a) is linearly independent, but it is not a basis under principle 10.32 because the arcs do not form a spanning tree. Set (d) does form a spanning tree. Thus it is the only basis of the four.

Network Basic Solutions

Simplex algorithms proceed from one basic feasible solution to another. For the network case, where capacities or upper bounds are present, we define basic solution in the upper-/lower-bound sense of Section 5.9 (principle [5.48]).

Principle 10.33 In **basic solutions** for network flow problems, nonbasic arcs have flow equal to either 0 or capacity $u_{i,j}$. Basic arcs have the unique flow achieving flow balance for the nonbasic values specified. The flow is **basic feasible** if all basis flows are within bounds.

Figure 10.17 illustrates for the OOI application of Figure 10.2. The corresponding basis is highlighted. There is positive flow on nonbasic arc (4, 3), but it equals capacity $u_{4,3} = 25$. Only basic arcs can have flow $x_{i,j}$ other than zero or $u_{i,j}$.

To compute this basic solution, we first assign flow 0 to all nonbasics except (4, 3) and 25 to $x_{4,3}$. Only one choice of basic flows can then meet net demand requirements at each node. For example, with nonbasic flow $x_{3,7} = 0$, the only flowbalancing choice for basic flow $x_{4,7} = 610$. The solution of Figure 10.17 is basic feasible because all such basis flows are within bound limits.

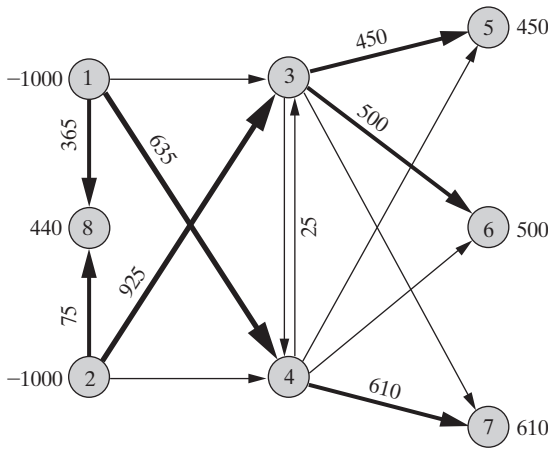
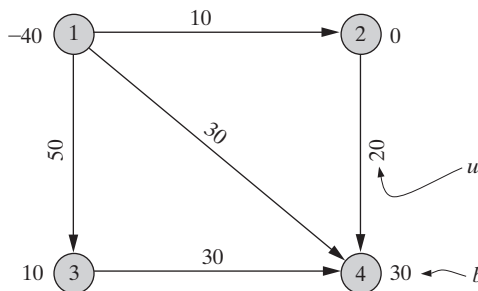


FIGURE 10.17 Initial Basic Solution for OOI Application

EXAMPLE 10.17: COMPUTING NETWORK BASIC SOLUTIONS

In the following network, numbers on nodes indicate net demands b_k and those on arcs show capacities $u_{i,j}$.



For each of the following choices on nonbasic arcs, compute the corresponding basic solution and determine whether it is basic feasible.

- (a) (1, 4) nonbasic lower-bounded, (2, 4) nonbasic upper-bounded
 (b) (3, 4) nonbasic lower-bounded, (1, 2) nonbasic upper-bounded

Solution: We apply principle [10.33](#).

(a) Here the basis is $\{(1, 2), (1, 3), (3, 4)\}$. With $x_{1,4} = 0$ and $x_{2,4} = 20$, the unique choice of basic values that meets net demand requirements at all nodes is $x_{1,2} = x_{1,3} = 20, x_{3,4} = 10$. This basis solution is not basic feasible because flow $x_{1,2}$ exceeds capacity $u_{1,2} = 10$.

(b) Here the basis is $\{(1, 3), (1, 4), (2, 4)\}$. With $x_{3,4} = 0$ and $x_{1,2} = 10$, the unique choice of basic values that meets net demand requirements at all nodes is $x_{1,3} = x_{2,4} = 10, x_{1,4} = 20$. Since all basic flows are within bounds, this basic solution is basic feasible.

Simplex Cycle Directions

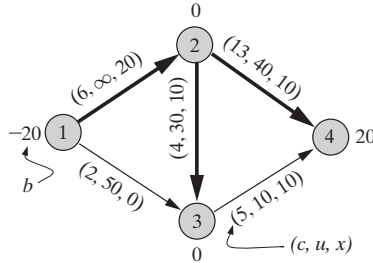
Simplex directions (principle [5.20](#) of Section 5.3) in linear programs are formed by increasing one nonbasic variable from its lower bound (or decreasing a nonbasic from its upper bound) and changing basic variables in the unique way assuring that main constraints remain satisfied. For network flow cases, every basic set of columns (arcs) forms a spanning tree (principle [10.32](#)). Thus introducing any nonbasic arc creates a unique cycle (principle [10.31](#)), and the corresponding cycle direction must be the simplex direction.

Definition 10.34 The **network simplex direction** for increasing nonbasic arc at flow $= 0$ is the cycle direction of the unique cycle formed when the corresponding nonbasic arc is introduced into the current basis tree and the orientation is the same as that arc. The simplex direction for decreasing a nonbasic arc at capacity flow $u_{i,j}$ is the cycle direction obtained when the corresponding nonbasic arc is introduced and the orientation is opposite to that arc.

Simplex cycle directions need not be feasible because some decreasing basic flow may already equal zero or some increasing basic flow may be at capacity. As with other simplex methods, however, we may ignore such **degeneracy** without much practical effect except step sizes $\lambda = 0$ (definition [5.39](#)).

EXAMPLE 10.18: CONSTRUCTING NETWORK SIMPLEX DIRECTIONS

Consider the following network. Numbers next to nodes are net demands b_k , and those on arcs are cost, capacities, and current flows $(c_{i,j}, u_{i,j}, x_{i,j})$.



Taking highlighted arcs as the basis, construct the simplex directions for all nonbasic arcs.

Solution: We apply principle [10.34]. Introducing nonbasic arc (1, 3) produces unique cycle 1–3–2–1. Since nonbasic arc (1, 3) is presently at flow 0, the corresponding simplex direction will be the cycle direction for 1–3–2–1 with (1, 3) a forward arc. That is, $\Delta x_{1,3} = +1, \Delta x_{2,3} = \Delta x_{1,2} = -1$.

Nonbasic arc (3, 4) is presently at capacity. To find its cycle/simplex direction we pass the unique cycle it forms in the basis so that (3, 4) is a reverse arc. That is, we employ cycle direction 4–3–2–4 with $\Delta x_{3,4} = \Delta x_{2,3} = -1, \Delta x_{2,4} = +1$.

Network Simplex Algorithm

Algorithm 10C combines these ideas in a special simplex version of cycle direction-based network search, Algorithm 10A. Instead of considering all possible cycle directions at each iteration, we check only the simplex directions induced by the unique cycles for increasing or decreasing nonbasic arcs. Simplex theory for all linear programs (principle [5.26] of Section 5.3) tells us that these directions are sufficient to find an optimal solution.

Comparison of this algorithm statement with the general simplex computations of Chapter 5 will highlight the convenience made possible by the special network flow structure. To find simplex directions and determine whether they improve the objective function, we need only trace our way through the basis tree.

Network Simplex Solution of OOI Application

Figure 10.18 details solution of our OOI application by Algorithm 10C, starting from the initial basic feasible solution in Figure 10.17. At the first iteration, the simplex directions for the 7 nonbasic arcs are as follows:

Nonbasic	Cycle	\bar{c}
Increase (1, 3)	1–3–2–8–1	3
Increase (2, 4)	2–4–1–8–2	-1
Increase (3, 4)	3–4–1–8–2–3 via (3, 4)	-4
Increase (3, 7)	3–7–4–1–8–2–3	8
Increase (4, 3)	3–4–1–8–2–3 via (4, 3)	-4
Increase (4, 5)	4–5–3–2–8–1–4	8
Increase (4, 6)	4–6–3–2–8–1–4	6

ALGORITHM 10C: NETWORK SIMPLEX SEARCH

Step 0: Initialization. Choose any starting basic feasible flow $\mathbf{x}^{(0)}$, identify the corresponding basis spanning tree, and set solution index $t \leftarrow 0$.

Step 1: Simplex Directions. For each nonbasic arc, examine the simplex direction associated with the unique cycle that arc forms in the basis spanning tree and apply test [10.16](#) to determine whether the direction improves.

Step 2: Optimal. If no simplex cycle direction improves, stop; flow $\mathbf{x}^{(t)}$ is globally optimal. Otherwise, choose as $\Delta\mathbf{x}$ some improving simplex cycle direction, and let (p, q) denote the corresponding nonbasic arc.

Step 3: Step Size. Compute the maximum feasible step λ in direction $\Delta\mathbf{x}$ (principle [10.17](#)):

$$\lambda^+ \leftarrow \min\{(u_{i,j} - x_{i,j}^{(t)}) : (i, j) \text{ forward}\} \text{ (+}\infty \text{ if none)}$$

$$\lambda^- \leftarrow \min\{x_{i,j}^{(t)} : (i, j) \text{ reverse}\} \text{ (+}\infty \text{ if none)}$$

$$\lambda \leftarrow \min\{\lambda^+, \lambda^-\}$$

If $\lambda = +\infty$, stop; the model is unbounded.

Step 4: Advance. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda\Delta\mathbf{x}$$

by increasing flows on forward arcs of the cycle direction and decreasing those on reverse arcs by the amount λ .

Step 5: New Basis. If some arc other than (p, q) established the minimum λ in step 3, replace any such arc in the basis spanning tree by (p, q) .

Increment $t \leftarrow t + 1$ and return to step 1.

Any simplex direction with $\bar{c} < 0$ could provide a means of improving flow $\mathbf{x}^{(0)}$. Figure 10.18 employs the one associated with increasing nonbasic, lower-bounded arc (2, 4). The unique cycle formed in the basis tree immediately identifies the rest of the associated cycle direction. Arcs (2, 4) and (1, 8) increase; (1, 4) and (2, 8) decrease. Thus $\lambda^+ = \infty$, $\lambda^- = \min\{75, 635\}$, and $\lambda = 75$. Arc (2, 8) established the λ limit, so (2, 4) replaces (2, 8) in the basis. Updating produces flow $\mathbf{x}^{(1)}$ of Figure 10.19. Notice that the new basis is also a spanning tree.

The simplex direction decreasing nonbasic, upper-bounded arc (4, 3) now improves for $\mathbf{x}^{(1)}$. The unique cycle formed by (4, 3) in the current basis yields a cycle direction decreasing (4, 3) and (2, 4) while increasing (2, 3). Move limit λ for this direction occurs when nonbasic (4, 3) reaches its lower bound of 0. Thus flows are updated, but the basis remains unchanged.

Flow $\mathbf{x}^{(2)}$ in Figure 10.18 is the result. Increasing nonbasic (3, 4) can improve this flow because $\bar{c} = -3$ on the corresponding simplex direction. Adjustment of $\lambda = 25$ around the implied cycle direction improves to flow $\mathbf{x}^{(3)}$. Again incoming arc (3, 4) itself stops improvement, so the basis remains unchanged.

t	Flow $x^{(t)}$	Simplex Direction Δx	Step
0			<p>(2, 4) increases $\bar{c} = -1$ $\lambda^+ = +\infty$ $\lambda^- = 75$ $\lambda = 75$ (2, 4) replaces (2, 8)</p>
1			<p>(4, 3) decreases $\bar{c} = -3$ $\lambda^+ = +\infty$ $\lambda^- = 25$ $\lambda = 25$ no basis change</p>
2			<p>(3, 3) increases $\bar{c} = -3$ $\lambda^+ = 25$ $\lambda^- = 50$ $\lambda = 25$ no basis change</p>
3		<p>No improving simplex directions exist. Flow $x^{(3)}$ is optimal.</p>	

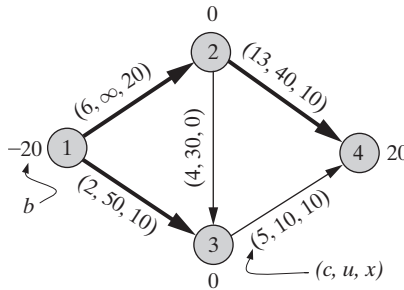
FIGURE 10.18 Network Simplex Solution of OOI Application

Comparison to Figure 10.3 will show that new flow $\mathbf{x}^{(3)}$ is now optimal. None of the 7 simplex cycle directions is improving.

EXAMPLE 10.19: APPLYING THE NETWORK SIMPLEX ALGORITHM

Return to the network flow example and starting basis of Example 10.18, and compute an optimal solution with network simplex Algorithm 10C.

Solution: At the first iteration, increasing (1, 3) produces a simplex direction for cycle 1–3–2–1 having $\bar{c}_{1,3} = 2 - 4 - 6 = -8$. Decreasing nonbasic (3, 4) yields the direction for cycle 4–3–2–4 with $\bar{c}_{3,4} = -5 - 4 + 13 = 4$. Thus we update along the former by $\lambda = 10$, and replace (2, 3) in the basis by (1, 2) because (2, 3) established λ . The new solution is



Now the cycle direction for increasing (2, 3) has $\bar{c}_{2,3} = 8$ and that of decreasing (3, 4) had $\bar{c}_{3,4} = 12$. Neither is improving, and the latest flow is optimal.

10.5 INTEGRALITY OF OPTIMAL NETWORK FLOWS

As far back as Chapter 2, we have seen that integer linear programs (ILPs) are generally much less tractable than linear programs (LPs). We are now prepared to show why network flows are often an exception. Under mild conditions, optimal network flows automatically take on integer (whole-number) values. Thus, if an ILP can be shown to have network form, we may solve it with methods based on Algorithm 10A–10C which are even more efficient than general-purpose LP methods of Chapters 5–7.

When Optimal Network Flows Must Be Integer

Suppose that all constraint data of a given minimum cost network flow model—supplies, demands, and capacities—happen to have integer values as they do in the OOI application of Figure 10.11. Then we can think through some observations about the steps in a two-phase or big- M implementation of Algorithm 10A that have important consequences.

- The starting artificial flow of principle 10.6 will be integer because the only nonzero flows occur on artificial arcs, and all those flows equal supplies or demands that we have assumed integer.

- Regardless of whether we proceed by two-phase or big- M application of Algorithm 10A, an optimal solution can be computed using only cycle directions (principle [10.18](#)).
- Each iteration of Algorithm 10A either increases arc flows by step size λ , decreases them by λ , or leaves the flows unchanged, because all components of cycle directions are $+1$, -1 , or 0 (definition [10.13](#)).
- Whenever the current flow $\mathbf{x}^{(t)}$ is integer, step size λ of [10.17](#), which will equal either some $x_{i,j}^{(t)}$ or some $(u_{i,j} - x_{i,j}^{(t)})$, must also be integer under our assumption of integer $u_{i,j}$.

The consequence is a fundamental **integrality property**. When a network flow model has integer constraint data, we can start with an integer flow and execute every step of two-phase of big- M solution by adding or subtracting integer quantities λ to current flows. If the result is an optimal solution (i.e., not infeasible or unbounded), that solution must have integer flows on all arcs.

Principle 10.35 | If a minimum cost network flow model with integer constraint data (supplies, demands, and capacities) has any optimal solution, it has an integer optimal solution.

Notice one assumption not required in principle [10.35](#). Nothing was said about arc costs $c_{i,j}$. Integrality property [10.35](#) depends only on constraint data. What is important is that flows start integer and change by integer amounts at each step. Costs have no effect because only supplies, demands, and capacities are involved in flow adjustments.

EXAMPLE 10.20: RECOGNIZING IF NETWORK OPTIMA WILL BE INTEGER

Each of the following details the constraint data of a minimum cost network flow problem. Assuming that the models are neither infeasible nor unbounded, determine whether each must have an integer optimal solution.

- (a) $\mathbf{b} = (100, 200, 0, -300)$, $\mathbf{u} = (90, 20, \infty, 220, 180)$, $\mathbf{c} = (8, 9, -4, 0, 6)$
- (b) $\mathbf{b} = (-30, 40, -10, 0)$, $\mathbf{u} = (20, 12\frac{1}{2}, 23, 15, 92)$, $\mathbf{c} = (11, 0, 3, 81, 6)$
- (c) $\mathbf{b} = (-13\frac{1}{3}, -20, 23\frac{1}{3}, 10)$, $\mathbf{u} = (10, 20, \infty, \infty, 40)$, $\mathbf{c} = (-4, 8, 0, 19, 31)$
- (d) $\mathbf{b} = (25, 15, 0, -40)$, $\mathbf{u} = (20, \infty, 30, 45, 10)$, $\mathbf{c} = (3.5, 9.6, -2.1, 11.77, \sqrt{2})$

Solution: We apply principle [10.35](#).

- (a) This model has integer supplies, demands, and capacities. An integer optimal solution will exist.
- (b) This model has a fractional capacity. An integer optimum is not assured.
- (c) This model has a fractional supply and a fractional demand. An integer optimum is not assured.
- (d) Even though cost data are highly noninteger, this model has integer constraint data. An integer optimal solution will exist.

Total Unimodularity of Node–Arc Incidence Matrices

The above “starts integer and stays integer” argument is only one way of seeing that network flow optima will be integer when all constraint data are integer. Since we know that optimal solutions, at least unique ones, must be extreme points of the corresponding LP-feasible region (Section 5.1, principle [5.4](#)), there must also be something special about the basic solution computations of extreme points (Section 5.2) in minimum cost network flow models.

The property is called total unimodularity.

Definition 10.36 | The constraint matrix \mathbf{A} of a linear program is **totally unimodular** if each square submatrix has determinant $+1$, 0 , or -1 .

The main constraint matrices of network flow models are their node–arc incidence matrices, and they have this very special property:

Principle 10.37 | Node–arc incidence matrices of network flow models are totally unimodular.

The relevance of total unimodularity comes in solving for basic (i.e., extreme-point) solutions to network LPs. The famous Cramer rule for solving systems of linear equations shows that the maximum denominator of the result is the determinant of the corresponding basis matrix. Under total unimodularity, that denominator will always be ± 1 , meaning that no fractions will be introduced.

An example from the OOI node–arc matrix of Table 10.1 will illustrate. Extracting rows for nodes 2, 3, and 4, along with columns for arcs (2, 3), (2, 4), and (3, 4), gives

$$\left(\begin{array}{c|ccc} & (2,3) & (2,4) & (3,4) \\ \hline 2 & -1 & -1 & 0 \\ 3 & +1 & 0 & -1 \\ 4 & 0 & -1 & +1 \end{array} \right)$$

All 1 by 1 submatrices are obviously $+1$, 0 , or -1 . The full 3 by 3 matrix has determinant $= 0$, its lower left corner yields

$$\det \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix} = -1$$

and all other 2 by 2 submatrices are similar.

The formal justification of principle [10.37](#) comes from an easy inductive argument on the size of the submatrix being considered. For size 1×1 , every entry in a node–arc incidence matrix is either 0 , 1 , or -1 , so its 1×1 determinant is the same. Now assume total unimodularity holds for all submatrices of size $k \times k$ or smaller, and consider a submatrix of size $k + 1 \times k + 1$. If the submatrix has an all-zero column, its determinant $= 0$. Otherwise, if all its columns have exactly one $+1$ and one -1 , which is the most non-zeros possible with a node–arc incidence

matrix, the row sum = 0, and the determinant = 0. The last possibility is that the matrix has a column with only nonzero ± 1 . But then the full determinant is ± 1 times the determinant of the $k \times k$ matrix excluding its row and column, which = 0, 1 or -1 by the inductive hypothesis.

10.6 TRANSPORTATION AND ASSIGNMENT MODELS

The OOI application of Section 10.1 contains all the major elements of network flow models, but it only begins to suggest the rich variety of problems that can be treated as networks. In this section we present the classic transportation and assignment special cases. Among the simplest are those posed on **bipartite graphs**—graphs on two nonoverlapping node sets S and T , with every arc or edge having one end in S and the other in T .

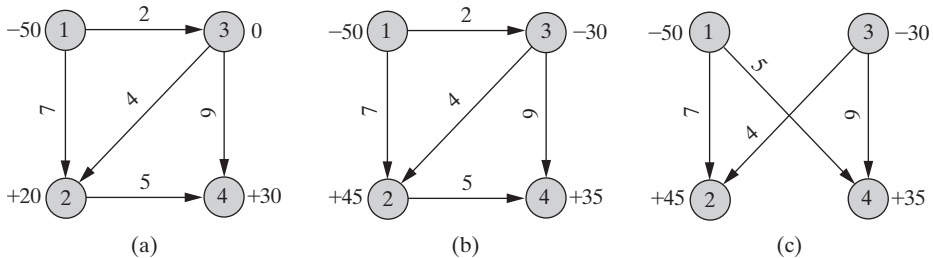
Transportation Problems

Definition 10.38 **Transportation problems** are special minimum cost network flow models for which every node is either a pure supply node (every arc points out) or a pure demand node (every arc points in).

That is, all flow goes immediately from some source node, where it is supplied, to a sink node where it is demanded. There are no intermediate steps and no transshipment nodes. The commodity flowing in a transportation problem may be people, water, oil, money, or almost anything else. It is only necessary that flows go direct from sources to sinks.

EXAMPLE 10.21: IDENTIFYING TRANSPORTATION PROBLEMS

Each of the following digraphs shows a minimum cost network flow problem. Numbers on vertices are net demands, b_k , and those on arcs are costs.



Determine which are transportation problems.

Solution:

(a) This network flow problem is not a transportation problem because node 3 is a pure transshipment node, with neither supply nor demand.

(b) This network flow problem is also not a transportation problem because nodes 2 and 3 have both inbound and outbound arcs.

(c) This network flow problem is a transportation problem. Nodes 1 and 3 are pure source nodes; nodes 2 and 4 are pure demand nodes.

Standard Form for Transportation Problems

Using constants

$s_i \triangleq$ supply at node i

$d_j \triangleq$ demand at node j

$c_{i,j} \triangleq$ unit cost of flow from i to j

and decision variables

$x_{i,j} \triangleq$ flow from i to j

the network flow formulation [10.3](#) of a transportation problem simplifies to

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{i,j} x_{i,j} \\ \text{s.t.} \quad & -\sum_j x_{i,j} = -s_i \quad \text{for all } i \\ & \sum_i x_{i,j} = d_j \quad \text{for all } j \\ & x_{i,j} \geq 0 \quad \text{for all } i, j \end{aligned}$$

As usual, flows leaving supply nodes carry negative signs, and those entering demand points have positive signs.

The more common statement of transportation problems in operations research uses the standard form obtained when signs are reversed on the first set of constraints.

Definition 10.39 For supplies s_i , demands d_j , and costs $c_{i,j}$, the **standard form of transportation problems** is

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_j x_{i,j} = s_i \quad \text{for all } i \\ & \sum_i x_{i,j} = d_j \quad \text{for all } j \\ & x_{i,j} \geq 0 \quad \text{for all } i, j \end{aligned}$$

EXAMPLE 10.22: FORMULATING TRANSPORTATION PROBLEMS

Write a standard-form transportation problem formulation for the diagram of Example 10.21(c).

Solution: Following [10.39], the model in standard form is

$$\begin{aligned}
 \min \quad & 7x_{1,2} + 5x_{1,4} + 4x_{3,2} + 9x_{3,4} \\
 \text{s.t.} \quad & x_{1,2} + x_{1,4} = 50 \\
 & x_{3,1} + x_{3,4} = 30 \\
 & x_{1,2} + x_{3,2} = 45 \\
 & x_{1,4} + x_{3,4} = 35 \\
 & x_{1,2}, x_{1,4}, x_{3,1}, x_{3,4} \geq 0
 \end{aligned}$$

APPLICATION 10.3: MARINE MOBILIZATION TRANSPORTATION PROBLEM

A really massive transportation problem application arises in officer mobilization planning for the U.S. Marine Corps.² During any international emergency, thousands of officers must be mobilized from their regular duty or reserve positions into billets required for the emergency. However, not every officer is qualified by rank, training, or experience to fulfill every assignment. Using the very efficient network flow methods of this chapter, Marine planners are able to develop a mobilization scheme in a few minutes by solving a transportation problem with over 100,000 arcs.

Mobilization options can be depicted in a digraph like the fictitious one of Figure 10.19. Supply nodes exist for each group of like-qualified officers presently

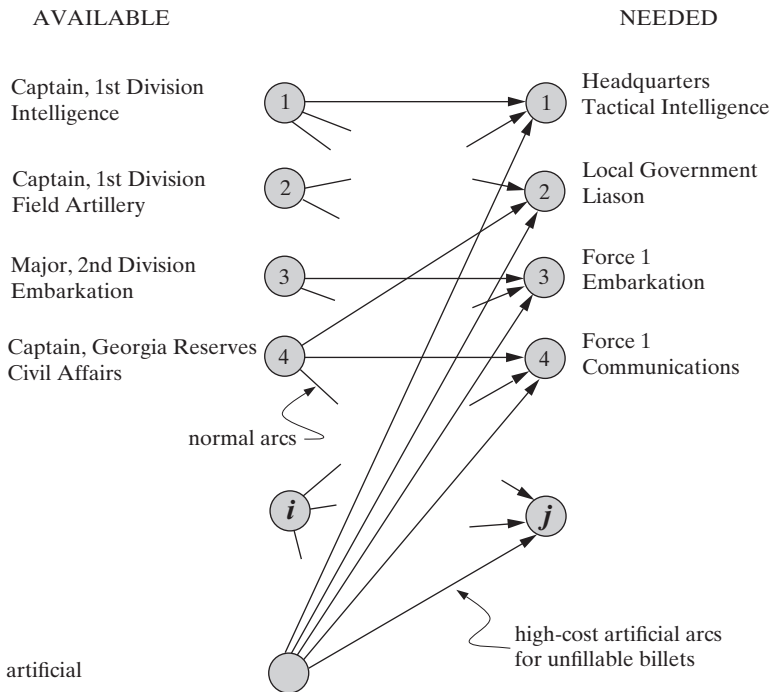


FIGURE 10.19 Marine Mobilization Transportation Problem

²Based on D. O. Bausch, G. G. Brown, D. R. Hundley, S. H. Rapp, and R. E. Rosenthal (1991), "Mobilizing Marine Corps Officers," *Interfaces*, 21:4, 26–38.

based in the same location. For example, Figure 10.19 shows one supply node for captains now in the 1st Division who are trained as intelligence officers. Another represents civil affairs–trained officers in Georgia units of the Marine Corps Reserve.

Demand nodes indicate needs for officers of particular qualifications at active locations of the emergency. For example, Figure 10.19 depicts the need for one or more local government liason officers in the forward area of a deployment.

Arcs exist whenever officers represented by a supply node would be qualified to fill billets of a demand node. Thus the Georgia civil affairs officers are linked to the local government liason billets but not to artillery or intelligence needs. There may be several feasible assignments for officers of any source node. For example, Figure 10.19 indicates that the same civil affairs officers could serve as communications officers with assault force 1.

The Marines' first priority is to fill all needed billets, but there are always some left unfilled. The transportation problem of Figure 10.19 models the possibility of unfilled billets with an artificial supply node connected to all demands. High cost on its arcs penalizes unfilled billets in the objective function.

Once as many requirements as possible have been filled, a secondary concern is to minimize turbulence. That is, the Marines try to assign officers to the same unit to which they were assigned before mobilization, or at least to assign them to a nearby unit by minimizing total travel cost.

Using the following notation:

$s_i \triangleq$ supply of officers available at source node i

$d_j \triangleq$ demand for officers at demand node j

$c_{i,j} \triangleq$ distance officers at supply node i would have to travel to report for billet j (a large positive number if i is the artificial node)

$I_i \triangleq$ set of officer supply nodes i suitable for billets j

$J_j \triangleq$ set of billet demand nodes j suitable for officers at supply node i

$x_{i,j} \triangleq$ number of officers at node i mobilized to billets at j

this Marine mobilization problem reduces to the standard-form transportation problem

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_{j \in J_i} x_{i,j} = s_i \quad \text{for all } i \quad (\text{supply of } i) \\ & \sum_{i \in I_j} x_{i,j} = d_j \quad \text{for all } j \quad (\text{demand for } j) \\ & x_{i,j} \geq 0 \quad \text{for all } i, j \end{aligned}$$

Assignment Problems

Another important class of network flow models, known as (linear) assignment problems, do not seem at first glance to have anything to do with something flowing.

Definition 10.40 | **Assignment problems** deal with optimal pairing or matching of objects in two distinct sets.

We might pair jobs to machines, male dating service clients to female, duties to employees, and so on.

Assignment problems are modeled using (discrete) decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if } i \text{ is assigned to } j \\ 0 & \text{otherwise} \end{cases}$$

The first subscript refers to items in one set; the second subscript identifies items in the other.

Assignment problems may have quite complicated objective functions (see Section 11.4). We address here only the most common form with a linear objective. In that case we have known costs

$$c_{i,j} \triangleq \text{cost of assigning } i \text{ to } j$$

An optimal solution minimizes (or maximizes) total cost (or benefit).

Denoting by A the set of allowed assignments (i, j) , we can express the linear assignment model as bipartite flow problem

$$\begin{aligned} \text{min or max} \quad & \sum_{(i,j) \in A} c_{i,j} x_{i,j} && \text{(min or max total cost)} \\ \text{s.t.} \quad & \sum_{j \text{ with } (i,j) \in A} x_{i,j} = 1 && \text{for all } i \quad \text{(every } i \text{ is assigned)} \\ & \sum_{i \text{ with } (i,j) \in A} x_{i,j} = 1 && \text{for all } j \quad \text{(every } j \text{ is assigned)} \\ & x_{i,j} \geq 0 && \text{for all } (i, j) \in A \end{aligned} \tag{10.4}$$

The first system of constraints guarantees every i is assigned exactly once by summing over all possible assignments j . The second system does the same to assure that every j is assigned exactly once.

Notice that this formulation is an integer linear program (ILP). Decision variables are allowed only the discrete values 0 and 1. Still, we will soon see how the discreteness can be ignored in computing an optimum.

EXAMPLE 10.23: FORMULATING LINEAR ASSIGNMENT MODELS

The following table shows a computerized dating service’s compatibility ratings for male customer $i = 1, \dots, 3$ with and female customers $j = 1, \dots, 3$.

i	j		
	1	2	3
1	90	30	12
2	40	80	75
3	60	65	80

Formulate an assignment model to find the highest compatibility arrangement providing each customer with a single date of the opposite gender.

Solution: Using variables $x_{i,j} = 1$ if male i is paired with female j , the model corresponding to system (10.4) is

$$\begin{aligned}
 \max \quad & 90x_{1,1} + 30x_{1,2} + 12x_{1,3} + 40x_{2,1} + 80x_{2,2} + 75x_{2,3} + 60x_{3,1} + 65x_{3,2} + 80x_{3,3} \\
 \text{s.t.} \quad & x_{1,1} + x_{1,2} + x_{1,3} = 1 \\
 & x_{2,1} + x_{2,2} + x_{2,3} = 1 \\
 & x_{3,1} + x_{3,2} + x_{3,3} = 1 \\
 & x_{1,1} + x_{2,1} + x_{3,1} = 1 \\
 & x_{1,2} + x_{2,2} + x_{3,2} = 1 \\
 & x_{1,3} + x_{2,3} + x_{3,3} = 1 \\
 & \text{all } x_{i,j} = 0 \text{ or } 1
 \end{aligned}$$

APPLICATION 10.4: CAM ASSIGNMENT

For a more realistic assignment problem, consider a computer-aided manufacturing (CAM) system that automatically routes jobs through workstations of a computer-controlled factory. Each job consists of a sequence of required machining and assembly operations.

Often, there are several different workstations where the same operation can be performed. Thus the computer control system must make routing decisions. Each time that a job completes some operation, the system must select the next station to which the job should be routed from among the several that could perform the next operation required.

One method for accomplishing such control decisions in an approximately optimal way is to periodically solve an assignment model.³ To illustrate, suppose that the 8 fictitious jobs i of Table 10.2 are either waiting for movement to their next workstation or will finish their current operation within the next 5 minutes.

TABLE 10.2 Transportation and Processing Times for CAM Assignment Application

Jobs, i	Next Workstations, j									
	1	2	3	4	5	6	7	8	9	10
1	8	—	23	—	—	—	—	—	5	—
2	—	4	—	12	15	—	—	—	—	—
3	—	—	20	—	13	6	—	8	—	—
4	—	—	—	—	19	10	—	—	—	—
5	—	—	—	8	—	—	12	—	—	16
6	14	—	—	—	—	—	8	—	3	—
7	—	6	—	—	—	—	—	27	—	12
8	—	5	15	—	—	—	—	32	—	—

³Based on J. Chandra and J. Talavage (1991), *Optimization-Based Opportunistic Part Dispatching in Flexible Manufacturing Systems*, School of Industrial Engineering, Purdue University, July.

The 10 workstations j to which they might be routed are also shown. Entries in the table reflect

$$\begin{pmatrix} \text{transporation} \\ \text{time to the} \\ \text{station} \end{pmatrix} + \begin{pmatrix} \text{waiting time} \\ \text{until the station} \\ \text{becomes free} \end{pmatrix} + \begin{pmatrix} \text{operation} \\ \text{processing time} \\ \text{at the station} \end{pmatrix}$$

That is, they show the short-term time implications of assigning jobs i to stations j . Missing values in the table reflect assignments that are not possible because the next required operation cannot be performed at a workstation.

Balancing Unequal Sets with Dummy Elements

Letting $A \triangleq \{ \text{feasible } (i, j) \text{ pairs} \}$, and $c_{i,j} \triangleq$ the times shown in Table 10.2, an optimal short-term control decision is to assign jobs to workstations in a manner minimizing total time. This is exactly what assignment model (10.5) will compute.

One small complication arises from the fact that there are more workstations than jobs. Model (10.4) assumes that the sets to be matched have equal numbers of objects.

This problem is easily solved with dummy members.

Principle 10.41 If the two sets to be paired in an assignment problem differ in size, the smaller can be augmented with **dummy members**. These dummy objects should be treated as assignable to all members of the other set at zero cost.

Principle 10.41 leads to dummy jobs $i = 9, 10$ in our CAM application.

Integer Network Flow Solution of Assignment Problems

Any of the general-purpose network flow methods of this chapter (Algorithms 10A, 10B, and 10C), or even linear programming methods, can be used to solve assignment problems (after multiplying the first set of constraints in (10.5) by -1). Furthermore, we know by integrality property 10.35 that the optimum will be binary. All supplies and demands = 1, and no capacities apply, so that Assignment model (10.5) is a binary ILP that can be solved by continuous methods of linear programming. The next section presents a Primal-Dual LP algorithm for the Assignment Problem that computes a binary optimum even more efficiently.

CAM Assignment Application Model

We are now ready to state a full formulation of our CAM application in the standard format of (10.4):

$$\begin{aligned} \min \quad & 8x_{1,1} + 23x_{1,3} + 5x_{1,9} + 4x_{2,2} + 12x_{2,4} + 15x_{2,5} \\ & + 20x_{3,3} + 13x_{3,5} + 6x_{3,6} + 8x_{3,8} + 19x_{4,5} + 10x_{4,6} \\ & + 8x_{5,4} + 12x_{5,7} + 16x_{5,10} + 14x_{6,1} + 8x_{6,7} + 3x_{6,9} \\ & + 6x_{7,2} + 27x_{7,8} + 12x_{7,10} + 5x_{8,2} + 15x_{8,3} + 32x_{8,8} \end{aligned}$$

$$\begin{aligned}
\text{s.t. } & x_{1,1} + x_{1,3} + x_{1,9} &= 1 & \text{(job 1)} \\
& x_{2,2} + x_{2,4} + x_{2,5} &= 1 & \text{(job 2)} \\
& x_{3,3} + x_{3,5} + x_{3,6} + x_{3,8} &= 1 & \text{(job 3)} \\
& x_{4,5} + x_{4,6} &= 1 & \text{(job 4)} \\
& x_{5,4} + x_{5,7} + x_{5,10} &= 1 & \text{(job 5)} \\
& x_{6,1} + x_{6,7} + x_{6,9} &= 1 & \text{(job 6)} \\
& x_{7,2} + x_{7,8} + x_{7,10} &= 1 & \text{(job 7)} \\
& x_{8,2} + x_{8,3} + x_{8,8} &= 1 & \text{(job 8)} \\
& x_{9,1} + x_{9,2} + x_{9,3} + x_{9,4} + x_{9,5} \\
& + x_{9,6} + x_{9,7} + x_{9,8} + x_{9,9} + x_{9,10} &= 1 & \text{(job 9)} \\
& x_{10,1} + x_{10,2} + x_{10,3} + x_{10,4} + x_{10,5} \\
& + x_{10,6} + x_{10,7} + x_{10,8} + x_{10,9} + x_{10,10} &= 1 & \text{(job 10)} \\
& x_{1,1} + x_{6,1} + x_{9,1} + x_{10,1} &= 1 & \text{(station 1)} \\
& x_{2,2} + x_{7,2} + x_{8,2} + x_{9,2} + x_{10,2} &= 1 & \text{(station 2)} \\
& x_{1,3} + x_{3,3} + x_{8,3} + x_{9,3} + x_{10,3} &= 1 & \text{(station 3)} \\
& x_{2,4} + x_{5,4} + x_{9,4} + x_{10,4} &= 1 & \text{(station 4)} \\
& x_{2,5} + x_{3,5} + x_{4,5} + x_{9,5} + x_{10,5} &= 1 & \text{(station 5)} \\
& x_{3,6} + x_{4,6} + x_{9,6} + x_{10,6} &= 1 & \text{(station 6)} \\
& x_{5,7} + x_{6,7} + x_{9,7} + x_{10,7} &= 1 & \text{(station 7)} \\
& x_{3,8} + x_{7,8} + x_{8,8} + x_{9,8} + x_{10,8} &= 1 & \text{(station 8)} \\
& x_{1,9} + x_{6,9} + x_{9,9} + x_{10,9} &= 1 & \text{(station 9)} \\
& x_{5,10} + x_{7,10} + x_{9,10} + x_{10,10} &= 1 & \text{(station 10)} \\
& x_{i,j} = 0 \text{ or } 1 \text{ for all } i = 1, \dots, 10; j = 1, \dots, 10
\end{aligned} \tag{10.5}$$

An optimal short-term routing of jobs to workstations has

$$x_{1,1}^* = x_{2,2}^* = x_{3,8}^* = x_{4,6}^* = x_{5,4}^* = x_{6,9}^* = x_{7,10}^* = x_{8,3}^* = 1$$

and all other $x_{i,j}^* = 0$.

10.7 HUNGARIAN ALGORITHM FOR ASSIGNMENT PROBLEMS

The Assignment Problem introduced in Section 10.6 (definition [10.40](#)) can be solved by general network flow Algorithms 10A, 10B, and 10E, or by any of the Linear Programming methods of Chapters 5–7. In this section we introduce the highly efficient **Hungarian Algorithm** specifically tailored to assignment that predated most of those more general methods, but draws upon concepts from both networks and linear programming. Algorithm 10D provides a full statement with details explained in the coming subsections.

Primal-Dual Strategy and Initial Dual Solution

The first set of principles on which the Hungarian Algorithm is based on come from viewing the Assignment Problem as a Linear Program. Recall that the LP formulation of (linear) Assignment problems arises on a bipartite graph over source set I ,

**ALGORITHM 10D: HUNGARIAN ALGORITHM
FOR LINEAR ASSIGNMENT**

Step 0: Initialization. Choose starting dual solutions by setting $\bar{u}_i \leftarrow \max\{c_{i,j} : (i, j) \in A\}$ and $\bar{v}_j \leftarrow \max\{c_{i,j} - \bar{u}_i : (i, j) \in A\}$. Then construct the equality subgraph $A^=$ as the set of links with $\bar{c}_{ij} = w_{ij} - \bar{u}_i - \bar{v}_j = 0$. Set the initial solution set \bar{A} empty. Root a solution tree at all $i \in I$. Label all roots "even" and leave all other nodes unlabeled.

Step 1: Solution Growth. If $|\bar{A}| = |I|$, stop, the assignment detailed in \bar{A} is optimal. Otherwise attempt to grow \bar{A} by finding an even-to-unassigned, unlabeled $(i, j) \in A^=$. If one exists, identify the alternating path P from j back to its tree root. Reverse the assignment along the P by $\bar{A} \leftarrow \bar{A} \cup P \setminus (\bar{A} \cap P)$, and erase labeling on the tree containing P . Repeat Step 1.

Step 2: Tree Growth. Attempt to grow a tree by finding an even-to-unlabeled $(i, j) \in A^=$ with j assigned. If one exists, grow its tree by putting both (i, j) and the $(k, j) \in \bar{A}$ assigned to j into the tree, labeling node j "odd" and node k "even". Then return to Step 1.

Step 3: Dual Change. Define $D \leftarrow$ all $(i, j) \in A$ with i "even" and j unlabeled. If D is empty, stop, the given assignment instance is infeasible. Otherwise, choose a dual-change step size according to

$$\lambda \leftarrow \min\{|\bar{c}_{ij}| : (i, j) \in D\}$$

Then update each "even" $i \in I$ by

$$\bar{u}_i \leftarrow \bar{u}_i - \lambda$$

and each "odd" $j \in J$ by

$$\bar{v}_j \leftarrow \bar{v}_j + \lambda$$

Finally update the equality subgraph $A^=$ according to $A^= \leftarrow \{(i, j) \in A : \text{new } \bar{c}_{ij} = 0\}$, and return to Step 1.

sink set J , and links (i, j) in set A connecting nodes in I to possible assignments in J . For the maximizing case, it has the form

$$\begin{aligned} \max \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{(i,j) \in A} x_{ij} = 1 \text{ for all } i \in I \\ & \sum_{(i,j) \in A} x_{ij} = 1 \text{ for all } j \in J \\ & x_{ij} \geq 0 \text{ for all } (i, j) \in A \end{aligned} \tag{10.6}$$

where $c_{ij} \triangleq$ the cost/weight of assigning i to j , and decision variables $x_{ij} = 1$ if i is assigned to j , and $= 0$ otherwise. A binary solution is actually required, but we know from Section 10.5 and property [10.35](#) that an optimal solution to LP (10.6) will necessarily be integer.

Following the sequence of Primal-Dual LP Algorithm 6B, computation begins by constructing a feasible dual solution to formulation (10.6).

Principle 10.42 | Unrestricted dual variable values \bar{u}_i and \bar{v}_j on the equality constraints for points $i \in I$ and $j \in J$, respectively, are feasible in the dual of LP (10.6) if and only if reduced costs satisfy

$$\bar{c}_{ij} \triangleq c_{ij} - \bar{u}_i - \bar{v}_j \leq 0 \text{ for all } (i, j) \in A$$

The signs of reduced costs completely capture the requirements for dual feasibility.

The simple LP structure of assignment models also makes it very easy to find a starting dual feasible solution at Step 0 of Algorithm 10D.

Principle 10.43 | An initial dual feasible solution \bar{u}, \bar{v} corresponding to primal formulation (10.6) can be obtained by setting $\bar{u}_i = \max\{c_{ij} : (i, j) \in A\}$ and $\bar{v}_j = \max\{c_{ij} - \bar{u}_i : (i, j) \in A\}$

Choosing each \bar{u}_i as the largest of weights for its row i makes interim $\bar{c}_{ij} \leq 0$. Then choosing \bar{v}_j as the largest (least negative) of the interim values in column j keeps final $\bar{c}_{ij} \leq 0$ as required in [10.42].

We will illustrate these and other computations of Hungarian Algorithm 10D with the simple example instance detailed in Table 10.3. Objects in source set $I \triangleq \{1, 2, 3, 4\}$ are to be paired with those in sink set $J \triangleq \{5, 6, 7, 8\}$. Figure 10.20(a) shows the starting dual values from [10.43] and corresponding reduced costs. For example, dual value \bar{u}_1 gets the maximum of weights from node 1 or $\max\{9, 6, 4, 7\} = 9$. Next are the $\bar{v}_j \leftarrow \max\{c_{ij} - \bar{u}_i : (i, j) \in A\}$. For \bar{v}_3 this leads to $\max\{4 - 7, 0 - 1, -1 - 3, 2 - 5\} = -3$. Then each \bar{c}_{ij} is computed as $c_{ij} - \bar{u}_i - \bar{v}_j$.

Equality Subgraph

The next task in the primal-dual strategy of Algorithm 10D is to define a **restricted primal** over only variables x_{ij} allowed to be positive under complementary slackness with the current dual solution, and then search for a primal feasible solution within those variables alone. For these tasks, the Hungarian Algorithm 10D departs the context of LP, and addresses the Assignment Problem on the underlying bipartite graph.

Specifically, an **equality subgraph** defined over links with values allowed to be positive by complementary slackness plays the role of a restricted primal.

TABLE 10.3 Costs/Weights of Maximizing Numerical Application

c_{ij}	$j = 5$	$j = 6$	$j = 7$	$j = 8$
$i = 1$	9	6	4	7
$i = 2$	1	3	0	1
$i = 3$	6	5	-1	3
$i = 4$	2	3	2	5

Definition 10.44 The equality subgraph corresponding to any dual feasible assignment solution is the restricted primal over links in set

$$A^- \triangleq \{(i, j) \in A : \bar{c}_{ij} = 0\}$$

Figure 10.20(b) shows the A^- resulting from initial duals and reduced costs in part (a). The computations in [10.43] guarantees at least some links will have $\bar{c}_{ij} = 0$ and belong to A^- . Here there are many. Link (3, 5) is included with $\bar{c}_{35} = 0$, but (3,7) is not because $\bar{c}_{37} = -4$.

Labeling to Search for a Primal Solution in the Equality Subgraph

Algorithm 10D Steps 1 and 2 attempt to grow a primal feasible assignment within successive equality subgraphs by labeling trees. If a full primal solution is discovered, the algorithm stops optimal.

Principle 10.45 If a primal feasible assignment can be produced within the link set A^- of any equality subgraph corresponding to a feasible solution to the dual of model (10.6), that assignment is optimal in the full assignment model.

Karush-Kuhn-Tucker conditions [6.54] are fulfilled for model (10.6) and its dual because both primal and dual solutions are feasible, and limiting search to A^- assures complementary slackness.

If optimality has not yet been reached, the current tree labeling can be used to revise dual values at algorithm Step 3 while preserving dual feasibility. Updated duals yield a revised equality subgraph, and the search for a complementary primal solution continues.

Labeling begins by rooting a tree at each $i \in I$ not now assigned and designating it “even”. Figure 10.20(b) shows this leads to all $i \in I$ being labeled “even” since the initial set of assigned links \bar{A} is empty.

As labeling proceeds, Step 1 of Algorithm 10D seeks a link in A^- from an even-labeled root node i to an unlabeled, unassigned node j . If one is found, adding (i, j) to M leads to the working solution.

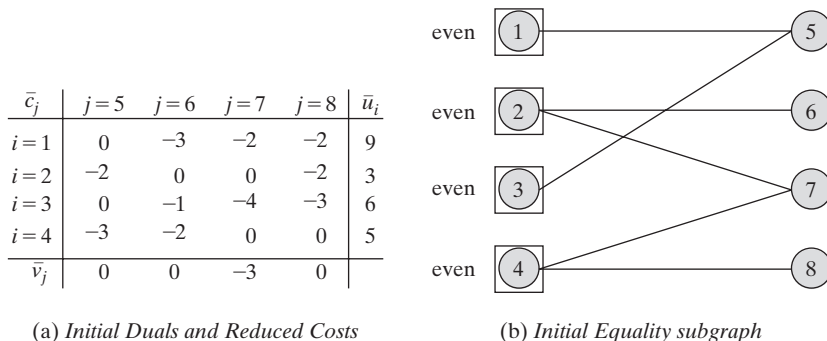


FIGURE 10.20 Initialization for Example Data of Table 10.3

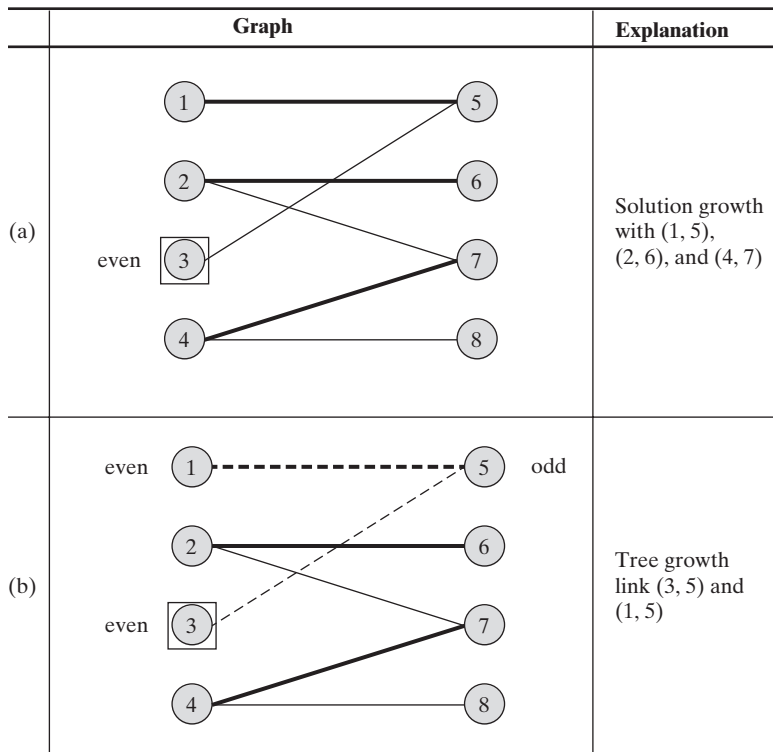
Figure 10.21(a) shows the result of three successive **solution growth** steps in the equality subgraph of Figure 10.20(b). First even-labeled root $i = 1$ is assigned to previously unlabeled, unassigned $j = 5$, and $(1, 5)$ joins \bar{A} . Then even-labeled root $i = 2$ is assigned to $j = 6$, and $i = 4$ is assigned to $j = 7$. All labels from those trees are erased and

$$\bar{A} = \{(1, 2), (2, 6), (4, 7)\}$$

Now only $i = 3$ remains to be assigned, and its only outgoing $A^=$ link leads to already assigned $j = 5$. The assignment cannot be grown immediately, but the tree rooted at $i = 3$ can be grown in the hope of eventually finding a more complex way to assign that node. This **tree growth** at Step 2 of Algorithm I0D adds links two at a time – one from an unassigned, even-labeled node to an assigned but unlabeled node j , and the other the link currently assigned to j . Node j is labeled “odd” to show it has been added to a tree, and the i to which it is now assigned is labeled “even.”

Part (b) of Figure 10.21 illustrates. Link $(3,5)$ is added to the tree even though node $j = 5$ is already assigned. Then node 5 is labeled “odd,” its current assignment $(1,3)$ is added to the tree, and node $i = 1$ becomes “even.”

Now further label progress is impossible. No even-labeled node i leads to any unlabeled node j . The dual solution must be updated to change the equality subgraph and permit further progress.



(bold links are assigned, dashed ones are included in a tree)

FIGURE 10.21 Labeling Progress on Initial Equality Subgraph

Dual Update and Revised Equality Subgraph

Dual change begins by identifying

$$D \leftarrow \{(i, j) \in A : i \text{ 'even' and } j \text{ unlabeled}\}$$

These are members of A not available in the current $A^=$ that would allow further solution or tree growth with Algorithm 10D Steps 1 or 2. If D is empty, no further progress is possible; the original assignment model was infeasible. Otherwise we update the dual solution as follows:

Principle 10.46 Set step size, $\lambda \leftarrow \min\{|\bar{c}_{i,j}| : (i, j) \in D\}$. Then update duals for even-labeled nodes $i \in I$ as $\bar{u}_i \leftarrow \bar{u}_i - \lambda$, and those for odd-labeled $j \in J$ by $\bar{v}_j \leftarrow \bar{v}_j + \lambda$. All other dual values remain on unchanged.

In the example of Figure 10.21(b), i nodes 1 and 3 are even-labeled, and within J , only node 5 is labeled. With A a complete graph, this labeling makes

$$D \leftarrow \{(1, 6), (1, 7), (1, 8), (3, 6), (3, 7), (3, 8)\} \text{ and}$$

$$\lambda \leftarrow \min\{|-3|, |-2|, |-2|, |-1|, |-4|, |-3|\} = 1$$

Figure 10.22 shows the full result of the update. Even-labeled duals \bar{u}_1 and \bar{u}_3 have decreased by $\lambda = 1$, and odd labeled \bar{v}_1 has increased by the same amount. Update of the reduced costs allows D-edge (3,6) to join the equality subgraph.

It is critical to realize that, in addition to retaining dual feasibility, the dual update of [10.46](#) preserves labeling progress already achieved, and opens the way to new labeling.

Principle 10.47 Dual change [10.46](#) adds at least one edge not in the previous equality subgraph that opens the way for either solution or tree growth. At the same time it retains all previous solution and tree edges and continues dual feasibility.

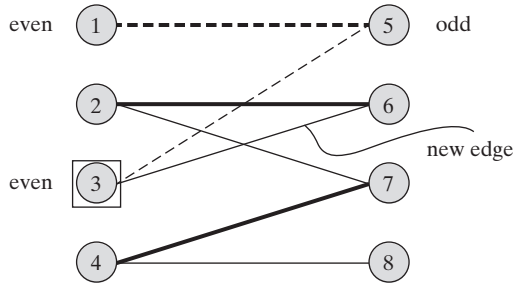
To verify these observations, enumerate the \bar{c}_{ij} impacts for different labelings:

Edge Status	\bar{u}_i change	\bar{v}_j change	Net \bar{c}_{ij}
even to odd	$-\lambda$	$+\lambda$	0
even to unlabeled	$-\lambda$	0	$+\lambda$
unlabeled to odd	0	$+\lambda$	$-\lambda$
unlabeled to unlabeled	0	0	0

Assigned and tree edges are either even-to-odd or unlabeled-to-unlabeled; both have reduced cost still = 0. Only even-to-unlabeled cases (in set D) increase \bar{c}_{ij} , and λ has been chosen to be sure at least one goes to = 0 but none become positive. Furthermore, even-to-unlabeled links are exactly what is needed to either grow the solution or grow the label trees.

	5	6	7	8	\bar{u}_i
1	0	-2	-1	-1	8
2	-3	0	0	-2	3
3	0	0	-3	-2	5
4	-4	-2	0	0	5
\bar{v}_j	1	0	-3	0	

(a) Updated Duals and \bar{c} matrix



(b) Updated Equality Subgraph

FIGURE 10.22 Results of Dual Update on Numerical Application

Solution Growth Along Alternating Paths

Figure 10.22 shows how progress continues in the new equality subgraph. New edge (3, 6) allows tree growth including assignment link (2, 6) that makes node $i = 2$ “even.” Then further tree growth adds edges (2, 7) and (7, 4) and labels $i = 4$ “even.” This opens the way to solution growth along an **alternating path** adding new assignment edge (4, 8).

$$P \leftarrow \{ (3, 6), (2, 6), (2, 7), (4, 7), (4, 8) \} \tag{10.7}$$

Principle 10.48 Labeling in Hungarian Algorithm 10D produces solution growth by highlighting an odd-cardinality alternating path in the equality subgraph starting from an unassigned root node in I , and continuing through nonassigned then assigned link pairs with tree growth, until a final link connects to an unassigned node in J . Swapping $\bar{A} \leftarrow \bar{A} \cup P \setminus (\bar{A} \cap P)$ results in an increase in the size of the current assignment by 1.

The path of (10.7) illustrates. Swapping its solution and nonsolution edges yields the optimal assignment of Figure 10.23 part (d).

Computational Order of the Hungarian Algorithm

To bound the computation (see Section 14.2) that could be required for Algorithm 10D in computing an optimal assignment or proving none exists, note that, unless infeasibility is detected first, constructing an assignment requires $O(|I|)$ rounds of solution growth. Each of these may involve $O(|I|)$ rounds of tree growth before all I -nodes have labels. Those, in turn, may be separated by $O(|I|)$ dual-changes. Then finally solution growth is completed by up to an $O(|I|)$ swap along an alternating path.

Principle 10.49 Hungarian Algorithm 10D runs in at most

$$O(|I|) \cdot (O(|I|^2) + O(|I|)) = O(|I|^3)$$

steps on an assistant problem with $|I|$ supplies and demands.

	Graph	Explanation
(c)		<p>Tree Growth (3, 6) and (2, 6) (2, 7) and (4, 4)</p>
(d)		<p>Solution growth $\bar{A} = I$, optimal solution found, stop</p>

(old links are assigned, dashed ones are in a tree)

FIGURE 10.23 Labeling Progress on Updated Equality Subgraph

10.8 MAXIMUM FLOWS AND MINIMUM CUTS

Maximum Flow and Minimum Cut problems are another simple special case of network flows. Still, initial work on the topics by Ford and Fulkerson (1956) is one of the founding achievements that led to all the tools and models reviewed in this chapter. This section provides a brief overview.

Definition 10.50 The **Max Flow** problem on a given directed graph $G(V, A)$ seeks simply to find the largest possible flow between a specified source node s and a specified sink node t , subject to conservation of flow at all other nodes and given arc capacities u_{ij} .

Minimum cut problems address the same input data, but focus on the complementary issue of identifying the arcs and capacities that most limit s -to- t flow. Specifically,

Definition 10.51 An **s-t cut set** (S, T) of directed graph $G(V, A)$ is a collection of arcs which, if deleted, partitions the graph into two non overlapping components – one on nodes $S \subset V$ including source node s and the other $T = V \setminus S$ containing sink node t . Then the **Min Cut** problem seeks an s-t cut set of minimum total capacity among “forward” arcs (i, j) with $i \in S$ and $j \in T$.

As usual, a simple application will clarify these definitions.

APPLICATION 10.5: BUILDING EVACUATION MAXIMUM FLOW

Maximum flow problems arise most often as subproblems in more complex operations research studies. However, they occur naturally in evaluating the safety of proposed building designs.⁴ Proper design requires adequate capacity for building evacuation in the event of an emergency.

Figure 10.24 shows a small example involving a proposed sports arena. Patrons in the arena would exit in an emergency through doors on all four sides that can accommodate 600 persons per minute. Those doors lead into an outer hallway that can move 350 persons per minute in each direction. Egress from the hallway is through four firestairs with capacity 400 persons per minute and a tunnel to the parking lot accommodating 800 persons per minute. Our interest is in the maximum rate of evacuation possible with this design.

Part (b) of Figure 10.24 shows how we reduce this safety analysis to a maximum flow model. Patron flows originate at source node 1. Outbound arcs model the four doorways. The flows around the outer hall lead to the four stairways and the tunnel. Persons exiting by any of those means pass to sink node 10. Capacities enforce the flow rates of the various facilities.

We wish to know the maximum flow from 1 to 10, subject to the capacities indicated. An optimal flow is provided in the arc labels of part (b). Patrons can escape at a total rate of 2100 per minute.

In application, the Min Cut analog of a Max Flow problem may have the greatest interest because it identifies bottlenecks that limit flow. Part (c) of Figure 10.24 depicts the minimum cut along with the optimal flow. Arcs of the cut partition the graph into two components over node subsets $S = \{1, 4, 5, 6, 7, 8\}$ and $T = \{2, 3, 10\}$ with source 1 in S and sink 10 in T . Forward arcs across this cut are $(1, 2)$, $(4, 3)$, $(8, 9)$, $(5, 10)$, and $(7, 10)$, all at capacity in the flow optimum. Summing their capacities gives cut capacity

$$600 + 350 + 350 + 400 + 400 = 2100$$

which exactly matches the maximum total flow. If designers wish to raise the evacuation rate of the design, one or more of those capacities will have to increase.

⁴Based in part on L. G. Chalmet, R. L. Frances, and P. B. Saunders (1982). “Network Models for Building Evacuation,” *Management Science*, 28, 86–105. All numerical data and diagrams were made up by the author of this book.

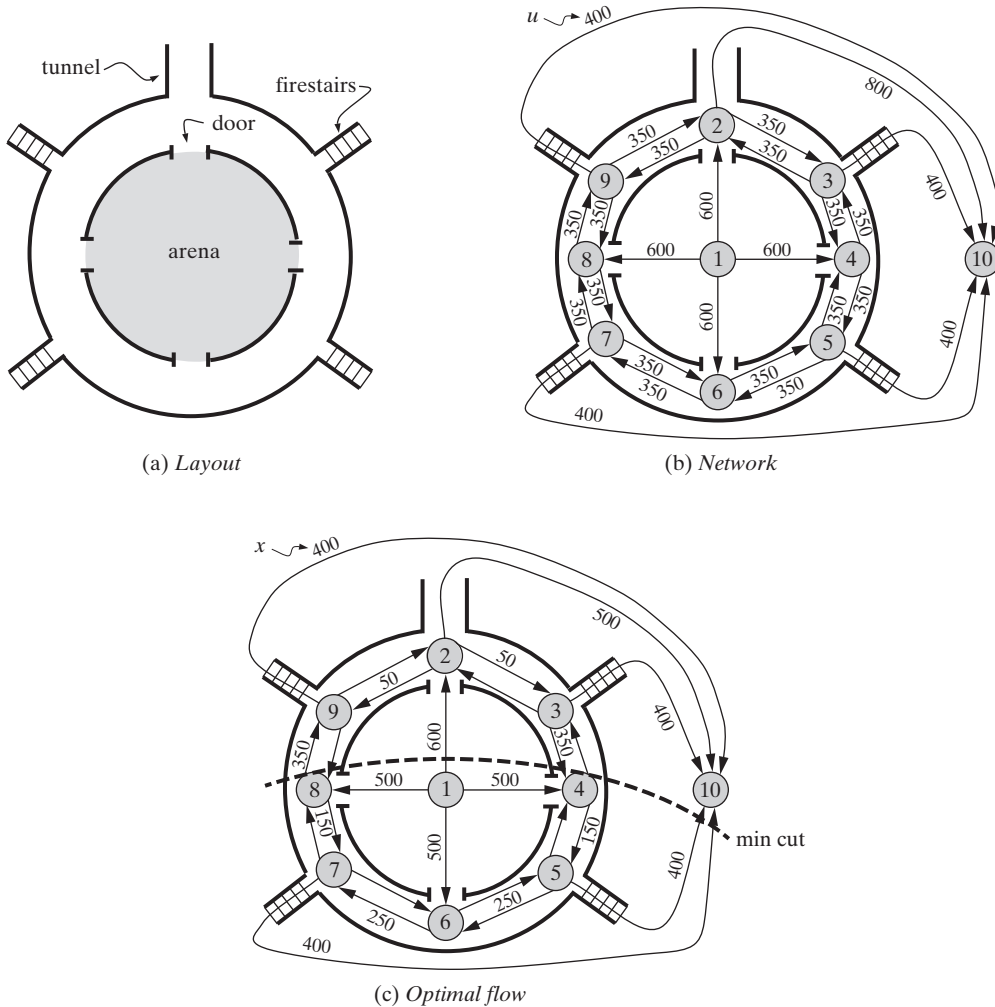


FIGURE 10.24 Building Evacuation Maximum Flow Application

Improving Feasible Cycle Directions and Flow Augmenting Paths

The key element of network flow Algorithms 10A, 10B, and 10C is improving feasible cycle directions used to better current feasible solutions. To discover the Max Flow analog, consider the example of Figure 10.25(a) with capacities shown on each (solid line) arc. One can imagine converting this max flow application to a standard network flow by (i) making costs on all existing arcs = 0, (ii) fixing all supplies and demands = 0, and (iii) adding the indicated single artificial return arc (6, 1) from sink node 6 to source node 1 with cost -1 and unlimited capacity. Then every improving and feasible cycle direction would include the return arc, which is the only opportunity for objective function improvement, together with a path from source to sink in the original graph that permits net flow increase. The latter becomes the target of an improving search algorithm for Max Flow/Min Cut.

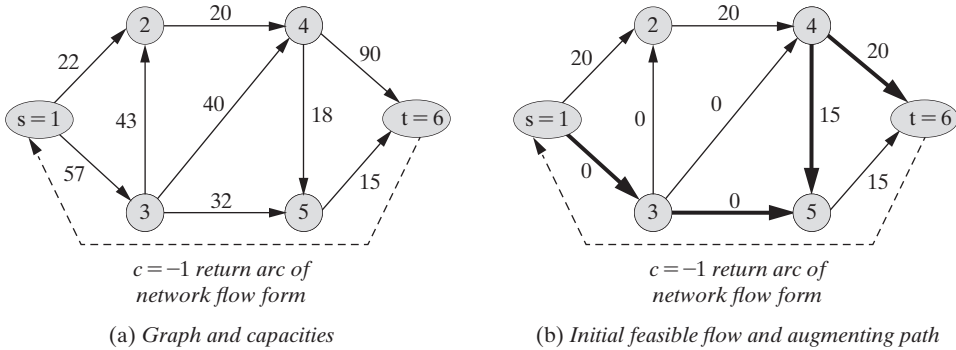


FIGURE 10.25 Numerical Application of Max Flow/Min Cut and Augmenting Paths

Definition 10.52 In a Max Flow problem on graph $G(V, A)$ with source node s , sink node t , arc capacities u_{ij} , and current feasible flow \bar{x}_{ij} , a **flow augmenting path** is a path from s to t with all forward arcs in the path having $\bar{x}_{ij} < u_{ij}$ and all reverse arcs having $\bar{x}_{ij} > 0$.

Figure 10.25(b) illustrates for the instance in part (a). Numbers on arcs are a starting feasible flow totalling 20 units from s to t . Heavier line arcs indicate a flow augmenting path 1–3–5–4–6 with all forward arcs below capacity, and the single reverse arc (4, 5) at positive flow 15. Notice that the imagined return arc would complete an improving feasible cycle direction in the corresponding min cost network flow model.

The Max Flow Min Cut Algorithm

As with Cycle Cancelling Algorithm 10B, a systematic way to find flow augmenting paths is to work with a **residual digraph** where attractive options are apparent.

Definition 10.53 The **max flow residual digraph** for graph $G(V, A)$ with capacities u_{ij} and current feasible flow \bar{x}_{ij} has the same nodes $i \in V$ as the given network. There is a forward/increase arc with capacity $u_{ij} - \bar{x}_{ij}$ for each arc (i, j) with $\bar{x}_{ij} < u_{ij}$, and a reverse/decrease arc with capacity \bar{x}_{ij} for each arc (i, j) with $\bar{x}_{ij} > 0$.

A directed $s - t$ path in this residual digraph corresponds to an augmenting path in the original network.

All the elements are now in place to detail a Max Flow Min Cut algorithm.

Solution of Max Flow Application of Figure 10.25(a) with Algorithm 10E

Figure 10.26 tracks application Algorithm 10E to the instance in Figure 10.25(a). Starting from the initial feasible solution Figure 10.25(b), computation for iteration $k = 1$ begins at Step 1 by constructing the corresponding residual digraph. Notice that arcs already at capacity like (2,4) have only a decrease options, those with

ALGORITHM 10E: MAXFLOW – MINCUT SEARCH

Step 0: Initialization. Choose any starting feasible flow $\bar{x}_{ij}^{(0)}$, and set solution index $k \leftarrow 1$.

Step 1: Residual Digraph. Construct the residual digraph corresponding to the current feasible flow, and attempt to find a directed path within it from s to t .

Step 2: Optimality and Minimum Cut. If no s - t directed path exists in the current residual digraph, stop; the current flow is maximum and the corresponding minimum cut (S, T) has

$$S \leftarrow \{i \in V \text{ reachable from source } s \text{ in the residual digraph}\}$$

$$T \leftarrow V \setminus S$$

Step 3: Step Size. Determine the flow augmenting path in the original graph corresponding to the identified directed path found in the residual digraph, and choose the maximum feasible augmentation step λ by

$$\lambda \leftarrow \min\{\lambda^+, \lambda^-\}$$

$$\lambda^+ \leftarrow \min\{(u_{ij} - \bar{x}_{ij}^{(k)}) : (i, j) \text{ is forward in the augmenting path}\}$$

$$\lambda^- \leftarrow \min\{\bar{x}_{ij}^{(k)} : (i, j) \text{ is reverse in the augmenting path}\}$$

Step 4: New Solution. Update the current flow solution by

$$\bar{x}_{ij}^{(k+1)} \leftarrow \begin{cases} \bar{x}_{ij}^{(k)} + \lambda & \text{if } (i, j) \text{ is forward in the augmenting path} \\ \bar{x}_{ij}^{(k)} - \lambda & \text{if } (i, j) \text{ is reverse in the augmenting path} \\ \bar{x}_{ij}^{(k)} & \text{otherwise} \end{cases}$$

Then advance $k \leftarrow k + 1$, and return to Step 1.

zero flow like (1, 3) have only an increase option, and others like (4, 5) have both. Examination of the residual digraph shows a directed path 1–3–5–4–6 (highlighted with heavy arcs) from s to t as an opportunity to increase flow. Step size λ is computed from the residual digraph capacities along the path as

$$\lambda^+ \leftarrow \min\{57, 32, 85\} = 32$$

$$\lambda^- \leftarrow \min\{15\} = 15$$

$$\lambda \leftarrow \min\{32, 15\} = 15$$

Increasing flow by 15 on corresponding forward arcs of the original graph, and decreasing by 15 on the one reverse arc, produces the updated flow shown for $k = 1$ which totals 35.

Now the process repeats with $k = 2$. The residual digraph shown yields directed path 1–3–4–6 and stepsize $\lambda = 40$. Then flows are updated to the values shown for $k = 2$ which total 75.

k	Residual Digraph	Updated Flow	Step
1			path 1-3-5-4-6 $\lambda^+ = 32$ $\lambda^- = 15$ $\lambda = 15$
2			path 1-3-4-6 $\lambda^+ = 40$ $\lambda^- = +\infty$ $\lambda = 40$
3		<p>Last flow is optimal because no directed path connects s to t in the residual digraph. Optimal cut set is (S, T) with $S \leftarrow \{1, 2, 3, 5\}$, $T \leftarrow \{4, 6\}$</p>	max flow = min cut = 75

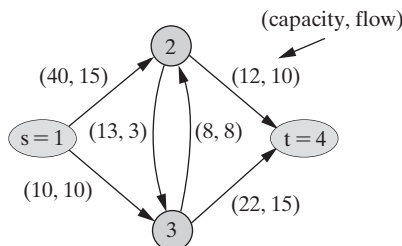
FIGURE 10.26 Progress of Algorithm 10E on Application of Figure 10.25(a).

At iteration $k = 3$, a residual digraph is again constructed. This time, however, it contains no directed s to t path. We can conclude that the last flow (from $k = 2$) is maximum with total value 75.

The residual digraph also defines a minimum cut by partitioning its nodes into those reachable from s and those not, that is, $S \leftarrow \{1, 2, 3, 5\}$ and $T \leftarrow V \setminus S = \{4, 6\}$. Forward capacity across this cut in Figure 10.25(a) totals $20 + 40 + 15 = 75$, exactly matching the maximum flow.

EXAMPLE 10.24: IDENTIFYING FLOWS, CUTS, AND AUGMENTING PATHS

Consider the max flow instance below with labels on arcs indicating (u_{ij}, \bar{x}_{ij}) .



- (a) Determine by inspection a maximum $s - t$ flow and a minimum cut.
- (b) Identify all the flow-augmenting paths available with the current flow, indicate whether member arcs are forward or reverse, and compute the maximum step λ that could be applied.
- (c) Construct the corresponding residual digraph, and note how each of the augmenting paths in part (b) are directed paths in the residual digraph.

Solution:

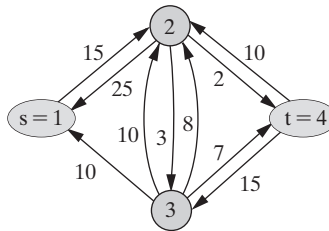
(a) A maximum flow would make $\bar{x}_{12} = 20$, $\bar{x}_{1,3} = 10$, $\bar{x}_{2,3} = 8$, $\bar{x}_{2,4} = 12$, $\bar{x}_{34} = 18$, for a total flow = 30. A minimum cut separates node $\{1, 2\}$ from $\{3, 4\}$ with total forward capacity of $10 + 8 + 22 = 30$.

(b) Available augmenting paths are

$$1 - 2 - 4 \quad \text{Both } (1, 2) \text{ and } (2, 4) \text{ fwd} \quad \lambda \leftarrow \min\{5, 2\} = 2$$

$$1 - 2 - 3 - 4 \quad (1, 2) \text{ fwd, } (3, 2) \text{ rev, } (3, 4) \text{ fwd} \quad \lambda \leftarrow \min\{5, 3, 7\} = 3$$

(c) The residual digraph is as follows:



Forward arcs are included for all opportunities to increase flows below capacity, and reverse arcs depict opportunities to decrease now-positive flows. This makes every augmenting path in the original graph a directed path in the residual digraph.

Equivalence of Max Flow and Min Cut Values

It is no accident that max flow and min cut capacity in the above example matched at optimality.

Principle 10.54 | Given a directed graph $G(V, A)$ with arc capacities u_{ij} , and specified source s and sink t , the maximum total feasible s -to- t flow equals to the minimum total forward capacity of cut sets separating s and t .

To see why this must be true, notice

$$\begin{aligned} \text{value of any feasible } s\text{-}t \text{ flow} &\leq \text{value of a maximum } s\text{-}t \text{ flow} \\ &\leq \text{capacity of a minimum } s\text{-}t \text{ cut} \\ &\leq \text{capacity of any } s\text{-}t \text{ cut} \end{aligned}$$

Every feasible unit of s -to- t flow must cross every s - t cut set, so the total flow is bounded by the forward capacity of every cut. It follows that if the value of any feasible s -to- t flow happens to equal the forward capacity of any s - t cut, both must be optimal in their respective problems.

This is exactly what occurs when Algorithm 10E terminates. The flow across the min cut it picks out must exactly equal its forward capacity because all forward arcs across the cut are at capacity, and all reverse arcs across the cut have zero flow. Otherwise, there would be a forward arc in the residual digraph, and there are none.

Computational Order of Algorithm 10E Effort

To bound the number of computational steps (see Section 14.2) or any instance of Max Flow - Min Cut on graph $G(V, A)$, observe first that each iteration k requires $O(|A|)$ computation in the worst case. All arcs may need to be considered in constructing the residual subgraph and choosing an s -to- t dipath within it. With a dipath in hand, $O(|V|)$ steps are needed to compute step-size λ and update flows in the original graph. With $O(|V|) \leq O(|A|)$ we conclude each iteration requires at most $O(|A|)$ time.

How many augmenting iterations can there be? Suppose we refine Algorithm 10E to seek augmenting paths of minimum cardinality at each iteration. This is easily done by breadth-first processing the residual digraph to find all nodes reachable in one step, then those reachable from already settled ones in two steps, and so on until sink t is encountered. Then the algorithm will first look for cardinality 1 paths until all are exhausted, then proceed to cardinality 2, and so on.

Notice that augmentation along such a minimum cardinality path will leave the residual digraph unchanged except at the arc blocking flow augmentation progress. Replacing that arc by its reverse, adds 2 to the cardinality of the same arc sequence from s to t , so the blocking arc can be used in a future augmentation only for paths of greater cardinality.

It follows that there can be at most $O(|A|)$ distinct updates per path cardinality. Furthermore, only $O(|V|)$ path cardinalities are possible. Combining with the effort per augmentation gives a computational bound.

Principle 10.55	On a graph $G(V, A)$ Max Flow - Min Cut Algorithm 10E implemented to choose minimum cardinality augmenting paths at each iteration requires at most $O(V A)$ augmentations and $O(V A ^2)$ total effort to compute a maximum flow and a minimum cut.
-----------------	---

10.9 MULTICOMMODITY AND GAIN/LOSS FLOWS

There is almost always a trade-off between the tractability of operations research models and the span of applications to which they can be adapted. Networks are no exception. In this section we explore briefly some more broadly applicable network forms that retain part, but not all, of the tractability of minimum cost network flows.

Multicommodity Flows

An implicit assumption in our previous flow models has been that all flows are in a **single commodity**. In the OOI application of Section 10.1, it was toaster ovens. In other applications the commodity was people, chemicals, or manufacturing tasks. In every case we assumed that demands at sink nodes could be filled from any supply node with a path to the sink. That is, we assumed that flows were interchangeable or fungible.

Multicommodity flow problems arise when flows passing thorough a common network must be kept separate; flow is not fungible.

Definition 10.56 **Multicommodity flow models** seek a minimum cost flow where separate commodities are moving through a common network.

APPLICATION 10.6: BAY FERRY MULTICOMMODITY FLOW

As usual, it will help to think of a simple (fictitious) application. Figure 10.27 depicts traffic flows in communities around an ocean bay. Each morning, the population of the three residential communities travels to the two industrial and two commercial centers in the region. Table 10.4 details flows by origin and destination. For example, 1250 of the 6000 daily trips originating at residential node 4 have industrial park node 7 as their destination.

At present, geography limits each trip to a single path. Numbers on arcs of the network in Figure 10.27 show the distance in kilometers between various points. For instance, those originating at node 1 and bound for node 7 must drive all the way around the bay through nodes 2 to 6. The 21,100 trips arising daily in the three residential communities produce a total of 399,250 kilometers of driving along such routes.

Regional planners are considering various improvements to reduce air pollution by reducing the number of kilometers driven. One idea is the ferry indicated by dashed arcs (2, 6) and (6, 2) in Figure 10.27. If a ferry were introduced, it could carry 2000 cars in each direction during the morning rush period. We want to know how many kilometers of driving might be saved.

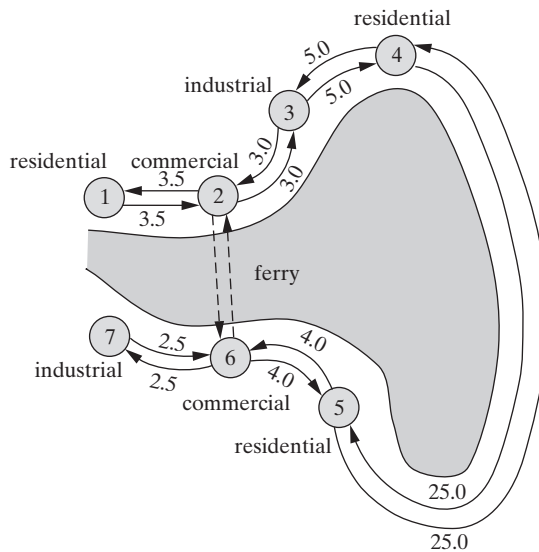


FIGURE 10.27 Bay Ferry Application Network

TABLE 10.4 Daily Trips in the Bay Ferry Application

Trip Origin Node	Total Trips	Trips by Destination						
		1	2	3	4	5	6	7
1	2850	—	900	750	40	10	600	550
4	6000	100	2000	1100	—	150	1400	1250
5	12250	110	4000	2200	200	—	3300	2440

Multicommodity Flow Models

Clearly, Figure 10.27 depicts a flow network. But what is flowing? If we treat all trips as equal, we have only a single commodity. Then, however, it would be feasible to fulfill the demand for 1250 trips from origin node 4 at sink node 7 with trips from any source. Naturally, a minimum distance solution would prefer ones from nearby source 5, leaving demands for node 5 trips at locations 2 and 3 to be filled from the closer source 1. Such a solution makes no sense for the application because trips are not fungible.

We must form separate commodity networks for trips from each of the three sources. That is, we must model in multiple commodities. Still, the commodities are not independent. All share the 2000-trip capacity of the proposed ferry. Such interdependencies are typical of multicommodity flows.

Principle 10.57 | Commodities of a multicommodity flow model cannot be analyzed separately because they interact through shared arc capacities.

Suppose that we employ constants

$c_{q,i,j} \triangleq$ unit cost of commodity q flow in arc (i, j)

$u_{i,j} \triangleq$ shared capacity of arc (i, j)

$b_{q,k} \triangleq$ net demand for commodity q at node k

and the decision variables

$x_{q,i,j} \triangleq$ commodity q flow in arc (i, j)

Definition 10.58 | The multicommodity network flow model on a digraph with nodes $k \in V$ and arcs $(i, j) \in A$ is

$$\begin{aligned} \min \quad & \sum_q \sum_{(i,j) \in A} c_{q,i,j} x_{q,i,j} \\ \text{s.t.} \quad & \sum_{(i,k) \in A} x_{q,i,k} - \sum_{(k,j) \in A} x_{q,k,j} = b_{q,k} \quad \text{for all } q, k \in V \\ & \sum_q x_{q,i,j} \leq u_{i,j} \quad \text{for all } (i, j) \in A \\ & x_{q,i,j} \geq 0 \quad \text{for all } q, (i, j) \in A \end{aligned}$$

Table 10.5 presents the corresponding formulation of our Bay Ferry application. There commodity 1 corresponds to flows from origin node 1, commodity 2 denotes flows from residential node 4, and commodity 3 relates to residential node 5. Notice that there are separate systems of flow conservation equations for each commodity, plus a common set of capacity constraints on the two arcs with flow limits.

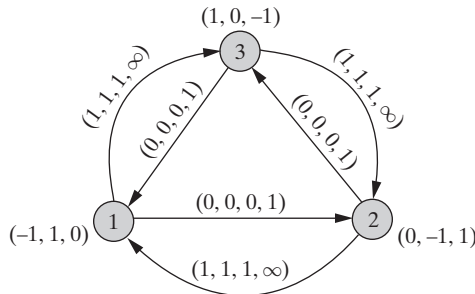
An optimal solution reduces the total driving to 280,770 kilometers, a savings of 29.7%. This is accomplished by accommodating $x_{1,2,6} = 1160$ commodity 1 trips on the 2-to-6 ferry, and $x_{2,2,6} = 840$ commodity 2. In the reverse direction the ferry carries $x_{3,6,2} = 2000$ commodity 3 trips.

TABLE 10.5 Bay Ferry Application Model

min	$3.5x_{1,1,2} + 3.5x_{1,2,1} + 3x_{1,2,3} + 3x_{1,3,2} + 5x_{1,3,4} + 5x_{1,4,3}$ $+ 15x_{1,4,5} + 15x_{1,5,4} + 4x_{1,5,6} + 4x_{1,6,5} + 2.5x_{1,6,7} + 2.5x_{1,7,6}$ $+ 3.5x_{2,1,2} + 3.5x_{2,2,1} + 3x_{2,2,3} + 3x_{2,3,2} + 5x_{2,3,4} + 5x_{2,4,3}$ $+ 15x_{2,4,5} + 15x_{2,5,4} + 4x_{2,5,6} + 4x_{2,6,5} + 2.5x_{2,6,7} + 2.5x_{2,7,6}$ $+ 3.5x_{3,1,2} + 3.5x_{3,2,1} + 3x_{3,2,3} + 3x_{3,3,2} + 5x_{3,3,4} + 5x_{3,4,3}$ $+ 15x_{3,4,5} + 15x_{3,5,4} + 4x_{3,5,6} + 4x_{3,6,5} + 2.5x_{3,6,7} + 2.5x_{3,7,6}$	(minimize driving)
s.t.	$x_{1,2,1} - x_{1,1,2} = -2,850$	(commodity 1)
	$x_{1,1,2} + x_{1,3,2} + x_{1,6,2} - x_{1,2,1} - x_{1,2,3} - x_{1,2,6} = 900$	
	$x_{1,2,3} + x_{1,4,3} - x_{1,3,2} - x_{1,3,4} = 750$	
	$x_{1,3,4} + x_{1,5,4} - x_{1,4,3} - x_{1,4,5} = 40$	
	$x_{1,4,5} + x_{1,6,5} - x_{1,5,4} - x_{1,5,6} = 10$	
	$x_{1,2,6} + x_{1,5,6} + x_{1,7,6} + x_{1,6,2} - x_{1,6,5} - x_{1,6,7} = 600$	
	$x_{1,6,7} - x_{1,7,6} = 550$	
	$x_{2,2,1} - x_{2,1,2} = 100$	(commodity 2)
	$x_{2,1,2} + x_{2,3,2} + x_{2,6,2} - x_{2,2,1} - x_{2,2,3} - x_{2,2,6} = 2,000$	
	$x_{2,2,3} + x_{2,4,3} - x_{2,3,2} - x_{2,3,4} = 1,100$	
	$x_{2,3,4} + x_{2,5,4} - x_{2,4,3} - x_{2,4,5} = -6,000$	
	$x_{2,4,5} + x_{2,6,5} - x_{2,5,4} - x_{2,5,6} = 150$	
	$x_{2,2,6} + x_{2,5,6} + x_{2,7,6} + x_{2,6,2} - x_{2,6,5} - x_{2,6,7} = 1,400$	
	$x_{2,6,7} - x_{2,7,6} = 1,250$	
	$x_{3,2,1} - x_{3,1,2} = 110$	(commodity 3)
	$x_{3,1,2} + x_{3,3,2} + x_{3,6,2} - x_{3,2,1} - x_{3,2,3} - x_{3,2,6} = 4,000$	
	$x_{3,2,3} + x_{3,4,3} - x_{3,3,2} - x_{3,3,4} = 2,200$	
	$x_{3,3,4} + x_{3,5,4} - x_{3,4,3} - x_{3,4,5} = 200$	
	$x_{3,4,5} + x_{3,6,5} - x_{3,5,4} - x_{3,5,6} = -12,250$	
	$x_{3,2,6} + x_{3,5,6} + x_{3,7,6} - x_{3,6,2} - x_{3,6,5} - x_{3,6,7} = 3,300$	
	$x_{3,6,7} - x_{3,7,6} = 2,440$	
	$x_{1,2,6} + x_{2,2,6} + x_{3,2,6} \leq 2,000$	(capacities)
	$x_{1,6,2} + x_{2,6,2} + x_{3,6,2} \leq 2,000$	
	all $x_{q,i,j} \geq 0$	

EXAMPLE 10.25: FORMULATING MULTICOMMODITY FLOWS

Consider the following multicommodity flow problem:



Labels on arcs show costs for three commodities and common capacity $(c_{1,i,j}, c_{2,i,j}, c_{3,i,j}, u_{i,j})$. Labels on nodes show net demands $(b_{1,k}, b_{2,k}, b_{3,k})$. Formulate the corresponding multicommodity network flow model.

Solution: Following format [10.58], the model is

$$\begin{aligned}
 \min \quad & x_{1,1,3} + x_{1,3,2} + x_{1,2,1} + x_{2,1,3} + x_{2,3,2} \\
 & + x_{2,2,1} + x_{3,1,3} + x_{3,3,2} + x_{3,2,1} \\
 \text{s.t.} \quad & x_{1,2,1} + x_{1,3,1} - x_{1,1,2} - x_{1,1,3} = -1 \\
 & x_{1,1,2} + x_{1,3,2} - x_{1,2,1} - x_{1,2,3} = 0 \\
 & x_{1,1,3} + x_{1,2,3} - x_{1,3,1} - x_{1,3,2} = 1 \\
 & x_{2,2,1} + x_{2,3,1} - x_{2,1,2} - x_{2,1,3} = 1 \\
 & x_{2,1,2} + x_{2,3,2} - x_{2,2,1} - x_{2,2,3} = -1 \\
 & x_{2,1,3} + x_{2,2,3} - x_{2,3,1} - x_{2,3,2} = 0 \\
 & x_{3,2,1} + x_{3,3,1} - x_{3,1,2} - x_{3,1,3} = 0 \\
 & x_{3,1,2} + x_{3,3,2} - x_{3,2,1} - x_{3,2,3} = 1 \\
 & x_{3,1,3} + x_{3,2,3} - x_{3,3,1} - x_{3,3,2} = -1 \\
 & x_{1,1,2} + x_{2,1,2} + x_{3,1,2} \leq 1 \\
 & x_{1,2,3} + x_{2,2,3} + x_{3,2,3} \leq 1 \\
 & x_{1,3,1} + x_{2,3,1} + x_{3,3,1} \leq 1 \\
 & \text{all } x_{q,i,j} \geq 0
 \end{aligned}$$

Tractability of Multicommodity Flow Models

Multicommodity flow formulations [10.58] are certainly linear programs. In fact, they are rather special linear programs for which unusually efficient algorithms can be developed. Also, we continue to think in terms of flows and represent complex models in simple diagrams.

Still, very little of the elegant structure presented for single-commodity cases in Sections 10.2 to 10.5 carries over to the multicommodity setting. Most important is a loss of integrality property [10.35].

Principle 10.59 | Even if all problem data are integer, optimal solutions to multicommodity flow problems may be fractional.

The instance of Example 10.25 illustrates. Each commodity is supplied at one node and demanded at another. Since costs on the inner dicycle 1–2–3–1 are zero, all commodities compete for the unit capacities on those arcs. It is easy to verify that the unique optimal solution makes

$$\begin{aligned}
 x_{1,1,2} = x_{1,2,3} = x_{1,1,3} &= \frac{1}{2} \\
 x_{2,2,3} = x_{2,3,1} = x_{2,2,1} &= \frac{1}{2} \\
 x_{3,3,1} = x_{3,1,2} = x_{3,3,2} &= \frac{1}{2}
 \end{aligned}$$

Total cost of this fractional solution is $\frac{3}{2}$, versus 2 for any all-integer flow.

Flows with Gains and Losses

Another implicit assumption of network models so far encountered is that the number of units entering one end of any arc is the same as the number of units leaving the other end. Many realistic modeling situations fail this assumption. Power distribution networks lose electricity along distribution lines, groundwater seeps into and out of sewers along their runs, and invested funds earn interest as they flow through time. All such circumstances produce **flows with gains and losses**.

Definition 10.60 Network flow problems with gains and losses model circumstances where a given constant $a_{i,j}$ units of flow exit arc (i, j) at node j for every 1 unit of flow entering at i . Value $a_{i,j} > 1$ indicates a gain, $a_{i,j} < 1$ implies a loss, and $a_{i,j} = 1$ yields ordinary network flows.

APPLICATION 10.7: TINYCO CASH FLOW WITH GAINS AND LOSSES

One of the most common settings for flows with gains and losses is financial transactions. A simple case is cash flow modeling.⁵

Cash flow management deals with cash and near equivalents such as short-term bonds. The object is to have cash available when it is needed to pay obligations while earning as much interest as possible on funds that are not needed immediately. In some cases it is also possible to borrow cash against future receipts.

Figure 10.28 presents a specific instance for a fictitious company we will call Tinyco. Nodes 1 to 5 represent cash over time, with numbers b_k next to each node showing net demand for cash by month (in thousands of dollars). Nodes 6 and 7 model funds invested in short-term bonds, starting from an initial holding of \$200,000.

We assume that cash can be invested to return 0.5% per month, and bonds 0.9%. This produces the gain multipliers $a_{i,i+1}$ shown in boxes on arcs in Figure 10.26 running forward in time. For example, arc $(3, 4)$ carries multiplier $a_{3,4} = 1.005$ because each dollar invested at month 3 yields 1.005 dollars after a month's interest at 0.5%.

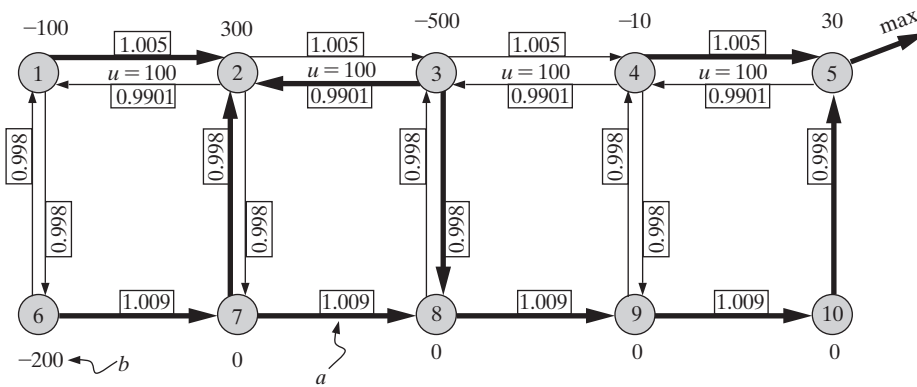


FIGURE 10.28 Tinyco Application Network

⁵Based on B. Golden and M. Liberatore (1979), “Models and Solution Techniques for Cash Flow Management,” *Computers and Operations Research*, 6, 13–20.

Reverse arcs along the cash part of Figure 10.28 illustrate loss arcs by modeling borrowing. We assume that our company can obtain up to \$100,000 in cash at 1% per month. Thus each dollar of next month's cash borrowed to meet current needs yields

$$\frac{1}{1.01} \approx 0.9901$$

dollars now.

Similar loss arcs connect cash and bond nodes in Figure 10.28. For example, arc (2, 7) represents cash invested in bonds during week 2. The loss multiplier $a_{2,7} = 0.998$ corresponds to a 0.2% investment fee. The similar loss on arc (7, 2) models an identical fee to convert bonds into cash.

The object of the management problem depicted in Figure 10.28 is to have as much money as possible at the end of the planning period. Thus the only nonzero objective function coefficient occurs on node 5 outflow.

Gain and Loss Network Flow Models

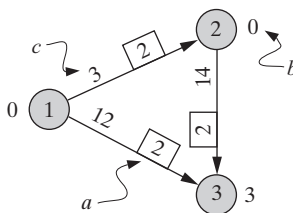
To reduce gain/loss flow application such as cash management problems to standard network flow form, we have only to include multipliers $a_{i,j}$ in the minimum cost network flow format of [10.3](#).

<p>Definition 10.61 The gain/loss flow problem on a digraph with nodes in V having net demand b_k, and arcs $(i, j) \in A$ having capacity $u_{i,j}$ and multiplier $a_{i,j}$ is</p>
$\min \sum_{(i,j) \in A} c_{i,j} x_{i,j}$
$\text{s.t.} \quad \sum_{(i,k) \in A} a_{i,k} x_{i,k} - \sum_{(k,j) \in A} x_{k,j} = b_k \quad \text{for all } k \in V$
$0 \leq x_{i,j} \leq u_{i,j} \quad \text{for all } (i,j) \in A$

Table 10.6 illustrates for our cash flow application. An optimal solution returns \$489,311 at the end of the planning period. Nonzero flows are on arcs highlighted in Figure 10.28.

EXAMPLE 10.26: FORMULATING FLOWS WITH GAINS

Taking numbers next to nodes as net demands b_k , quantities in boxes on arcs as gain multipliers $a_{i,j}$, and unboxed numbers on arcs as costs $c_{i,j}$, formulate the flow with gains model corresponding to the following figure:



Solution: Following format 10.61, the model is

$$\begin{aligned}
 \min \quad & 3x_{1,2} + 14x_{2,3} + 12x_{3,1} \\
 \text{s.t.} \quad & 2x_{3,1} - x_{1,2} = 0 \\
 & 2x_{1,2} - x_{2,3} = 0 \\
 & 2x_{2,3} - x_{3,1} = 3 \\
 & x_{1,2}, x_{2,3}, x_{3,1} \geq 0
 \end{aligned}$$

TABLE 10.6 Tynco Application Model

$\max r$		(max return)
$\text{s.t. } 0.9901x_{2,1} + 0.9980x_{6,1} - x_{1,2} - x_{1,6}$	$= -100$	(note 1)
$1.005x_{1,2} + 0.9901x_{3,2} + 0.9980x_{7,2} - x_{2,1} - x_{2,3} - x_{2,7}$	$= 300$	(node 2)
$1.005x_{1,3} + 0.9901x_{4,3} + 0.9980x_{7,3} - x_{3,2} - x_{3,4} - x_{3,8}$	$= -500$	(node 3)
$1.005x_{3,4} + 0.9901x_{5,4} + 0.9980x_{9,4} - x_{4,3} - x_{4,5} - x_{4,9}$	$= -10$	(node 4)
$1.005x_{4,5} + 0.9980x_{10,5} - x_{5,4} - r$	$= 30$	(node 5)
$0.9980x_{1,6} - x_{6,1} - x_{6,7}$	$= -200$	(node 6)
$0.9980x_{2,7} + 1.009x_{6,7} - x_{7,2} - x_{7,8}$	$= 0$	(node 7)
$0.9980x_{3,8} + 1.009x_{7,8} - x_{8,9} - x_{8,9}$	$= 0$	(node 8)
$0.9980x_{4,9} + 1.009x_{8,9} - x_{9,4} - x_{9,10}$	$= 0$	(node 9)
$1.009x_{9,10} - x_{10,5}$	$= 0$	(node 10)
$x_{2,1} \leq 100$		(cash limit)
$x_{3,2} \leq 100$		
$x_{4,3} \leq 100$		
$x_{5,4} \leq 100$		
$\text{all } x_{i,j} \geq 0$		

Tractability of Network Flows with Gains and Losses

As with multicommodity flows, models with gain and loss flows are linear programs with some exploitable structure. It is still convenient to think in terms of flows and to represent models in simple diagrams.

Still, many elegant properties of ordinary flows are lost when gains and losses are permitted. In particular, integrality property 10.35 does not extend.

Principle 10.62 | Even if all problem data are integer, optimal solutions to flow problems with gains and losses may be fractional.

The simple instance in Example 10.26 provides an example. All data are integer, but the only feasible solution is

$$x_{12} = \frac{6}{7}, \quad x_{2,3} = \frac{12}{7}, \quad x_{3,1} = \frac{3}{7}$$

10.10 MIN/MAX SPANNING TREES

This section presents another special network/graph model with an elegant solution: the **min/max spanning tree** problem. As usual, we begin with a fictional application.

APPLICATION 10.8: WILDERNESS ENERGY (WE)

Wilderness Energy (WE), a natural gas drilling company, wants to build roads to facilitate travel to/from and among drilling sites in a wilderness area it controls. Figure 10.29 shows the area of interest (shaded), and the seven sites that have been selected by air and satellite analysis. It also shows possible road alignments between pairs of sites that have been identified, including the estimated cost to construct each (in \$ thousand). Only site 1 will link the others to the outside world. WE wishes to choose a minimum total cost collection of the identified road options among the sites to produce a network including a path between every pair of sites. An optimal solution to this problem is shown in bold with total cost \$80,000.

Minimum/Maximum Spanning Trees and the Greedy Algorithm

What Wilderness Energy needs is a subgraph like the bold edges of the Figure 10.29 that connects all sites without extra road links which add cost.

Recall from Section 10.7 that a **spanning tree** of a given graph is a connected subgraph containing no cycles and spanning all nodes. A **min/max spanning tree** is one of minimum, respectively maximum total weight. WE Application 10.8 seeks a min spanning tree of the road network depicted in Figure 10.29.

Although it may seem challenging to compute optimal spanning trees, there is a remarkably simple **greedy algorithm** that can find either a minimum or a maximum total weight solution with comparatively little effort. Algorithm 10F provides the details.

Solution of the WE Application 10.8 by Greedy Algorithm 10F

Greedy search starts by sorting all the edges in Figure 10.29 in non decreasing construction cost from (3, 6) with cost \$2 thousand through (2, 3) at \$41 thousand, and initializing solution set $T \leftarrow \emptyset$.

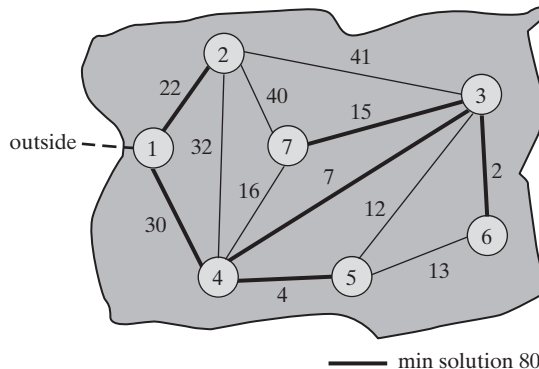


FIGURE 10.29 Wilderness Energy Drill Sites and Possible Roads

ALGORITHM 10F: GREEDY SEARCH FOR A MIN/MAX SPANNING TREE

Step 0: Initialization. Given undirected graph $G(V, E)$, create a list of edges in set E sorted in non-decreasing cost sequence for a minimize problem (non-increasing for a maximize), and initialize solution set $T \leftarrow \emptyset$.

Step 1: Edge Processing. If $|T| = |V| - 1$, stop, T defines an optimal spanning tree. Otherwise, consider the next sequential edge e in the sorted list. If e creates a cycle with the edges already in solution set T , skip it. If not, update $T \leftarrow T \cup e$. Either way repeat step 1.

To begin, the shortest edge $(3, 6)$ is placed in T . Next-shortest edges $(4, 5)$ and $(3, 4)$ can also be added, in turn, without creating a cycle, to produce $T = \{(3, 6), (4, 5), (3, 4)\}$. The next edge in the sort order is $(3, 5)$, but adding it to solution set T would create a cycle $3-4-5$. Thus it is skipped. Similarly, edges $(3, 5)$ and $(4, 7)$ are skipped because each creates a cycle with T . Continuing with edges $(3, 7)$, $(1, 2)$, and $(1, 4)$, none of which creates a cycle, leaves solution set

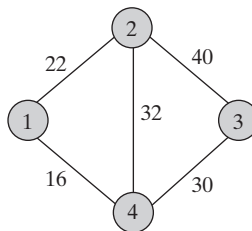
$$T^* = \{(3, 6), (4, 5), (3, 4), (3, 7), (1, 2), (1, 4)\}$$

With $|T^*| = |V| - 1 = 6$, which is the greatest number of edges possible without creating cycles. Algorithm 10F terminates with an optimal spanning tree.

The name “greedy” for Algorithm 10F comes from the fact that it selects the best-cost edge immediately available at each step without considering consequences for later choices. Only edges that would immediately create cycles are skipped because they cannot help the current solution. If any skipped edge were added, we would have to drop another already chosen edge of the cycle it creates. Since all the edges in such a cycle have costs no worse than the skipped one because they were chosen before it, adding the skipped edge cannot benefit the solution.

EXAMPLE 10.26: GREEDY COMPUTATION OF A MAX SPANNING TREE

Consider the following graph with numbers on edges showing weights.



Apply Algorithm 10F to compute a maximum weight spanning tree.

Solution: Taking edges highest weight first, edges $(2, 3)$ and $(2, 4)$ would be selected. Next edge $(3, 4)$ creates a cycle, so processing moves to $(1, 2)$. This completes the optimal spanning tree $T^* = \{(1, 2), (2, 3), (2, 4)\}$.

Representing Greedy Results in a Composition Tree

To more rigorously justify why the greedy algorithm produces an optimal tree and detail an implementation, it is useful to track the evolution of the solution in a node-set composition tree.

Definition 10.63 The **composition tree** corresponding to an Algorithm 10F greedy search represents how, starting with each node in its own singleton set, selection of new tree edges joins nodes of the graph into bigger and bigger connected components.

Figure 10.30 illustrates for the WE application of Figure 10.29. Each node forms a subset of its own at the bottom of the tree. Larger collections of nodes are created by joining those connected by the newly selected edge at each step. Edge (3, 6) is selected first, and its corresponding end nodes are spanned to form subset $S_1 = \{3, 6\}$. The next cheapest edge (4, 5) joins subsets $\{4\}$ and $\{5\}$ to create a $S_2 = \{4, 5\}$. Then, edge (3, 4) connects subsets S_1 and S_2 to form $S_3 = \{3, 4, 5, 6\}$ for the graph component spanning those 4 nodes. The next cheapest edge (3, 5) with cost 12 was skipped because it has both ends in the same, already-spanned subset $\{3, 4, 5, 6\}$; it forms a cycle with edges already there. Edge (5, 6) is skipped for the same reason, but edge (4, 7) can connect node 7 with the others to produce $S_4 = \{3, 4, 5, 6, 7\}$. In a similar way, nodes 1 and 2 are first joined by greedy edge (1, 2) in S_5 , then combined with the rest of the nodes by edge (1, 4). The result is full vertex set $V \triangleq S_6 = \{1, 2, 3, 4, 5, 6, 7\}$.

ILP Formulation of the Spanning Tree Problem

To build a formal justification for Algorithm 10F on undirected graph $G(V, E)$, we will formulate it as an integer linear problem over edge cost c_{ij} and edge decision variables $x_{ij} = 1$ if edge (i, j) is in the solution and $= 0$ otherwise.

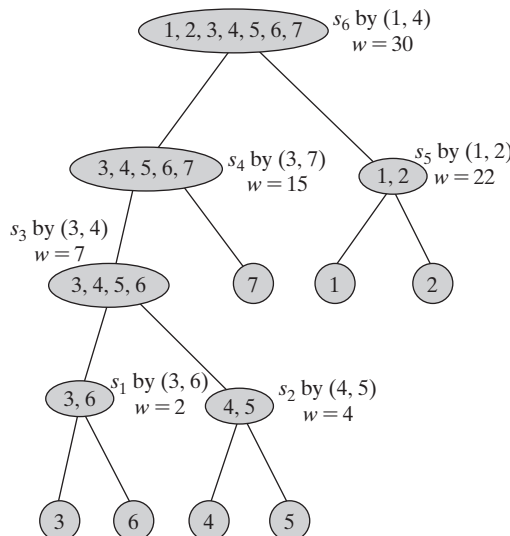


FIGURE 10.30 Composition Tree for WE Application 10.8

Principle 10.64 The minimum (or maximum) spanning tree problem on graph $G(V, E)$ can be formulated as the ILP

$$\begin{aligned}
 & \mathbf{min} \text{ (or max)} && \sum_{(i,j) \in E} c_{ij} x_{ij} \\
 \text{s.t.} &&& \sum_{(i,j) \in E} x_{ij} = |V| - \mathbf{1} \\
 &&& \sum_{(i,j) \in S} x_{ij} \leq |S| - 1 \text{ for all } S \subset V, |S| > 1 \\
 &&& x_{ij} = 0 \text{ or } 1 \text{ for all } (i, j) \in E
 \end{aligned}$$

The first constraint of the formulation in [10.64] requires that exactly $|V| - 1$ edges are selected as required for a spanning tree. All inequality constraints prevent cycles by limiting the selected edges joining elements of any node set S to number no more than $|S| - 1$. Clearly all these are satisfied by every spanning tree solution. Still, since there is one such constraint for every proper node subset of cardinality > 1 , the full constraint list becomes exponential.

To see how the greedy algorithm implicitly yields an LP relaxation optimum also optimal in the ILP without explicitly treating all such constraints, we also need its LP relaxation dual.

Principle 10.65 Using dual variables u_S for the main constraints in the LP relaxations of primal spanning tree formulation [10.64], corresponding dual formulations of the minimize and maximize cases are respectively

$$\begin{aligned}
 & \mathbf{max} && \sum_{S \subseteq V, |S| > 1} (|S| - 1) u_S \\
 \text{s.t.} &&& \sum_{S \supset (i,j)} u_S \leq c_{ij} \text{ for all } (i, j) \in E \\
 &&& u_S \leq 0 \text{ for all } S \subset V, u_v \text{ URS}
 \end{aligned}$$

and

$$\begin{aligned}
 & \mathbf{min} && \sum_{S \subseteq V, |S| > 1} (|S| - 1) u_S \\
 \text{s.t.} &&& \sum_{S \supset (i,j)} u_S \geq c_{ij} \text{ for all } (i, j) \in E \\
 &&& u_S \geq 0 \text{ for all } S \subset V, u_v \text{ URS}
 \end{aligned}$$

The optimal greedy solution from Algorithm 10F provides a primal feasible solution to both the ILP of [10.64] and its LP relaxation.

Principle 10.66 Using the solution constructed by greedy Algorithm 10F, a primal feasible solution to both ILP [10.64] and its LP relaxation are obtained by setting $\bar{x}_j = 1$ on greedy-chosen edges and $\bar{x}_j = 0$ otherwise.

Exactly $|V| - 1$ edges are selected, and the absence of cycles assures satisfaction of all the subset cardinality inequalities.

The composition tree for the search shows how to construct a corresponding feasible solution to LP relaxation dual [10.65].

Principle 10.67 | A dual feasible solution for any given instance of the LP relaxation of [10.64] can be constructed by setting the all $\bar{u}_S \leftarrow 0$ except on subsets S_k in the corresponding composition tree with $|S_k| > 1$. Then, $\bar{u}_V \leftarrow$ the cost of the last edge selected. Other non-singleton tree sets S_k get, $\bar{u}_{S_k} \leftarrow$ (cost of the edge creating subset S_k) (cost of the edge creating its tree-parent).

We can illustrate the construction of [10.67] by computing the dual solution implied in the composition tree of WE application Figure 10.30. First, $\bar{u}_{S_1} \leftarrow 2 - 7 = -5$, because the edge creating S_1 has weight $c_{3,6} = 2$ and its tree-parent S_4 was created by edge $(3, 4)$ with weight $c_{3,4} = 7$. Continuing, $\bar{u}_{S_2} \leftarrow 4 - 7 = -3$, $\bar{u}_{S_3} \leftarrow 7 - 15 = -8$, $\bar{u}_{S_4} \leftarrow 15 - 30 = -15$, and $\bar{u}_{S_5} \leftarrow 22 - 30 = -8$. With no parent, $\bar{u}_V \triangleq \bar{u}_{S_6}$ get 3, 0, the weight of the last chosen edge. Summing over nonzero duals, the objective value is then $-5 * 1 - 3 * 1 - 8 * 3 - 15 * 4 - 8 * 1 + 30 * 6 = 80$ which matches the primal solution value. This is no accident.

Principle 10.68 | After application of Algorithm 10F, the primal solution of [10.66] and the dual solution of [10.67] are both feasible in their respective problems and share the same objective function value. Thus both are optimal. Furthermore, since the relaxation primal produces an integer optimum, that solution is also optimal in the full ILP [10.64].

First, to see why the dual computation in [10.67] always produces agreement of primal and dual solution values, note that the creation of each new set S_k in the composition tree by new edge (i, j) adds $|S_k| - 1$ copies of c_{ij} to the dual sum. But its children, say S_i and S_j , have already subtracted all but one of these because $|S| = |S_i| + |S_j|$. This leaves the running dual sum simply the total of costs on edges in the greedy solution, which is also the primal value.

To see that the constructed dual solution is also feasible, focus on the min case. All constructed dual values \bar{u}_{S_k} on $S_k \subset V$ will be non positive as required for the corresponding primal \leq constraints. Ones not in the composition tree are fixed = 0, and those in the tree, subtract parent edge costs at least as large as those of their children.

Main dual inequalities require $c_{ij} \geq$ the sum of \bar{u}_S along the path from the first tree set to which i and j both belong, call it (k, l) , up to the root for V . That sum begins with the c_{kl} , then both subtracts and add the costs of selected edges higher in the composition tree, so that the total is first set cost c_{kl} . Now consider three cases. For any edge (i, j) creating a set in the composition and taking part in the primal optimum, its set is exactly the first time i and j have shared a component. Thus, the inequality for (i, j) is satisfied as equality conforming to complementary slackness with $\bar{x}_{ij} > 0$. For an edge (i, j) not selected by the greedy algorithm, but still having both ends in some composition set, the cost selected there, say c_{kl} , will have $c_{kl} \leq c_{ij}$ as required for dual feasibility because (k, l) was chosen over (i, j) by the algorithm. Finally, if no set in the composition tree except the last contains both ends of edge (i, j) , all included variables in its dual constraint will have value = 0, and again (with $c_{ij} \geq 0$), dual feasibility holds.

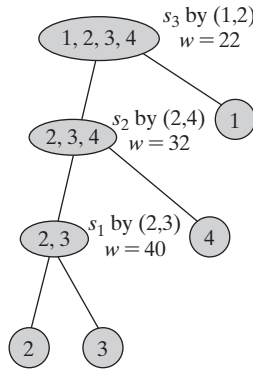
EXAMPLE 10.27: CONSTRUCTING COMPOSITION TREE AND DUAL SOLUTION

Return to the max spanning tree of Example 10.26.

- (a) Construct the composition tree corresponding to its greedy solution.
- (b) Construct the associated primal solution to the corresponding ILP [10.64].
- (c) Construct the corresponding LP relaxation dual solution of [10.67], and verify its solution value matches the primal.
- (d) Verify that your dual solution is dual feasible.

Solution:

(a) The composition tree is shown below. Nodes 2 and 3, then 4, and finally 1 are joined by greedy-chosen edges into graph components defined by node sets.



(b) The primal optimum has greedy-chosen edge values $\bar{x}_{1,2} = \bar{x}_{2,3} = \bar{x}_{2,4} = 1$, and all others = 0. This yields solution value $40 + 32 + 22 = 94$.

(c) Dual variable values will = 0 on all subsets except those in the composition tree. For those, we have $\bar{u}_{S_1} \leftarrow 40 - 32 = 8$, $\bar{u}_{S_2} \leftarrow 32 - 22 = 10$, and $\bar{u}_V = \bar{u}_{S_3} = 22$. The corresponding solution value is $1 \cdot 8 + 2 \cdot 10 + 3 \cdot 22 = 94$ which matches the primal.

(d) All dual values are nonnegative as expected. Summing duals along path from the first composition tree node including both ends of each arc and comparing to its objective coefficient to check main dual constraints gives $22 = c_{12}$ for (1, 2), $22 > 16 = c_{14}$ for (1, 4), $8 + 10 + 22 = 40 = c_{23}$, $10 + 22 = 32 = c_{24}$, and $10 + 22 = 32 > 30 = c_{34}$. The dual solution is indeed feasible.

Computational Order of the Greedy Algorithm

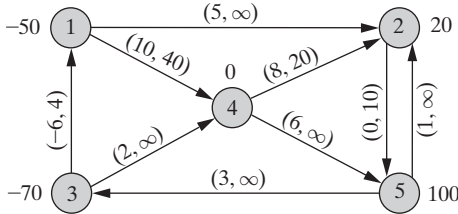
The core of greedy algorithm on graph $G(V, E)$ is a sort of edges by cost, which requires $O(|E| \log |E|)$ effort (see section 14.2), followed by examination of edges one-by-one until a tree is finalized, which could be $O(|E|)$ in the worst case. The more subtle issue is how to track whether new edges form cycles with ones already

chosen. This can be done by keeping a record of the component/subset number to which each node belongs as computation continues. Define $t_k \triangleq$ the component number including vertex k . The algorithm starts with each node with its own component or $t_k \leftarrow k$ for all $k \in V$. Then if the two ends of a candidate edge (i, j) have $t_i = t_j$, it joins 2 nodes in the same component and forms a cycle; it should be skipped. If $t_i \neq t_j$, then (i, j) connects different components. The algorithm accepts the edge in the solution and combines the components for i and j by replacing $t_j \leftarrow t_i$ for all nodes in component j before the selection. Such t -label updates are done $O(O(|V|))$ times as edges are selected and require $O(|V|)$ effort per update. Summarizing,

Principle 10.69 On given instance $G(V, E)$, Algorithm 10F runs in $O(|E|\log|E|) + O(|E|) + O(|V|^2) = O(|E|\log|E| + |V|^2)$ time.

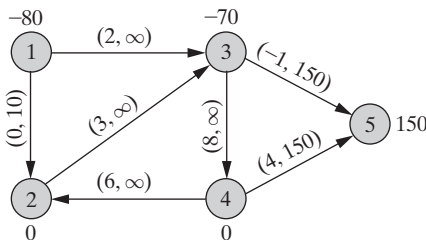
EXERCISES

10-1 The figure that follows depicts a minimum cost network flow problem. Numbers on the nodes show net demand, while those on arcs show unit cost and capacity.



- ✓ (a) Identify the node set V and the arc set A of the network.
- ✓ (b) Classify all nodes as source, sink, or transshipment.
- (c) Verify that total supply equals total demand.
- ✓ (d) Formulate the corresponding minimum cost network flow problem as a linear program.
- ✓ (e) Show the node–arc incidence matrix for the network.

10-2 Do Exercise 10-1 for the network



10-3 Consider the matrix

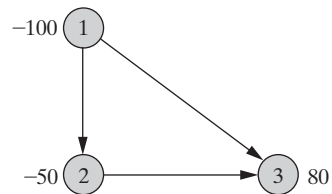
$$\begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

- ✓ (a) Explain why it is a node–arc incidence matrix.
- ✓ (b) Draw the corresponding digraph.

10-4 Do Exercise 10-3 for the matrix

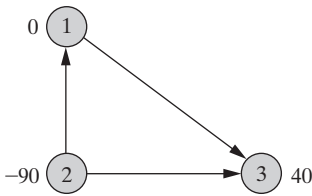
$$\begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

10-5 The following digraph represents a network flow problem with values at nodes showing net demand.



- ✓ (a) Verify that total supply exceeds total demand.
- ✓ (b) Add a new sink to create an equivalent network with total supply equal to total demand.

10-6 Do Exercise 10-5 for the network



10-7 Super Sleep is a company making mattresses for king-size beds. Mattresses can be shipped directly from either of its plants to retail store customers, or they may be transshipped through the company’s single warehouse. The table that follows shows the unit cost of shipping mattresses from the plants and the warehouse to Super Sleep’s 2 customers. Shipping from either plant to the warehouse costs \$15 per mattress. The table also shows the number of mattresses that can be produced at each plant over the next week and the number demanded by each customer.

	Shipping Costs		Capacity
	$j = 1$	$j = 2$	
Plant 1	25	30	400
Plant 2	45	23	600
Warehouse	11	14	—
Demand	160	700	—

Super Sleep seeks a minimum total shipping cost way to supply its customers.

- ✓ (a) Formulate a linear program to determine an optimal shipping plan.
- ✓ (b) Use class optimization software to compute an optimal solution to your LP.
- ☐ (c) Show that your LP can be represented as a minimum cost flow model (with total supply = total demand) by sketching the corresponding digraph and labeling as in Exercise 10-1.
- ✓ (d) Classify nodes in your digraph as source, sink, or transshipment.

10-8 Crazy Crude oil company can produce 1500 barrels per day from one of its fields and 1210 from the other. From there the crude oil can be piped to either of two tank farms, one at Axel and the other at Bull. Axel then trucks oil on to the Crazy Crude refinery at \$0.40 per barrel to help meet its daily demand of 2000 barrels. Bull

trucks to the refinery at \$0.33 per barrel. It costs Crazy Crude \$0.10 per barrel to pipe from field 1 to Axel and \$0.35 per barrel to pipe from field 1 to Bull. Corresponding values for field 2 are \$0.25 and \$0.56. Also, the tank farms can truck between themselves at \$0.12 per barrel.

Do (a) through (d) as in Exercise 10-7.

10-9 Return to the Super Sleep problem of Exercise 10-7, and suppose now that we wish to plan over a 2-week time horizon. Customer demands remains 160 and 700 in the first week, but they are predicted to be 300 and 810 in the second. There is no initial inventory, but mattresses may be held in the warehouse at \$10 per week. All other parameters are the same in each week as those given in Exercise 10-7.

- ✓ (a) Formulate a time-expanded linear program to determine an optimal shipping and holding plan.
- ✓ (b) Use class optimization software to compute an optimal solution to your LP.
- ☐ (c) Show that your LP can be represented as a minimum cost flow model (with total supply = total demand) by sketching the corresponding digraph and labeling as in Exercise 10-1.

10-10 Return to the Crazy Crude problem of Exercise 10-8, and suppose now that we wish to plan over a 2-day time horizon. Refinery demand remains 2000 on the first day but will be 3000 on the second. There are no initial inventories, but either tank farm can hold over petroleum at \$0.05 per barrel per day. All other parameters are the same on each day as those given in Exercise 10-8.

Do (a) through (c) as in Exercise 10-9.

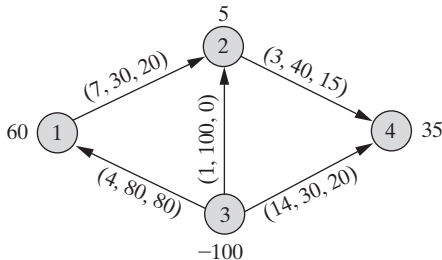
10-11 Determine whether each of the following sequences is a chain, a path, a cycle, and/or a dicycle of the network in Exercise 10-1.

- ✓ (a) 1–2–4–1
- ✓ (b) 3–4–2
- ✓ (c) 3–4–1
- ✓ (d) 3–4–5–3

10-12 Determine whether each of the following sequences is a chain, a path, a cycle and/or a dicycle of the network in Exercise 10-2.

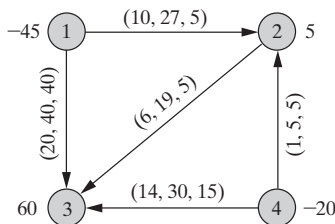
- (a) 2–3–4–2
- (b) 1–3–5–4
- (c) 3–4–2
- (d) 2–3–4–5

10-13 The following digraph shows a partially solved minimum cost network flow problem with node labels indicating net demand and arcs labels showing unit cost, capacity, and current flow.



- (a) Verify that the current flow is feasible.
- ✓ (b) Generate the six possible cycle directions of flow change.
- (c) Verify for the first (one) of your cycle directions that a step λ in the direction would leave flow balanced at all nodes.
- ✓ (d) Determine whether each of your cycle directions is improving.
- ✓ (e) Determine whether each of your cycle directions is feasible.
- ✓ (f) For those directions that are feasible, compute the maximum step size λ that could be applied without losing feasibility.

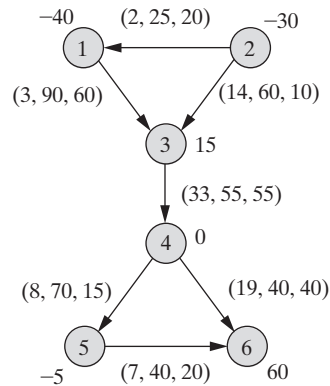
10-14 Do Exercise 10-13 for the network



10-15 Solve each of the following by rudimentary cycle direction Algorithm 10A starting from the solution given in the figure. Find needed cycle directions by inspection.

- ✓ (a) The network in Exercise 10-13
- (b) The network in Exercise 10-14

10-16 The digraph below shows an instance of the minimum cost network flow problem. Labels on arcs are (cost, capacity, current flow), and those on nodes are net demand.



- (a) Show the corresponding Node–Arc Incidence Matrix.
- (b) Start from the given solution and solve the instance to optimality by the Rudimentary Cycle Direction Algorithm 10A. Choose cycle directions to use by inspection, establish that each is improving and feasible, and show the revised flow after each iteration. Also identify the ultimate optimum.

10-17 Add an artificial node and artificial arcs to prepare each of the following networks for two-phase or big- M solution starting with a zero flow on all original arcs. Show the starting flow, and verify that it balances at the artificial node.

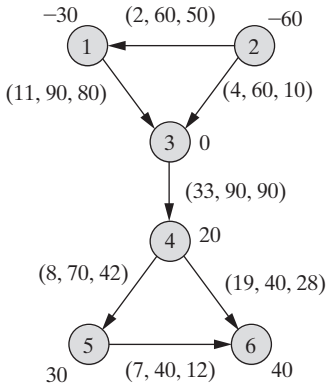
- ✓ (a) The network of Exercise 10-1
- (b) The network of Exercise 10-2

10-18 Refer to the partially solved minimum cost network flow problem of Exercise 10-25.

- ✓ (a) Verify that the given flow is feasible.
- ✓ (b) Construct the residual digraph corresponding to the current flow.
- ✓ (c) Use Floyd-Warshall Algorithm 9B on your residual digraph to identify an improving feasible cycle direction with respect to the given flow.
- ✓ (d) Compute the maximum feasible step λ that could be applied to your cycle direction.

10-19 Do Exercise 10-18 on the network of Exercise 10-26.

10-20 The digraph below shows a minimum cost network flow instance. Labels on arcs are (cost, capacity, current flow). Labels on the nodes show net demand.



- ✓ (a) Establish that the given flow is feasible.
- ✓ (b) Start from the given solution and solve the instance to optimality by the Cycle Cancelling Algorithm 10B. Show the residual digraph at each step, but choose improving feasible cycle directions from them by inspection.
- (c) Compute bound $\lfloor 10.26 \rfloor$ on the Algorithm 10B computational effort for this instance and compare to the number of steps in your solution of part (b).

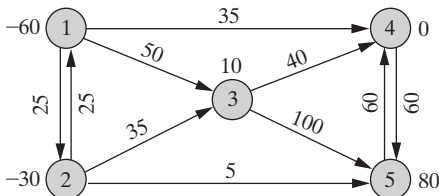
10-21 Do Exercise 10-20(b) on each of the following networks.

- (a) The network of Exercise 10-13
- (b) The network of Exercise 10-14
- (c) The network of Exercise 10-25
- (d) The network of Exercise 10-26

10-22 Demonstrate that columns of the node-arc incidence matrix corresponding to arcs in each of the following cycles of the digraph in Exercise 10-1 form a linearly dependent set.

- ✓ (a) (2, 5), (5, 2)
- (b) (4, 2), (5, 2), (4, 5)
- ✓ (c) (1, 2), (4, 2), (1, 4)
- (d) (1, 4), (4, 5), (5, 3), (3, 1)

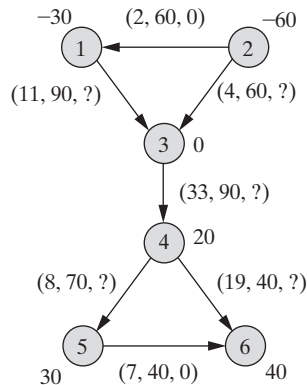
10-23 The following depicts a network flow problem, with labels on nodes indicating net demand and those on arcs showing capacity.



For each of the following lists of possible nonbasic arcs, either compute the corresponding basic solution and indicate whether it is basic feasible, or apply principle $\lfloor 10.33 \rfloor$ to demonstrate that the unlisted arcs do not form a basis of the implied flow balance constraints.

- ✓ (a) (1, 2), (2, 1), (3, 4), (5, 4) lower-bounded, (1, 3), (2, 5) upper-bounded
- (b) (1, 2), (1, 4), (3, 4), (4, 5), (5, 4) lower-bounded, (1, 3), (3, 5) upper-bounded
- ✓ (c) (1, 3), (2, 1), (2, 5), (3, 4), (5, 4) lower-bounded, (1, 4) upper-bounded
- (d) (1, 2), (1, 4), (2, 3), (3, 4) lower-bounded, (2, 5) upper-bounded
- ✓ (e) (1, 2), (1, 4), (4, 5) lower-bounded, (2, 5), (3, 4) upper-bounded
- (f) (1, 2), (2, 1), (2, 5), (3, 4), (5, 4) lower-bounded, (1, 4) upper-bounded
- ✓ (g) (1, 2), (1, 4), (3, 4), (5, 4) lower-bounded, (1, 3), (2, 5), (3, 5) upper-bounded
- (h) (1, 2), (1, 4), (3, 4), (4, 5) lower-bounded, (1, 3), (2, 5) upper-bounded

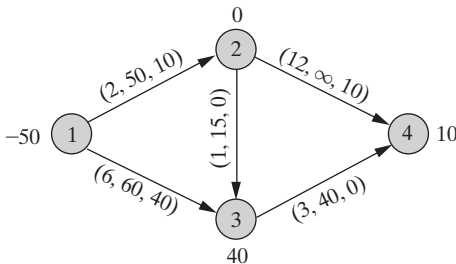
10-24 Return to the minimum cost network flow instance of Exercise 10-20. The figure below depicts the same instance with different values for the current flow, including many marked ‘?’ for unknown.



- ✓ (a) Demonstrate that arcs with flow indicated as ‘?’ form a basis of the corresponding LP.
- ✓ (b) Compute the corresponding basic solution and establish that it is feasible.
- ✓ (c) Explain why the basic solution of (b) is degenerate.

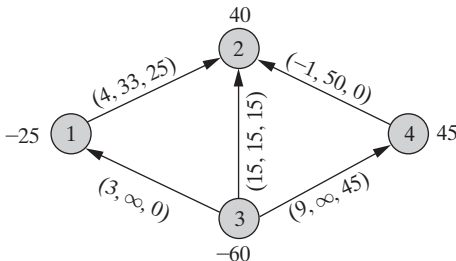
- ✔ (d) Start from the solution of (b), and apply Network Simplex Algorithm 10C to compute an optimal solution. Show details of reduced costs, stepsize computation, basis updates, etc.

10-25 The following digraph depicts a partially solved minimum cost flow problem with labels on nodes indicating net demand and those on arcs showing unit cost, capacity, and current flow.



- ✔ (a) Verify that the given solution is a basic feasible solution for basis $\{(1, 2), (1, 3), (2, 4)\}$.
- ✔ (b) Compute all simplex directions available at this basis.
- ✔ (c) Determine whether each of the simplex directions is improving.
- ✔ (d) Regardless of whether they are improving, determine the maximum feasible step λ that could be applied to each of the simplex directions.

10-26 Do Exercise 10-25 on the problem.

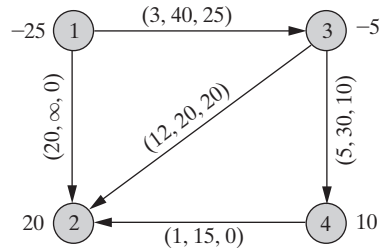


and basis $\{(1, 2), (3, 1), (3, 4)\}$.

10-27 Apply network simplex Algorithm 10C to compute an optimal flow in each of the following networks. Start from the flow given in the figure, using the basis specified below.

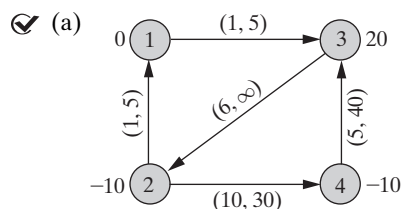
- ✔ (a) The network of Exercise 10-13 with basis $\{(1, 2), (2, 4), (3, 4)\}$
- (b) The network of Exercise 10-14 with basis $\{(1, 2), (2, 3), (4, 3)\}$
- ✔ (c) The network of Exercise 10-25 with basis $\{(1, 2), (1, 3), (2, 4)\}$
- (d) The network of Exercise 10-26 with basis $\{(1, 2), (3, 1), (3, 4)\}$

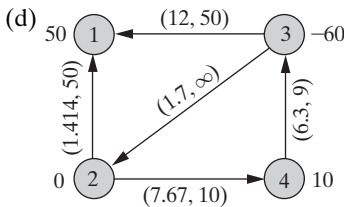
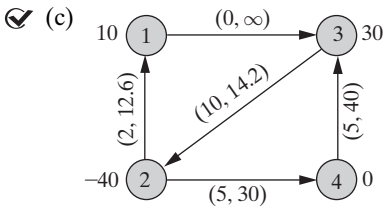
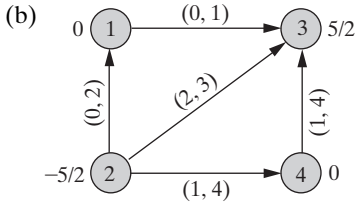
10-28 The following digraph depicts a partially solved minimum cost flow problem with labels on the nodes indicating net demand and those on the arcs showing unit cost, capacity, and current flow.



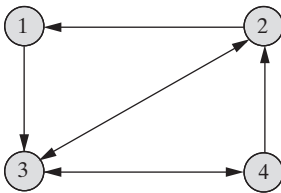
- ✔ (a) Verify that the given solution is a basic feasible solution for basis $\{(1, 2), (1, 3), (3, 4)\}$.
- ✔ (b) Identify a cycle for which the corresponding cycle direction could be pursued by cycle cancelling Algorithm 10B but not by network simplex Algorithm 10C with the basis of part (a).
- ✔ (c) Identify a cycle for which the corresponding cycle direction could be pursued by network simplex Algorithm 10C with the basis of part (a) but not by cycle cancelling Algorithm 10B.

10-29 Each of the following depicts a minimum cost network flow problem, with labels on nodes indicating net demand and those on arcs showing unit cost and capacity. Determine for each (without solving) whether any unique optimal flow would have to be integer valued.





10-30 Consider the following digraph.



- ✓ (a) Exhibit the Node–Arc Incidence Matrix (NAIM) of this digraph.
- ✓ (b) Select a column submatrix of your NAIM with the maximum possible number of linearly independent columns.
- ✓ (c) Demonstrate that your submatrix of (b) has determinant +1 or -1 after one row is deleted to make the submatrix full row rank.
- ✓ (d) Confirm that your NAIM is totally unimodular by selecting two 2 by 2, two 3 by 3, and two more 4 by 4 submatrices, then showing all have determinant = 0, +1 or -1.

10-31 The Quick Chip gravel company has received a contract to supply two new construction

projects in the towns of Brock and Wurst. A total of 60 truckloads are needed at Brock in the next month and 90 at Wurst. Quick Chip has idle gravel pits in the towns of Nova, Scova, and Tova, each with a monthly production capacity of 50 truckloads. Travel distances from each pit to each project site are shown in the following table.

Pit	To Brock	To Wurst
Nova	23	77
Scova	8	94
Tova	53	41

The company wants to fulfill its contract at least total truck travel distance.

- ✓ (a) Formulate a linear program to choose an optimal shipping plan.
- ✓ (b) Use class optimization software to compute an optimal solution to your LP.
- ✓ (c) Show that your LP can be represented as a minimum cost transportation problem by sketching the corresponding bipartite digraph and labeling as in Exercise 10-1.

10-32 Maize Mills has 800 thousand, 740 thousand, and 460 thousand bushels of corn stored at its three rural elevators. Its three processing plants will soon require 220 thousand, 1060 thousand, and 720 thousand respectively, to make cornstarch. The following table shows the cost per thousand bushels of shipping from each elevator to each plant.

Elevator	Plant		
	1	2	3
1	10	13	22
2	15	12	11
3	17	14	19

Maize wants to move its corn to plants at minimum total shipping cost.

Do (a) through (c) as in Exercise 10-31.

10-33 The table below shows the results of triaging 7 persons injured in a recent terrorist attack. Each patient has been scored (1 = lowest to 5 = highest) according the compatibility of his/her needs vs. the capabilities of each of the 4 available hospitals. No more than 3 patients can be sent to any hospital.

Patient	Hospital			
	1	2	3	4
1	2	1	5	3
2	3	3	1	2
3	1	1	5	2
4	4	2	3	2
5	3	1	3	2
6	4	2	4	1
7	2	3	3	5

- (a) Assign a symbolic parameter name to the values in the table, including detailing required subscripts.
- (b) Formulate this problem as an instance of the Transportation Problem in terms of appropriate decision variables and the parameters of part (a).
- (c) Draw the corresponding bipartite network, including supplies/demands on nodes, and costs/scores on arcs.
- (d) Explain why flow requirements in your formulation of part (b) are not balanced to have total supply = total demand. Then briefly explain how to modify the model to obtain an equivalent one that is balanced.

10-34 Senior design students are negotiating which of the four members of the team will take primary responsibility for each of the four project tasks the team must complete. The following table shows the composite ratings (0 to 100) they have prepared to estimate the ability of each member to manage each task.

Member	Task Rating			
	1	2	3	4
1	90	78	45	69
2	11	71	50	89
3	88	90	85	93
4	40	80	65	39

The team wants to find a maximum total score plan that allocates exactly one task to each team member.

- ✓ (a) Formulate a linear assignment problem (LP) to choose an optimal plan.
- ✓ (b) Use class optimization software to compute an optimal assignment.

- ✓ (c) Show that your assignment problem can be represented as a minimum total cost flow model by sketching the corresponding bipartite digraph and labeling as in Exercise 10-1.
- ✓ (d) A feasible assignment must have decision variables = 0 or 1 in part (a), yet the model can be solved as a linear program. Explain how the underlying network nature of the problem makes this possible.

10-35 Paltry Properties has just acquired four rental homes. Paltry wishes to have the houses painted within the next week so that all can be available for the prime rental season. This means that each house will have to be painted by a different contractor. The following table shows the bids (thousands of dollars) received from four contractors on the four houses.

House	Painter Bid			
	1	2	3	4
1	2.5	1.3	3.6	1.8
2	2.9	1.4	5.0	2.2
3	2.2	1.6	3.2	2.4
4	3.1	1.8	4.0	2.5

Paltry want to decide which bids to accept in order to paint all houses at minimum total cost.

Do (a) through (d) as in Exercise 10-34.

10-36 The following table shows the weights for assigning rows i to columns j in a maximum total weight assignment problem.

	$j = 4$	5	6
$i = 1$	25	13	22
2	21	14	19
3	20	25	29

- ✓ (a) Formulate the model as a linear assignment model.
- ✓ (b) Construct the starting dual solution and equality subgraph of Hungarian Algorithm 10D for the given weights.
- ✓ (c) Make a starting assignment of (2 to 4) and (3, 6). Then explain why this solution is complementary with the dual of part (a).
- ✓ (d) Beginning from the solution of (c) complete Hungarian Algorithm 10D to identify an optimal assignment. Detail the labeling to grow the assignment and/or

associated label trees at each step, as well as computations to change duals. Also verify at each dual change that no assignment of tree-labeled edges are dropped in the equality subgraph as the dual is updated.

- ✔ (e) Show the optimal primal and dual solutions, and demonstrate that they are complementary.
- ✔ (f) Compute bound $\overline{10.49}$ on the Algorithm 10D computational effort for this instance and compare to the number of steps in your solution of part (c).

10-37 Do Exercise 10-36 on each of the following assignment models.

- (a) The digraph of Exercise 10-34(c).
- (b) The digraph of Exercise 10-35(c).

10-38 A relief agency is urgently trying to get the maximum possible quantity of supplies from its base at Alto to the volcano-ravaged city of Epi. One available road goes via Billi. The agency estimates that the Alto-to-Billi part of that road can carry 500 tons per day, and the Billi-to-Epi segment, 320 tons. A second route goes via Chau and Domo, with the Alto-to-Chau part having capacity 650 tons, the Chau-to-Domo section, 470, and the Domo-to-Epi segment, 800. There is also a small mountain road with capacity 80 tons that connects Billi to Domo.

- ✔ (a) Formulate this problem as a maximum flow model by sketching the associated digraph. Indicate the source, the sink, and all capacities.
- ✔ (b) Solve your maximum flow problem by inspection.
- ✔ (c) Show how to modify your digraph of part (a) as in Figure 10.25 to represent the model as a minimum cost network flow problem.

10-39 Makers of the new Ditti Doll are urgently trying to get as many to market as possible because a craze has created almost unlimited demand. One plant can supply up to 8 thousand per week to its distribution center, but that center can then get only 3 thousand per week to east region customers, and 1 thousand to west. The other plant supplies up to 3 thousand per week to its

(distinct) distribution center, which can ship 2 thousand per week to the east and the same number to the west.

Do (a) through (c) as in Exercise 10-38.

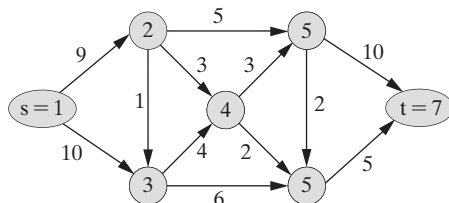
10-40 Formulate as a minimum cost network flow problem, and solve by inspection, the problem of finding a maximum flow from the specified source to the specified sink in each of the following networks. Use capacities specified on the original digraph.

- ✔ (a) Source 3, sink 2 in the network of Exercise 10-13
- (b) Source 1, sink 3 in the network of Exercise 10-14
- ✔ (c) Source 1, sink 4 in the network of Exercise 10-35
- (d) Source 3, sink 2 in the network of Exercise 10-26

10-41 The capital city Capria of the remote Republic of Democracio (ROD) is under growing attack from murderous terrorist forces. To defend themselves the ROD forces in Capria have an urgent need for rocket propelled grenades (RPGs) which they use in close combat. There is a supply of 400 RPGs in the ROD city of Butey, but the only way they can be shipped to Capria is along a mountainous road supporting trucks able to carry a total of 250 per day. There are also 1200 units of the weapon available in a European depot, and 10,000 at one in North America. Large cargo, planes can fly 500 RPGs from Europe and/or 800 from North America per day to Moderna, which is the only unoccupied airfield in ROD. From there the weapons must be sent via a narrow-gauge railroad capable of shipping 1600 units per day through the mountains to Capria. ROD leaders and their allies wish to formulate a plan to get the maximum number of RPGs to Capria over the next 3 days.

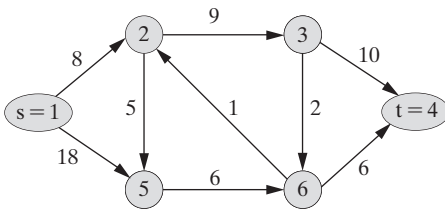
Do (a) through (c) as in Exercise 10-38.

10-42 Consider the following digraph.



- (a) Starting from an all zero flow, apply Bellman Ford Algorithm 10E to compute a max flow from node s to node t . Show the residual digraph at each iteration, and choose your augmenting paths by inspection.
- (b) Process the final residual digraph of (a) to identify a min cut, and verify that max flow = min cut in this instance.
- (c) Repeat parts (a) and (b), this time augmenting along shortest cardinality paths in the residual digraphs.
- (d) Comment on how results of the two computations differ if at all, and why.
- (e) Compute bound 10.55 and compare to your actual effort in parts (a) and (c).

10-43 Consider the digraph below. Numbers on arcs are capacities.



Do (a)–(e) of Exercise 10-42.

10-44 Do Exercise 10-42(a)–(b) for each of the following maximum flow models:

- (a) Maximum flow Exercise 10-38
- (b) Maximum flow Exercise 10-39
- (c) Maximum flow Exercise 10-41

10-45 Although Dynamic Programming methods of Chapter 9 are usually more efficient, the problem of finding a shortest path from a given origin s to a destination t in a graph with no negative dicycles can easily be represented as a minimum cost flow problem. Using arc lengths as costs, it is only necessary to make s a node with supply = 1, t a node with demand = 1, and treat all other nodes as transshipment.

- (a) Illustrate and justify how this produces an optimal path by formulating the problem of finding a shortest path from $s = 3$ to $t = 5$ in the digraph of Exercise 10-1.
- (b) Do part (a) for a path from $s = 1$ to $t = 5$ in the digraph of Exercise 10-2.

10-46 Three workstations are located on the circular conveyor of a manufacturing facility. Most of the flow between them moves from one station to the next on the conveyor. However, 7 units per minute must move from each station to the station after the next. As much as possible of this 2-step flow should be carried within the 11 units per minute capacity of each link of the conveyor. The rest will be transported manually.

- ✓ (a) Formulate a linear program to determine an optimal way to carry the flows.
- ✓ (b) Use class optimization software to compute an optimal solution to your LP.
- ✓ (c) Show that your LP can be represented as a multicommodity flow problem by sketching the corresponding digraph and labeling with costs, capacities, and net demands.
- ✓ (d) Explain why your multicommodity model would give meaningless results if all flow were combined into a single commodity.
- ✓ (e) The optimal solution in this problem is fractional. Explain how this is compatible with the network nature of your multicommodity flow model.

10-47 The Wonder Waste disposal company has 5 truckloads of nuclear waste and 5 truckloads of hazardous chemical wastes that must be moved from its current cleanup site to nuclear and chemical disposal facilities, respectively. The following table shows that many of the available roads are restricted for one or the other type of waste.

Road		Nuclear OK	Chemical OK
From	To		
Site	NDisp	Yes	No
Site	CDisp	Yes	Yes
Site	Inter	No	Yes
NDisp	CDisp	No	Yes
CDisp	Inter	Yes	No
Inter	NDisp	Yes	Yes

Also, Wonder Waste wants to distribute any risk by allowing no more than half the 10 total truckloads on any road. One particular road, the link from crossing Inter to the nuclear disposal facility, is especially well suited to hazardous transfer

because it runs through very remote areas. Wonder Waste seeks a feasible shipping plan that maximizes the use of that road.

Do (a) through (e) as in Exercise 10-44.

10-48 Maine Miracle’s 2 restaurants sell lobsters obtained from 3 fisherman. A total of 350 lobsters per day are served at the first restaurant, and 275 at the second. Each fisherman can ship up to 300 per day, but not all arrive suitable for serving. The following table show the cost (including shipping) and the yield of servable lobsters for each combination of fisherman and restaurant.

Fisherman	Restaurant			
	Cost		Yield (%)	
	1	2	1	2
1	\$7	\$7	70	60
2	8	8	80	80
3	5	5	60	70

Maine Miracle seeks a minimum total cost way to meet its restaurant needs.

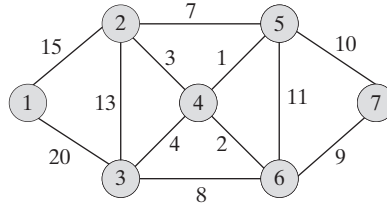
- ✔ (a) Formulate a linear program to determine an optimal plan.
- ✔ (b) Use class optimization software to compute an optimal solution to your LP.
- ✔ (c) Show that your LP can be represented as a minimum cost flow model with gains or losses by sketching the corresponding digraph and labeling as in Example 10.30.
- ✔ (d) The optimal solution in this problem is fractional. Explain how this is compatible with the network nature of your flow with gains or losses model.

10-49 A new grocery store has 3 weeks to train its full staff of 39 employees. There are 5 employees now. At least 2 employees must work on preparing the store during the next week, 5 employees the week after, and 10 in the final week before opening. Employees assigned to these duties earn \$300 per week. Any other available employees, including those who were themselves trained in just the preceding week, can be assigned to train new workers. If an employee trains just one other, the two of them cost \$500 for the week. If two are trained, the three

employees cost \$800 per week, including trainer overtime. Managements seeks a minimum total cost plan to meet all requirements.

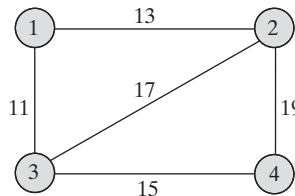
Do (a) through (d) as in Exercise 10-48. You may use nonzero lower bounds on some variables.

10-50 Consider the following undirected graph, taking numbers on edges as costs/weights.



- ✔ (a) Apply standard Greedy Algorithm 10F to compute a maximum spanning tree of the graph.
- ✔ (b) Sketch a tree like Figure 10.30 to track the new subtrees formed as edges are added to the solution, as well as the weight of the edge added to create each subtree.
- ✔ (c) List the active primal constraints associated with each tree in (b).
- ✔ (d) Use the structure of (b) to determine dual variable values for those active constraints.
- ✔ (e) Combine (a)–(d) to demonstrate that your greedy solution is primal feasible in the LP relaxation of the problem, your dual solution of (d) is dual feasible, and that together they mutually satisfy complementary slackness.
- (f) Compute bound 10.69 and compare to your actual computational effort of part (a).

10-51 The following graph shows an instance of the Maximum Spanning Tree Problem.



Do (a)–(f) of Exercise 10-50.

10-52 Hilltop University (HU) is building a new campus on one of the highest hills in its hometown. The following table shows details about 6 key buildings under construction at the new site.

No.	Name	Coords		Altitude
		x	y	
1	Administration	100	100	300
2	Library	52	55	210
3	Student Union	151	125	204
4	Engineering	50	208	150
5	Management	147	25	142
6	Residence Hall	210	202	100

HU wants to construct a minimum total length system of sidewalks that contains a path from every building to every other, taking into account both the distance between their sites and the steepness caused by changes in altitude. Specifically a link joining buildings i and j should be viewed as having weighted length.

$$d_{ij} \triangleq (\text{Euclidean distance from } i \text{ to } j) \times (1 + \text{Absolute difference of altitudes for } i \text{ and } j)$$

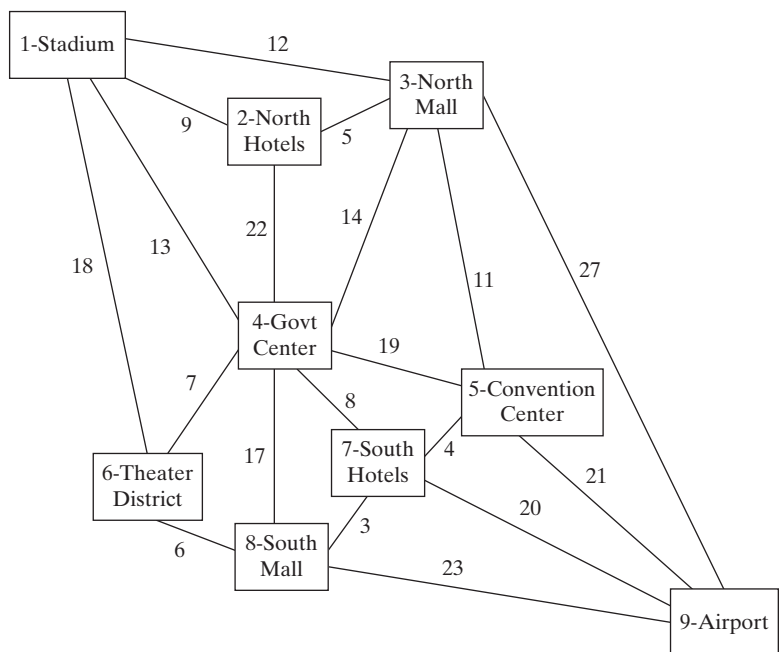
- (a) Compute lengths d_{ij} defined above for all pairs of buildings i and j .
- (b) Explain why an optimal sidewalk network for the new HU campus will be a minimum total weight spanning tree of the complete graph with nodes at the 6 buildings and edges of weight computed in part (a).

- (c) Apply Greedy Algorithm 10F to compute such a minimum weight spanning tree, commenting on how subtrees are formed and merged as the algorithm proceeds, and on why some attractive edges were skipped.

10-53 The Metropolis Regional Authority (MRA) is planning the construction of a light rail network connecting the major activity centers of the region. The figure below shows the 9 centers involved along with possible segments connecting them. Estimated costs of construction are also shown on each such link (in \$ million).

Eventually traffic on the chosen background network will be organized into routes with intermediate stops. For now, however, MRA simply wants to find the least total cost background network that offers a path between each pair of the 9 centers.

- (a) Explain why an optimal background network for the MRA light rail will be a minimum total weight spanning tree of the graph below.
- (b) Apply Greedy Algorithm 10F to compute such a minimum weight spanning tree, commenting on how subtrees are formed and merged as the algorithm proceeds, and on why some attractive edges were skipped.



10-54 Forest fire control organizations in Canada's provinces⁶ must reposition the numbers of observation aircraft available at stations $i = 1, \dots, 11$ on a daily basis to adjust for changing fire threats. The required number r_i and the present number p_i are known for all stations, along with the cost $c_{i,j}$ of moving an aircraft from any station i to station j . Explain how the problem of choosing a minimum total cost plan for repositioning aircraft can be modeled as a transportation problem by identifying the sources, sinks, supplies, demands, and arc costs.

10-55 A new highway⁷ is being built through terrain points $i = 1, \dots, 40$. The distance from i to $i + 1$ is d_i . To level the route, net earth deficits b_i truckloads must be corrected at all nodes ($b_i < 0$ if surplus, $\sum_i b_i = 0$). This will be done by moving truckloads of earth from surplus to deficit points along the route of the highway, but the same earth should not be handled more than once. Explain how the problem of choosing a minimum total truck travel distance leveling plan can be modeled as a transportation problem by identifying the sources, sinks, supplies, demands, and arc costs.

10-56 To estimate the impact of proposed tax changes, the U.S. Department of the Treasury⁸ maintains two data files of records statistically characterizing the taxpayer population. Each of the $i = 1, \dots, 10,000$ records in the first file represents a known number of families a_i and describes corresponding characteristics such as family size and age distribution. Records $j = 1, \dots, 40,000$ in the second file also represent a known number of families b_j and contain some of the same characteristics as the first ($\sum_i a_i = \sum_i b_i$). However, most of the entries in the second file relate to the sources of income for family class j . To do a better job of analyzing proposals, the Treasury wants to merge these files into one with new records

containing information drawn from both inputs. Each new record will represent a collection of families formed by matching some or all of those in population a_i with some of all in b_j . The quality of the similarity between classes i of the first file and j of the second can be described by a distance measure $d_{i,j}$, and the Treasury seeks a minimum total distance merge. Explain how this problem can be modeled as a transportation problem by identify sources, sinks, supplies, demands, and costs. Also, explain how an optimal flow can be understood as a merge.

10-57 Freight trains⁹ run a regular weekly schedule in both the forward and reverse directions of a railroad's main line through section boundary points $i = 1, \dots, 22$. Dividing the week into hourly time blocks $t = 1, \dots, 168$ (with $t = 1$ following $t = 168$), a train leaving i bound for $j > i$ advances through both time and space as it transits sections $(i, i + 1)$, $(i + 1, i + 2)$, and so on. Summing requirements for all scheduled trains to pull the anticipated load over grades in the section, engine needs $f_{i,t}$ can be estimated for each forward section $(i, i + 1)$, and each time t to $t + p_i$, where p_i is the number of hours required to transit the section. Reverse requirements $r_{i,t}$ provide the same information for trains moving i at t to $i - 1$ at $t + q_i$ with q_i the time to transit segment $(i, i - 1)$. We assume that all engine units are identical. Those located at any place and time where they are not immediately needed, may be held there, turned around to go the opposite way, or added as extras on a passing train. Cost c_i , d_i , and h reflect the cost of running one unit over segment $(i, i + 1)$, running one over $(i, i - 1)$, and holding for an hour at any site, respectively. Show that the problem of computing a minimum total cost engine schedule can be modeled as a network flow problem over a suitable time-expanded network, by sketching a representative node and showing all adjacent nodes, arcs, costs,

⁶Based on P. Kourtz (1984), "A Network Approach to Least-Cost Daily Transfers of Forest Fire Control Resources," *INFOR*, 22, 283–290.

⁷Based on A. M. Farley (1980), "Levelling Terrain Trees: A Transshipment Problem," *Information Processing Letters*, 10, 189–192.

⁸Based on F. Glover and D. Klingman (1977), "Network Application in Industry and Government," *AIIE Transactions*, 9, 363–376.

⁹Based on M. Florian, G. Bushell, J. Ferland, G. Guerin, and L. Nastanshy (1976), "The Engine Scheduling Problem in a Railway Network," *INFOR*, 14, 121–138.

bounds, and node net demands. Some arcs will have nonzero lower bounds.

10-58 A substantial part of United Parcel Service's¹⁰ freight traffic moves as trailer-on-flatcars (i.e., with truck trailers traveling most of the way on railroad flatcars). The required number of truckloads $d_{i,j}$ to be shipped between points $i, j = 1, \dots, n$ in this way is known, but UPS can use either its own trailers at unit cost $c_{i,j}$ or rent trailers from the railroad at unit cost $r_{i,j}$. Rented trailers can be left anywhere, but UPS wishes to balance the number of its own trailers available at every point. That is, the number of company trailers inbound at any point should equal the number outbound. If necessary, trailers may be returned empty from i to j at unit cost $e_{i,j}$ to meet this requirement. Show that the problem of finding a minimum total cost shipping plan can be modeled as a (single-commodity) network flow problem by detailing the arcs that would join each node i to another j , including both their cost and any applicable capacity. Also indicate the net flow demand at each node. (*Hint*: Represent rented trailer flows indirectly from flows of loaded company-owned trailers with capacity $d_{i,j}$.)

10-59 KS brand tires¹¹ are shaped in changeable molds type $i = 1, \dots, m$, installed in the company's 40 presses. Production plans dictate the minimum $r_{i,t}$ and maximum $R_{i,t}$ numbers of molds i that should be operational during time period $t = 1, \dots, n$, and the planning period begins $t = 0$ with numbers of molds b_i installed ($\sum_i b_i = 40$). There is an adequate supply of molds, but changeovers from one mold to another are expensive, costing an amount c_i depending on the mold installed. All 40 presses should be in use in each time period. Demonstrate that the problem of finding a minimum total changeover cost mold schedule can be modeled as a network flow on a suitable time-expanded network by sketching the graph for a case with $m = 2$ and $n = 3$, and

showing all nodes, arcs, costs, bounds, and node net demands. Some arcs will have nonzero lower bounds. (*Hint*: Also include supernodes balancing the total number of molds removed and inserted in each time period.)

10-60 Major league baseball umpires¹² work in crews that move among league cities to officiate series of 2–4 games. After every series, all crews move on to another that must involve different teams. To provide adequate travel time, any crew that works a series ending with a night game must also not go directly to one starting with a day game. Within these limits, league management would like to plan crew rotation to minimize the total of city i -to-city j travel costs $c_{i,j}$. Experience has shown that good results can be obtained by deciding each all crew move independently (i.e., without regard to where crews were before the most recent series or where they will be after the next). Explain how the problem of planning a move can be modeled as a linear assignment problem by describing the two sets being matched, the collection of feasible pairings, the associated linear costs, and whether the total is to be minimized or maximized.

10-61 One of the ways that airlines operating through major hubs can improve their service is to allow as many transferring passengers as possible who land at one of the scheduled arrival–departure peaks to carry on with their next flight on the same plane.¹³ Such through-flight connections must involve flights scheduled for the same type of aircraft, and flights with sufficient arrival to departure time to complete the required servicing. The number of passengers $p_{i,j}$ arriving on flight i and continuing on flight j at any peak can be estimated in advance. Explain how this problem of optimizing through flights can be modeled as a linear assignment problem by describing the two sets being matched, the collection of feasible pairings, the associated linear costs, and whether the total is to be minimized or maximized.

¹⁰Based on R. B. Dial (1994), “Minimizing Trailer-on-Flat-Car Costs: A Network Optimization Model,” *Transportation Science*, 28, 24–35.

¹¹Based on R. R. Love and R. R. Vemuganti (1978), “The Single-Plant Mold Allocation Problem with Capacity and Changeover Restrictions,” *Operations Research*, 26, 159–165.

¹²Based on J. R. Evans, (1988), “A Microcomputer-Based Decision Support System for Scheduling Umpires in the American Baseball League,” *Interfaces*, 18:6, 42–51.

¹³Based on J. F. Bard and I. G. Cunningham (1987), “Improving Through-Flight Schedules,” *IIE Transactions*, 19, 242–250.

10-62 As commercial airliner¹⁴ makes stops $j = 1, \dots, n$ of its daily routine and returns to where it started it takes on fuel for the next leg. Fuel is added at stop j to assure that the plane will arrive at stop $j + 1$ with at least the required safety reserve r_{j+1} . Fuel unit costs c_j (dollars per pound) vary considerably from stop to stop, so it is sometimes economical to carry more fuel than the minimum required in order to buy less at high-cost stops. However, the takeoff fuel load at any j must not exceed safety limit t_j . The amount of fuel at takeoff also affects the weight of the aircraft and thus its fuel consumption during flight. For each leg from stop j to $j + 1$, the fuel required can be estimated as a constant α_j plus a slope β_j times the onboard fuel at takeoff from j .

- (a) Formulate this fuel management problem as a linear program in the decision variables ($j = 1, \dots, n$)

$x_j \triangleq$ fuel added at stop j
 $y_j \triangleq$ onboard fuel at takeoff from j

Assume that stop 1 is the successor of stop n , and use nonzero lower bounds if needed.

- (b) Show how your model can be viewed as a flow with gains where the x_j correspond to 1-ended arcs, and the arcs associated with the y_j have both upper and (nonzero) lower bounds. Sketch the graph for a case with $n = 3$ and show all nodes, arcs, costs, bounds, and node net demands.

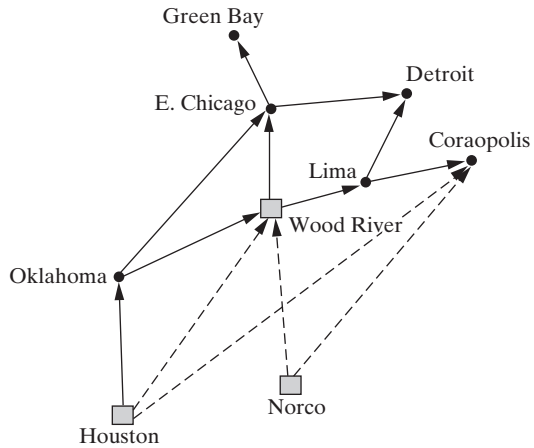
10-63 American Olean¹⁵ makes product families $i = 1, \dots, 10$ of tile at plants $p = 1, \dots, 4$ to meet demands $d_{i,k}$ (square feet) at sales distribution points (SDPs) $k = 1, \dots, 120$. Variable costs of production and transportation total $c_{i,p,k}$ per square foot to make tile of family i at plant p and ship to SDP k . Each plant can produce up to 100% of capacity, with $u_{i,p}$ being the capacity if plant p makes only group i . Management wants to find a minimum total cost way to meet demand.

- (a) Formulate an LP model to compute an optimal plan in terms of the decision variables ($i = 1, \dots, 10; p = 1, \dots, 4; k = 1, \dots, 120$)

$x_{i,p,k} \triangleq$ fraction of capacity at plant p devoted to making tile family i for shipment to SDP k

- (b) Show that your model can be viewed as a flow with gains of the transportation problem type by identifying sources, sinks, supplies, demands, arc gain multipliers, and arc costs.

10-64 The figure that follows shows a part of a distribution network like those used by Shell Oil Company¹⁶ to supply the midwest with its three main classes of product: gasoline, kerosene/jet fuel, and fuel oil.



Solid links shown indicate available pipelines, but products can also be shipped by barge from the refineries at Houston and Norco to depots at Wood River and/or Coraopolis. Wood River is also a refinery. Flows must meet known demands $d_{i,p}$ for the various products p at all nodes i . The three refineries have known production capacities

¹⁴Based on J. S. Stroup and R. D. Wollmer (1992), "A Fuel Management Model for the Airline Industry," *Operations Research*, 40, 229–237.

¹⁵Based on M. J. Liberatore and T. Miller (1985), "A Hierarchical Production Planning System," *Interfaces* 15:4, 1–11.

¹⁶Based on T. K. Zierer, W. A. Mitchell, and T. R. White (1976), "Practical Applications of Linear Programming to Shell's Distribution Problems," *Interfaces*, 6:4, 13–26.

$b_{i,p}$ for production of product p at refinery i , and a pipeline from point i to point j can carry combined total of at most $u_{i,j}$ barrels of product. Barge capacity is essentially unlimited. Refining costs are assumed fixed, and cost per barrel $c_{i,j}$ to transport along links from i to j is the same for all products.

- (a) Formulate an LP model to determine a minimum total cost distribution plan.
- (b) Show that your model can be viewed as a multicommodity flow problem by sketching the corresponding network, labeling arcs with costs and capacities, and nodes with net demands.

REFERENCES

Ahuja, Ravindra K., Thomas L. Magnanti, and James B. Orlin (1993), *Network Flows*, Prentice Hall, Upper Saddle River, New Jersey.

Bazaraa, Mokhtar, John J. Jarvis, and Hanif D. Sherali (2010), *Linear Programming and Network Flows*, John Wiley, Hoboken, New Jersey.

Lawler, Eugene (1976), *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehardt and Winston, New York, New York.

This page intentionally left blank

Discrete Optimization Models

Most network flow, shortest path, and dynamic programming models of Chapters 9 and 10 solve elegantly, even if decision variables are treated as discrete. Fortunately, such special discrete models do occur in operations research practice. Unfortunately, they are hardly the norm. The overwhelming majority of integer and combinatorial optimization models prove much more challenging.

Before exploring methods for dealing with hard discrete models, we need to get some concept of their enormous range. In this chapter we formulate a series of classic types using real application contexts often drawn from published reports. In Chapter 12 we address integer programming methods.

11.1 LUMPY LINEAR PROGRAMS AND FIXED CHARGES

One broad class of discrete optimization problems add either/or side constraints or objective functions to what is otherwise a linear program. For want of a better term, we might call such models **lumpy linear programs**.

Swedish Steel Application with All-or-Nothing Constraints

The Swedish steel blending application of Section 4.2 provides an illustration. The model repeated below, which was formulated there, chooses a minimum cost mix of scrap metal and pure additives to produce a “charge” of steel. Constraints restrict the chemical content of the charge.

$$\begin{aligned}
 \min \quad & 16x_1 + 10x_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 && \text{(cost)} \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 &= & 1000 && \text{(weight)} \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 &\geq & 6.5 && \text{(carbon)} \\
 & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 &\leq & 7.5 && \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 &\geq & 30 && \text{(nickel)} \\
 & 0.180x_1 + 0.032x_2 + 1.0x_5 &\leq & 35 && \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 &\geq & 10 && \text{(chromium)} && (11.1) \\
 & 0.120x_1 + 0.011x_2 + 1.0x_6 &\leq & 12 && \\
 & 0.001x_2 + 1.0x_7 &\geq & 11 && \text{(molybdenum)} \\
 & 0.001x_2 + 1.0x_7 &\leq & 13 && \\
 & x_1 \leq 75 && && \text{(availability)} \\
 & x_2 \leq 250 && && \\
 & x_1, \dots, x_7 \geq 0 && &&
 \end{aligned}$$

In real application, steel blending often has an added complexity. Some of the scrap elements considered in the blend may be large blocks of reclaimed steel that cannot be subdivided. Either the entire block is used, or none of it is.

ILP Modeling of All-or-Nothing Requirements

Such indivisible input elements illustrate the need to model **all-or-nothing** phenomena. The usual approach to dealing with such cases is to rescale to new discrete variables.

Principle 11.1	All-or-nothing variable requirements of the form
$x_j = 0 \text{ or } u_j$	
can be modeled by substituting $x_j = u_j y_j$, with new discrete variable	
$y_j = 0 \text{ or } 1$	

The new y_j can be interpreted as the fraction of limit u_j chosen.

Swedish Steel Model with All-or-Nothing Constraints

Suppose in our model (11.1) that the first two ingredients had this lumpy character. That is, we may use either none or all 75 kilograms of ingredient 1 and none or all 250 kilograms of ingredient 2. Then, instead of continuous decision variables x_1 and x_2 to reflect the quantities of each ingredient used, we simply employ discrete alternatives

$$y_j \triangleq \begin{cases} 1 & \text{if scrap } j \text{ is part of the blend} \\ 0 & \text{otherwise} \end{cases}$$

In terms of these new variables, the quantity of scraps 1 and 2 included in the mix are $75y_1$ and $250y_2$. Substituting produces the following integer linear program (definition [2.37](#)):

$$\begin{aligned}
 \min \quad & 16(75)y_1 + 10(250)y_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 \\
 \text{s.t.} \quad & 75y_1 + 250y_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 \\
 & 0.0080(75)y_1 + 0.0070(250)y_2 + 0.0085x_3 + 0.0040x_4 \geq 6.5 \\
 & 0.0080(75)y_1 + 0.0070(250)y_2 + 0.0085x_3 + 0.0040x_4 \leq 7.5 \\
 & 0.180(75)y_1 + 0.032(250)y_2 + 1.0x_5 \geq 30 \\
 & 0.180(75)y_1 + 0.032(250)y_2 + 1.0x_5 \leq 35 \\
 & 0.120(75)y_1 + 0.011(250)y_2 + 1.0x_6 \geq 10 \\
 & 0.120(75)y_1 + 0.011(250)y_2 + 1.0x_6 \leq 12 \\
 & 0.001(250)y_2 + 1.0x_7 \geq 11 \\
 & 0.001(250)y_2 + 1.0x_7 \leq 13 \\
 & x_3, \dots, x_7 \geq 0 \\
 & y_1, y_2 = 0 \text{ or } 1
 \end{aligned} \tag{11.2}$$

An optimal solution now sets

$$\begin{aligned}
 y_1^* &= 1, & y_2^* &= 0, & x_3^* &= 736.44, & x_4^* &= 160.06 \\
 x_5^* &= 16.50, & x_6^* &= 1.00, & x_7^* &= 11.00
 \end{aligned}$$

Total cost increases to 9967.1 kroner, versus 9953.7 for the linear programming version (11.1), because of the all-or-nothing requirements.

EXAMPLE 11.1: MODELING ALL-OR-NOTHING VARIABLES

Consider the linear program

$$\begin{aligned}
 \max \quad & 18x_1 + 3x_2 + 9x_3 \\
 \text{s.t.} \quad & 2x_1 + x_2 + 7x_3 \leq 150 \\
 & 0 \leq x_1 \leq 60 \\
 & 0 \leq x_2 \leq 30 \\
 & 0 \leq x_3 \leq 20
 \end{aligned}$$

Revise the model so that each variable can be used only at zero or its upper bound.

Solution: Following principle [11.1](#), we introduce new 0–1 variables

$$y_j \triangleq \text{fraction of upper bound } u_j \text{ used}$$

Then substituting, the model becomes

$$\begin{aligned}
 \max \quad & 18(60y_1) + 3(30y_2) + 9(20y_3) \\
 \text{s.t.} \quad & 2(60y_1) + (30y_2) + 7(20y_3) \leq 150 \\
 & y_1, y_2, y_3 = 0 \text{ or } 1
 \end{aligned}$$

or

$$\begin{aligned} \max \quad & 1080y_1 + 90y_2 + 180y_3 \\ \text{s.t.} \quad & 120y_1 + 30y_2 + 140y_3 \leq 150 \\ & y_1, y_2, y_3 = 0 \text{ or } 1 \end{aligned}$$

ILP Modeling of Fixed Charges

Another common source of lumpy phenomena in what are otherwise linear programs arises when the objective function involves **fixed charges**. For example, a nonnegative decision variable x may have cost

$$\theta(x) \triangleq \begin{cases} f + cx & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

That is, a fixed **initial**, or **construction**, or **setup** cost, f , must be paid before continuous decision variable x can be used at any nonzero level. Thereafter, the usual **variable** cost c of linear programming applies.

If such fixed charges are nonnegative, which is almost always true, they can be modeled in mixed-integer linear programs by using new fixed charge variables.

Principle 11.2 Minimize objective functions with nonnegative fixed charges for making variables $x_j > 0$ can be modeled by introducing new fixed charge variables

$$y_j \triangleq \begin{cases} 1 & \text{if } x_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

The objective coefficient of y_j is the fixed cost of x_j , and the coefficient of x_j is its variable cost.

New switching constraints are also required to link y_j with corresponding x_j .

Definition 11.3 **Switching constraints** model the requirement that continuous variable $x_j \geq 0$ can be used only if a corresponding binary variable $y_j = 1$ by

$$x_j \leq u_j y_j$$

where u_j is a given or derived upper bound on the value of x_j in any feasible solution.

If $y_j = 1$, x_j can assume any LP-feasible value. If $y_j = 0$, then $x_j = 0$ too.

Swedish Steel Application with Fixed Charges

To illustrate the modeling of fixed charges, return again to our original Swedish Steel model (11.1). This time, suppose that there are setup costs. Specifically, assume that ingredients 1 to 4 can be used in the furnace only after injection mechanisms are setup at a cost of 350 kroner each.

To model these fixed charges, we introduce new discrete variables

$$y_j \triangleq \begin{cases} 1 & \text{if setup for } j \text{ is performed} \\ 0 & \text{otherwise} \end{cases}$$

for $j = 1, \dots, 4$.

We also require upper bounds on feasible values of the first four x_j . Bounds $u_1 = 75$ and $u_2 = 250$ are given in model statement (11.1). We must derive corresponding upper bounds for x_3 and x_4 in switching constraints [11.3](#). Any value implied by constraints on these variables is valid, although we will see in Section 12.3 that it helps to employ the smallest possible. For simplicity, we look here only at the first main constraint of model (11.1). Since it sets the total weight of the charge at 1000, we know that $u_3 = u_4 = 1000$ are valid upper bounds.

Introducing these new variables and switching constraints of definition [11.3](#) produces the fixed-charge version of our Swedish Steel model:

$$\begin{aligned} \min \quad & 16x_1 + 10x_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 \\ & + 350y_1 + 350y_2 + 350y_3 + 350y_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 \\ & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 \geq 6.5 \\ & 0.0080x_1 + 0.0070x_2 + 0.0085x_3 + 0.0040x_4 \leq 7.5 \\ & 0.180x_1 + 0.032x_2 + 1.0x_5 \geq 30 \\ & 0.180x_1 + 0.032x_2 + 1.0x_5 \leq 35 \\ & 0.120x_1 + 0.011x_2 + 1.0x_6 \geq 10 \\ & 0.120x_1 + 0.011x_2 + 1.0x_6 \leq 12 \\ & 0.001x_2 + 1.0x_7 \geq 11 \\ & 0.001x_2 + 1.0x_7 \leq 13 \\ & x_1 \leq 75y_1 \\ & x_2 \leq 250y_2 \\ & x_3 \leq 1000y_3 \\ & x_4 \leq 1000y_4 \\ & x_1, \dots, x_7 \geq 0 \\ & y_1, \dots, y_4 = 0 \text{ or } 1 \end{aligned}$$

An optimal solution is

$$\begin{aligned} x_1^* &= 75, & x_2^* &= 0, & x_3^* &= 736.44, & x_4^* &= 160.06 \\ x_5^* &= 16.5, & x_6^* &= 1.00, & x_7^* &= 11.00 \\ y_1^* &= 1, & y_2^* &= 0, & y_3^* &= 1, & y_4^* &= 1 \end{aligned}$$

which sets up for ingredients 1, 3, and 4. Total cost is 11,017.1 kroner, versus 9953.7 in the linear version (11.1).

EXAMPLE 11.2: MODELING FIXED CHARGES

Consider a fixed-charge objective function

$$\min \theta_1(x_1) + \theta_2(x_2)$$

where

$$\theta_1(x_1) \triangleq \begin{cases} 150 + 7x_1 & \text{if } x_1 > 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\theta_2(x_2) \triangleq \begin{cases} 110 + 9x_2 & \text{if } x_2 > 0 \\ 0 & \text{otherwise} \end{cases}$$

For each of the following systems of constraints on x_1 and x_2 , form a corresponding mixed-integer linear programming model.

- (a) $x_1 + x_2 \geq 8$
 $0 \leq x_1 \leq 3$
 $0 \leq x_2 \leq 8$
- (b) $x_1 + x_2 \geq 8$
 $2x_1 + x_2 \leq 10$
 $x_1, x_2 \geq 0$

Solution: We introduce y_1 to carry fixed charges as in [11.2](#) and switching constraints as in [11.3](#).

(a) In this case upper bounds 3 and 8 are provided. Thus the mixed-integer formulation is

$$\begin{aligned} \min \quad & 7x_1 + 9x_2 + 150y_1 + 110y_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 8 \\ & x_1 \leq 3y_1 \\ & x_2 \leq 8y_2 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 = 0 \text{ or } 1 \end{aligned}$$

(b) Upper bounds on x_1 and x_2 are not explicit in these constraints. Still, we may infer from the second main constraint that any feasible solution has $x_1 \leq 5$ and $x_2 \leq 10$. Thus a mixed-integer formulation is

$$\begin{aligned} \min \quad & 7x_1 + 9x_2 + 150y_1 + 110y_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 8 \\ & 2x_1 + x_2 \leq 10 \\ & x_1 \leq 5y_1 \\ & x_2 \leq 10y_2 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 = 0 \text{ or } 1 \end{aligned}$$

11.2 KNAPSACK AND CAPITAL BUDGETING MODELS

In contrast to the cases of Section 11.1, which involve linear programs with some discrete side conditions, **knapsack** and **capital budgeting** problems are completely discrete. We must select an optimal collection of objects, or features, or projects, or investments subject to limits on budget resources. Each element is either all in or all out of the result, with no partial selections allowed.

Knapsack Problems

The simplest of these object choosing discrete problems, and indeed the simplest of all integer linear programs, are knapsack problems.

Definition 11.4 A **knapsack model** is a pure integer linear program with a single main constraint.

All knapsack decision variables are 0–1 in most applications.

The knapsack name derives from the problem confronted by a hiker packing a backpack. He or she must choose the most valuable collection of items to take subject to a volume or weight limit on the size of the pack.

APPLICATION 11.1: INDY CAR KNAPSACK

We may illustrate more realistic forms of knapsack problems by considering the (fictitious) dilemma of mechanics in the Indy Car racing team. Six different features might still be added to this year's car to improve its top speed. Table 11.1 lists their estimated costs and speed enhancements.

Suppose first that Indy Car wants to maximize the performance gain without exceeding a budget of \$35,000. Using decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if feature } j \text{ is added} \\ 0 & \text{otherwise} \end{cases} \quad (11.3)$$

we can formulate the problem as the knapsack model

$$\begin{aligned} \max \quad & 8x_1 + 3x_2 + 15x_3 + 7x_4 + 10x_5 + 12x_6 && \text{(mph gain)} \\ \text{s.t.} \quad & 10.2x_1 + 6.0x_2 + 23.0x_3 + 11.1x_4 + 9.8x_5 + 31.6x_6 \leq 35 && \text{(budget)} \quad (11.4) \\ & x_1, \dots, x_6 = 0 \text{ or } 1 \end{aligned}$$

That is, we maximize total performance subject to a budget constraint. An optimal solution chooses features 1, 4, and 5 for a gain of 25 miles per hour.

Suppose now that the Indy Car team decides they simply must increase speed by 30 miles per hour to have any chance of winning the next race. Ignoring the budget, they wish to find the minimum cost way to achieve at least that much performance.

TABLE 11.1 Indy Car Application Alternatives

	Proposed Feature, j					
	1	2	3	4	5	6
Cost (\$ 000's)	10.2	6.0	23.0	11.1	9.8	31.6
Speed increase (mph)	8	3	15	7	10	12

This scenario leads to an alternative, minimize knapsack form. With variables (11.3), we obtain

$$\begin{aligned} \min \quad & 10.2x_1 + 6.0x_2 + 23.0x_3 + 11.1x_4 + 9.8x_5 + 31.6x_6 \quad (\text{cost}) \\ \text{s.t.} \quad & 8x_1 + 3x_2 + 15x_3 + 7x_4 + 10x_5 + 12x_6 \geq 30 \quad (\text{mph required}) \quad (11.5) \\ & x_1, \dots, x_6 = 0 \text{ or } 1 \end{aligned}$$

This model minimizes cost subject to a performance requirement. An optimal solution now chooses features 1, 3, and 5 at cost \$43,000.

EXAMPLE 11.3: FORMULATING KNAPSACK MODELS

Readily available U.S. coins are denominated 1, 5, 10, and 25 cents. Formulate a knapsack model to minimize the number of coins needed to provide change amount q cents.

Solution: We employ decision variables x_1, x_5, x_{10} , and x_{25} to represent the number of coins chosen from each denomination. Then the knapsack model is

$$\begin{aligned} \min \quad & x_1 + x_5 + x_{10} + x_{25} \quad (\text{total coins}) \\ \text{s.t.} \quad & x_1 + 5x_5 + 10x_{10} + 25x_{25} = q \quad (\text{correct change}) \\ & x_1, x_5, x_{10}, x_{25} \geq \text{and integer} \end{aligned}$$

Notice that these discrete variables are not limited to 0 and 1.

Capital Budgeting Models

The typical maximize form of a knapsack problem has its single main constraint enforcing a budget. When there are budget limits over more than one time period, or multiple limited resources, we obtain more general capital budgeting or multidimensional knapsack models.

Definition 11.5 **Capital budgeting models** (or **multidimensional knapsacks**) select a maximum value collection of project, investments, and so on, subject to limitations on budgets or other resources consumed.

APPLICATION 11.2: NASA CAPITAL BUDGETING

The U.S. space agency, NASA, must deal constantly with such decision problems in choosing how to divide its limited budgets among many competing missions proposed.¹ Table 11.2 shows a fictitious list of alternatives.

We must decide which of the 14 indicated missions to include in program plans over 5 stages of a 25-year era. Thus it should be clear that the needed decision variables are

$$x_j \triangleq \begin{cases} 1 & \text{if mission } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad (11.6)$$

¹Based on G. W. Evans and R. Fairbairn (1989), "Selection and Scheduling of Advanced Missions for NASA Using 0–1 Integer Linear Programming," *Journal of the Operational Research Society*, 40, 971–981.

TABLE 11.2 Proposed Missions in NASA Application

<i>j</i>	Mission	Budget Requirements (\$ billion)					Value	Not With	Depends On
		Stage 1	Stage 2	Stage 3	Stage 4	Stage 5			
1	Communications satellite	6	—	—	—	—	200	—	—
2	Orbital microwave	2	3	—	—	—	3	—	—
3	Io lander	3	5	—	—	—	20	—	—
4	Uranus orbiter 2020	—	—	—	—	10	50	5	3
5	Uranus orbiter 2010	—	5	8	—	—	70	4	3
6	Mercury probe	—	—	1	8	4	20	—	3
7	Saturn probe	1	8	—	—	—	5	—	3
8	Infrared imaging	—	—	—	5	—	10	11	—
9	Ground-based SETI	4	5	—	—	—	200	14	—
10	Large orbital structures	—	8	4	—	—	150	—	—
11	Color imaging	—	—	2	7	—	18	8	2
12	Medical technology	5	7	—	—	—	8	—	—
13	Polar orbital platform	—	1	4	1	1	300	—	—
14	Geosynchronous SETI	—	4	5	3	3	185	9	—
	Budget	10	12	14	14	14			

Budget Constraints

The budget constraints that give capital budgeting problems their name limit project expenditures in particular time periods.

Definition 11.6 **Budget constraints** limit the total funds or other resources consumed by selected projects, investments, and so on, in each time period not to exceed the amount available.

Budget requirements in Table 11.2 span 5-year stages. We form budget constraints for each of these periods by summing project decision variables times their needs.

$$6x_1 + 2x_2 + 3x_3 + 1x_7 + 4x_9 + 5x_{12} \leq 10 \quad (\text{Stage 1})$$

$$3x_2 + 5x_3 + 5x_5 + 8x_7 + 5x_9 + 8x_{10} \leq 12 \quad (\text{Stage 2})$$

$$+ 7x_{12} + 1x_{13} + 4x_{14} \leq 12$$

$$8x_5 + 1x_6 + 4x_{10} + 2x_{11} + 4x_{13} + 5x_{14} \leq 14 \quad (\text{Stage 3})$$

$$8x_6 + 5x_8 + 7x_{11} + 1x_{13} + 3x_{14} \leq 14 \quad (\text{Stage 4})$$

$$10x_4 + 4x_6 + 1x_{13} + 3x_{14} \leq 14 \quad (\text{Stage 5})$$

EXAMPLE 11.4: FORMULATING BUDGET CONSTRAINTS

A department store is considering 4 possible expansions into presently unoccupied space in a shopping mall. The following table shows how much (in millions of dollars) each expansion would cost in the next two fiscal years, and the required floor space (in thousands of square feet).

	Expansion, j			
	1	2	3	4
Year 1	1.5	5.0	7.3	1.9
Year 2	3.5	1.8	6.0	4.2
Space	2.2	9.1	5.3	8.6

Using decision variables

$$x_j \triangleq \begin{cases} 1 & \text{expansion } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

formulate implied constraints on investment funds and floor space assuming that 10 million dollars are available in each of the two years and that the expansion cannot exceed 17 thousand square feet.

Solution: The three required budget constraints are

$$\begin{aligned} 1.5x_1 + 5.0x_2 + 7.3x_3 + 1.9x_4 &\leq 10 && \text{(year 1 budget)} \\ 3.5x_1 + 1.8x_2 + 6.0x_3 + 4.2x_4 &\leq 10 && \text{(year 2 budget)} \\ 2.2x_1 + 9.1x_2 + 5.3x_3 + 8.6x_4 &\leq 17 && \text{(floor space)} \end{aligned}$$

Modeling Mutually Exclusive Choices

Capital budgeting problems often come with other constraints besides simple budget limits. For example, two or more proposed projects may be **mutually exclusive**. That is, at most one of them can be included in a solution.

Table 11.2 indicates three such conflicts. Possibilities $j = 4$ and 5 represent alternative timing for the same mission. Technologies for missions $j = 8$ and 11 are incompatible. Numbers $j = 9$ and 14 involve two different ways to accomplish the SETI program.

Such incompatibilities are easily modeled with 0–1 decision variables (11.6).

Definition 11.7 **Mutually exclusiveness** conditions allowing at most one of a set of choices are modeled by $\sum x_j \leq 1$ constraints summing over each choice set.

For our NASA application, the result is

$$\begin{aligned} x_4 + x_5 &\leq 1 \\ x_8 + x_{11} &\leq 1 \\ x_9 + x_{14} &\leq 1 \end{aligned}$$

EXAMPLE 11.5: FORMULATING MUTUALLY EXCLUSIVE CONSTRAINTS

Suppose that a real estate development company is considering 5 investment decisions. Only 1 of the first 3 can be chosen because all require the same piece of land. Investment 4 is a new office building, and investment 5 is the same project delayed a year. Formulate suitable mutually exclusive constraints in terms of decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if investment } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Solution: The fact that only 1 of the first 3 can be selected produces constraint

$$x_1 + x_2 + x_3 \leq 1$$

Since x_4 and x_5 refer to the same investment, we also want

$$x_4 + x_5 \leq 1$$

Modeling Dependencies between Projects

Another characteristic relationship between projects arises when one project depends on another. We cannot choose such a dependent project unless we also include the option on which it depends.

We can model dependencies among projects as easily as mutual exclusiveness.

Principle 11.8 | **Dependence** of choice j on choice i can be enforced on corresponding binary variables by constraint $x_j \leq x_i$.

Variable x_j cannot = 1 unless x_i does too.

Table 11.2 shows that mission $j = 11$ depends on mission 2 in our NASA application, and also that mission $j = 3$ must be chosen if any of projects 4, . . . , 7 is. Implied constraints are

$$x_{11} \leq x_2$$

$$x_4 \leq x_3$$

$$x_5 \leq x_3$$

$$x_6 \leq x_3$$

$$x_7 \leq x_3$$

EXAMPLE 11.6: MODELING PROJECT DEPENDENCIES

Phase 1 of a new city hall project will construct the first story of a building that could grow to two stories in phase 2, and to three in phase 3. Using the variables

$$x_j \triangleq \begin{cases} 1 & \text{if phase } j \text{ is constructed} \\ 0 & \text{otherwise} \end{cases}$$

write corresponding dependency constraints.

Solution: Obviously, the second floor cannot be built without the first, and the third without the second. Thus we have project dependency constraints

$$x_2 \leq x_1$$

$$x_3 \leq x_2$$

NASA Application Model

To complete formulation of our version of NASA's decision problem, we need an objective function. Like almost all public agencies, NASA has many. They may try to maximize the intellectual gains of selected missions, maximize the direct benefit to life on earth, and so on.

We will assume that a weighted sum of these different objective functions can be used to estimate a value for each mission. Resulting values are included in Table 11.2. Combining with previous elements, we obtain the NASA application model:

$$\begin{aligned}
 \max \quad & 200x_1 + 3x_2 + 20x_3 + 50x_4 + 70x_5 && \text{(total value)} \\
 & + 20x_6 + 5x_7 + 10x_8 + 200x_9 + 150x_{10} \\
 & + 18x_{11} + 8x_{12} + 300x_{13} + 185x_{14} \\
 \text{s.t.} \quad & 6x_1 + 2x_2 + 3x_3 + 1x_7 + 4x_9 + 5x_{12} \leq 10 && \text{(Stage 1)} \\
 & 3x_2 + 5x_3 + 5x_5 + 8x_7 + 5x_9 + 8x_{10} && \text{(Stage 2)} \\
 & \quad + 7x_{12} + 1x_{13} + 4x_{14} \leq 12 \\
 & 8x_5 + 1x_6 + 4x_{10} + 2x_{11} + 4x_{13} + 5x_{14} \leq 14 && \text{(Stage 3)} \\
 & 8x_6 + 5x_8 + 7x_{11} + 1x_{13} + 3x_{14} \leq 14 && \text{(Stage 4)} \\
 & 10x_4 + 4x_6 + 1x_{13} + 3x_{14} \leq 14 && \text{(Stage 5)} \\
 & x_4 + x_5 \leq 1 && \text{(mutually} \\
 & x_8 + x_{11} \leq 1 && \text{exclusive)} \\
 & x_9 + x_{14} \leq 1 \\
 & x_{11} \leq x_2 && \text{(dependent} \\
 & x_4 \leq x_3 && \text{missions)} \\
 & x_5 \leq x_3 \\
 & x_6 \leq x_3 \\
 & x_7 \leq x_3 \\
 & x_j = 0 \text{ or } 1 \quad \text{for all } j = 1, \dots, 14
 \end{aligned} \tag{11.7}$$

11.3 SET PACKING, COVERING, AND PARTITIONING MODELS

Our capital budget models of Section 11.2 included mutual exclusiveness constraints involving subsets of decision variables, at most one of which can take part in a solution. **Set packing, covering, and partitioning models** feature such constraints. Using decision variables that = 1 if an object is part of a solution and = 0 otherwise, these models formulate problems where the core issue is membership in specified subsets.

APPLICATION 11.3: EMS LOCATION PLANNING

As always, it will help to think of a specific application. A classic example occurred when Austin, Texas undertook a study of the positioning of its emergency medical service (EMS) vehicles.² That city was divided into service districts needing EMS services, and vehicle stations selected from a list of alternatives so that as much of the population as possible would experience a quick response to calls for help.

Figure 11.1 shows the fictitious map we will assume for our numerical version. Our city is divided into 20 service districts that we wish to serve from some

²Based on D. J. Eaton, M. S. Daskin, D. Simmons, Bill Bulloch, and G. Jansma (1985), “Determining Emergency Medical Service Vehicle Deployment in Austin, Texas,” *Interfaces*, 15:1, 96–108.

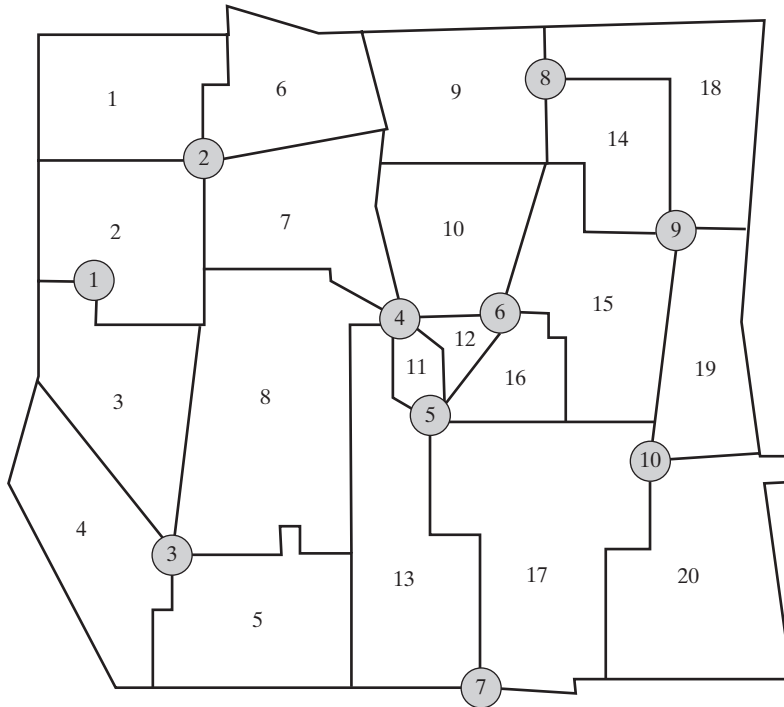


FIGURE 11.1 Service District and Candidate Locations for EMS Application

combination of the 10 indicated possibilities for EMS stations. Each station can provide service to all adjacent districts. For example, station 2 could service districts 1, 2, 6, and 7. Main decision variables are

$$x_j \triangleq \begin{cases} 1 & \text{if location } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Set Packing, Covering, and Partitioning Constraints

The defining constraints of set packing, covering, and partitioning models deal with subcollections of problem objects. In our EMS application the subcollections are locations that can provide satisfactory response for a service district. For instance, Figure 11.1 shows that any of locations 4, 5, and 6 can protect downtown district 12.

Covering constraints demand that at least one member of each subcollection belongs to a solution, packing constraints allow at most one member, and partitioning constraints require exactly one member. Mathematical forms follow easily with 0–1 variables.

Definition 11.9 **Set covering constraints** requiring that at least one member of subcollection J belongs to a solution are expressed as

$$\sum_{j \in J} x_j \geq 1$$

Definition 11.10 | **Set packing constraints** requiring that at most one member of subcollection J belongs to a solution are expressed as

$$\sum_{j \in J} x_j \leq 1$$

Definition 11.11 | **Set partitioning constraints** requiring that exactly one member of subcollection J belongs to a solution are expressed as

$$\sum_{j \in J} x_j = 1$$

In our EMS application, we are covering, because each district should be protected by at least one location, but more is even better. In other settings, such as radio station location, we pack; at most one station on any particular frequency should reach a service area (see Exercise 11-7). The mutual exclusiveness constraints of capital budgeting also have the packing form. Partitioning applies when exactly one decision option can serve each sector. This is the case, for example, if decision variables relate to possible election districts, and constraints demand that each geographic unit belong to exactly one district (see Exercise 11-12).

EXAMPLE 11.7: FORMULATING SET PACKING, COVERING, AND PARTITIONING

A university is acquiring mathematical programming software for use in operations research classes. The four codes available and the types of optimization algorithms they provide are indicated by ×'s in the following table.

Algorithm Type	Code, j			
	1	2	3	4
LP	×	×	×	×
IP	—	×	—	×
NLP	—	—	×	×
Objective	3	4	6	14

- Taking objective function coefficients as code costs, formulate a set covering model to acquire a minimum cost collection of codes providing LP, IP, and NLP capability.
- Taking objective function coefficients as code costs, formulate a set partitioning model to acquire a minimum cost collection of codes with exactly one providing LP, one providing IP, and one providing NLP.
- Taking objective function coefficients as indications of code quality, formulate a set packing model to acquire a maximum quality collection of codes with at most one providing LP, at most one providing IP, and at most one providing NLP.

Solution: In each case we use the decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if code } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

(a) Following form [11.9](#), the required model is

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 + 6x_3 + 14x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \geq 1 && \text{(LP)} \\ & x_2 + x_4 \geq 1 && \text{(IP)} \\ & x_3 + x_4 \geq 1 && \text{(NLP)} \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

(b) Following form [11.11](#), the required model is

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 + 6x_3 + 14x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 1 && \text{(LP)} \\ & x_2 + x_4 = 1 && \text{(IP)} \\ & x_3 + x_4 = 1 && \text{(NLP)} \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

(c) Following form [11.10](#), the required model is

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 + 6x_3 + 14x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \leq 1 && \text{(LP)} \\ & x_2 + x_4 \leq 1 && \text{(IP)} \\ & x_3 + x_4 \leq 1 && \text{(NLP)} \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

Minimum Cover EMS Model

The most obvious approach to modeling our EMS application of Figure 11.1 is to minimize the number of locations needed to cover all districts. The following set covering model results.

$$\begin{aligned} \min \quad & \sum_{j=1}^{10} x_j && \text{(number of sites)} \\ \text{s.t.} \quad & x_2 \geq 1 && \text{(district 1)} \\ & x_1 + x_2 \geq 1 && \text{(district 2)} \\ & x_1 + x_3 \geq 1 && \text{(district 3)} \\ & x_3 \geq 1 && \text{(district 4)} \\ & x_3 \geq 1 && \text{(district 5)} \\ & x_2 \geq 1 && \text{(district 6)} \\ & x_2 + x_4 \geq 1 && \text{(district 7)} \end{aligned} \tag{11.8}$$

$$\begin{aligned}
x_3 + x_4 &\geq 1 && \text{(district 8)} \\
x_8 &\geq 1 && \text{(district 9)} \\
x_4 + x_6 &\geq 1 && \text{(district 10)} \\
x_4 + x_5 &\geq 1 && \text{(district 11)} \\
x_4 + x_5 + x_6 &\geq 1 && \text{(district 12)} \\
x_4 + x_5 + x_7 &\geq 1 && \text{(district 13)} \\
x_8 + x_9 &\geq 1 && \text{(district 14)} \\
x_6 + x_9 &\geq 1 && \text{(district 15)} \\
x_5 + x_6 &\geq 1 && \text{(district 16)} \\
x_5 + x_7 + x_{10} &\geq 1 && \text{(district 17)} \\
x_8 + x_9 &\geq 1 && \text{(district 18)} \\
x_9 + x_{10} &\geq 1 && \text{(district 19)} \\
x_{10} &\geq 1 && \text{(district 20)} \\
x_1, \dots, x_{10} &= 0 \text{ or } 1
\end{aligned}$$

One optimal solution chooses the six sites 2, 3, 4, 6, 8, and 10. That is,

$$\begin{aligned}
x_2^* = x_3^* = x_4^* = x_6^* = x_8^* = x_{10}^* &= 1 \\
x_1^* = x_5^* = x_7^* = x_9^* &= 0
\end{aligned}$$

Maximum Coverage EMS Model

In the Austin case, as in many other real instances, the straightforward covering model (11.8) proves inadequate because it calls for too many sites. Suppose that we have funds for only 4 EMS locations. How can we find the collection of 4 that minimizes coverage insufficiency?

For this version of the model we need estimates of the demand or importance of covering each service district. We will assume the following values have been estimated by EMS staff:

District <i>i</i>	Value	District <i>i</i>	Value	District <i>i</i>	Value
1	5.2	8	12.2	15	15.5
2	4.4	9	7.6	16	25.6
3	7.1	10	20.3	17	11.0
4	9.0	11	30.4	18	5.3
5	6.1	12	30.9	19	7.9
6	5.7	13	12.0	20	9.9
7	10.0	14	9.3		

Next we introduce extra decision variables to model uncovered districts i .

Principle 11.12 Set packing, covering, and partitioning models can be modified to penalize uncovered items i by introducing new variables

$$y_i \triangleq \begin{cases} 1 & \text{if item } i \text{ is uncovered in the solution} \\ 0 & \text{otherwise} \end{cases}$$

into each constraint i .

Including such variables, the EMS model becomes

$$\begin{aligned}
 \min \quad & 5.2y_1 + 4.4y_2 + 7.1y_3 + 9.0y_4 + 6.1y_5 && \text{(uncovered} \\
 & + 5.7y_6 + 10.0y_7 + 12.2y_8 + 7.6y_9 + 20.3y_{10} && \text{district} \\
 & + 30.4y_{11} + 30.9y_{12} + 12.0y_{13} + 9.3y_{14} + 15.5y_{15} && \text{importance)} \\
 & + 25.6y_{16} + 11.0y_{17} + 5.3y_{18} + 7.9y_{19} + 9.9y_{20} \\
 \text{s.t.} \quad & x_2 + y_1 && \geq 1 && \text{(district 1)} \\
 & x_1 + x_2 + y_2 && \geq 1 && \text{(district 2)} \\
 & x_1 + x_3 + y_3 && \geq 1 && \text{(district 3)} \\
 & x_3 + y_4 && \geq 1 && \text{(district 4)} \\
 & x_3 + y_5 && \geq 1 && \text{(district 5)} \\
 & x_2 + y_6 && \geq 1 && \text{(district 6)} \\
 & x_2 + x_4 + y_7 && \geq 1 && \text{(district 7)} \\
 & x_3 + x_4 + y_8 && \geq 1 && \text{(district 8)} \\
 & x_8 + y_9 && \geq 1 && \text{(district 9)} \\
 & x_4 + x_6 + y_{10} && \geq 1 && \text{(district 10)} \\
 & x_4 + x_5 + y_{11} && \geq 1 && \text{(district 11)} \\
 & x_4 + x_5 + x_6 + y_{12} && \geq 1 && \text{(district 12)} \\
 & x_4 + x_5 + x_7 + y_{13} && \geq 1 && \text{(district 13)} \\
 & x_8 + x_9 + y_{14} && \geq 1 && \text{(district 14)} \\
 & x_6 + x_9 + y_{15} && \geq 1 && \text{(district 15)} \\
 & x_5 + x_6 + y_{16} && \geq 1 && \text{(district 16)} \\
 & x_5 + x_7 + x_{10} + y_{17} && \geq 1 && \text{(district 17)} \\
 & x_8 + x_9 + y_{18} && \geq 1 && \text{(district 18)} \\
 & x_9 + x_{10} + y_{19} && \geq 1 && \text{(district 19)} \\
 & x_{10} + y_{20} && \geq 1 && \text{(district 20)} \\
 & \sum_{j=1}^{10} x_j && \leq 4 && \text{(at most four)} \\
 & x_1, \dots, x_{10} = 0 \text{ or } 1 \\
 & y_1, \dots, y_{20} = 0 \text{ or } 1
 \end{aligned} \tag{11.9}$$

Here the objective function minimizes the total importance of uncovered districts. The last (noncovering) main constraint limits solutions to four EMS sites.

An optimal solution for this more realistic version chooses

$$\begin{aligned}x_3^* &= x_4^* = x_5^* = x_9^* = 1 \\y_1^* &= y_2^* = y_6^* = y_9^* = y_{20}^* = 1\end{aligned}$$

with all other decision variables = 0. That is, sites 3, 4, 5, and 9 are chosen, leaving districts 1, 2, 6, 9, and 20 uncovered. The total importance of those districts, which is the optimal objective value, equals 32.8.

EXAMPLE 11.8: FORMULATING MAXIMUM COVERING MODELS

Return to the set covering case of Example 11.7(a). Revise the model to maximize the number of algorithms available within a budget of 12.

Solution: We introduce new variables y_{LP} , y_{IP} , and y_{NLP} that = 1 if the indicated algorithm is not provided, and = 0 otherwise. Then the required model is

$$\begin{aligned}\min \quad & y_{LP} + y_{IP} + y_{NLP} \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 + y_{LP} \geq 1 \quad (\text{LP}) \\ & x_2 + x_4 + y_{IP} \geq 1 \quad (\text{IP}) \\ & x_3 + x_4 + y_{NLP} \geq 1 \quad (\text{NLP}) \\ & 3x_1 + 4x_2 + 6x_3 + 14x_4 \leq 12 \quad (\text{budget}) \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \\ & y_{LP}, y_{IP}, y_{NLP} = 0 \text{ or } 1\end{aligned}$$

Column Generation Models

Another common family of set packing, covering, and partitioning models are derived from problems involving a combinatorially large array of possibilities too complex to be modeled concisely. Column generation adopts a two-part strategy for such problems.

Definition 11.13 **Column generation** approaches deal with complex combinatorial problems by first enumerating a sequence of columns representing viable solutions to parts of the problem, and then solving a set partitioning (or covering or packing) model to select an optimal collection of these alternatives fulfilling all problem requirements. (See also Section 13.1).

The convenience of this approach comes from its flexibility. Any appropriate scheme—however ad hoc—can be employed to generate a rich family of columns meeting complex and difficult-to-model constraints. Optimization is reserved for the second part of the strategy, when a well-formed model over the columns can be addressed by standard ILP technology.

APPLICATION 11.4: AA CREW SCHEDULING

A classic application of column generation arises in the enormously complex problem of scheduling crews for airlines. For example, American Airlines³ reports spending over \$1.3 billion per year on salaries, benefits, and travel expenses of air crews. Careful scheduling, or **crew pairing** as it is called, can produce enormous savings.

Figure 11.2 illustrates a (tiny) sequence of flights to be crewed in our fictitious case. For example, flight 101 originates in Miami and arrives in Chicago some hours later.

Each pairing is a sequence of flights to be covered by a single crew over a 2- to 3-day period. It must begin and end in the base city where the crew resides.

Table 11.3 enumerates possible pairings of flights in Figure 11.2. For example, pairing $j = 1$ begins at Miami with flight 101. After a layover in Chicago, the crew covers flight 203 to Dallas–Ft. Worth and then flight 406 to Charlotte. Finally, flight 308 returns them to Miami.

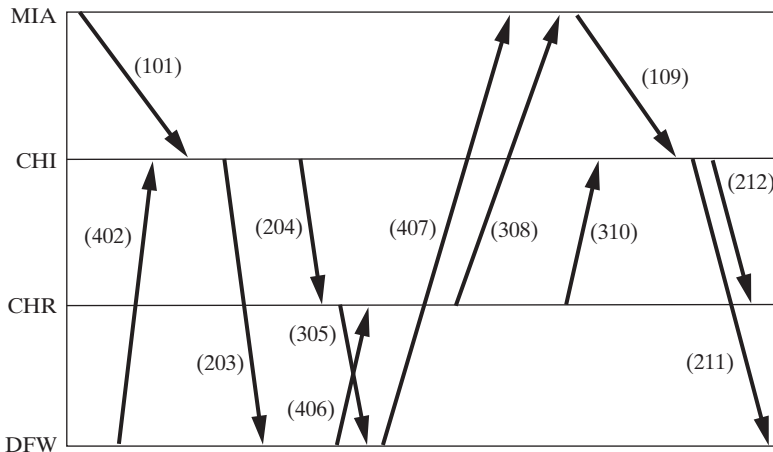


FIGURE 11.2 Flight Schedule for AA Application

TABLE 11.3 Possible Pairings for AA Application

j	Flight Sequence	Cost	j	Flight Sequence	Cost
1	101–203–406–308	2900	9	305–407–109–212	2600
2	101–203–407	2700	10	308–109–212	2050
3	101–204–305–407	2600	11	402–204–305	2400
4	101–204–308	3000	12	402–204–310–211	3600
5	203–406–310	2600	13	406–308–109–211	2550
6	203–407–109	3150	14	406–310–211	2650
7	204–305–407–109	2550	15	407–109–211	2350
8	204–308–109	2500			

³Based on R. Anbil, E. Gelman, B. Patty, and R. Tanga (1991), “Recent Advances in Crew-Pairing Optimization at American Airlines,” *Interfaces*, 21:1, 62–74.

In real applications, intricate government and union rules regulate exactly which sequences of flights constitute a reasonable pairing. Complex software is employed to generate a list such as that of Table 11.3. Here we simply allow all closed sequences of 3 or 4 flights in Figure 11.2. (See Section 13.1 for more discussion).

Column Generation Model for AA Application

Having enumerated a list of alternatives like Table 11.3, the remaining task is to find a minimum total cost collection of columns staffing each flight exactly once. Define decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if pairing } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

Then the following set partitioning model does the job:

$$\begin{aligned} \min \quad & 2900x_1 + 2700x_2 + 2600x_3 + 3000x_4 + 2600x_5 \\ & + 3150x_6 + 2550x_7 + 2500x_8 + 2600x_9 + 2050x_{10} \\ & + 2400x_{11} + 3600x_{12} + 2550x_{13} + 2650x_{14} + 2350x_{15} \quad (11.10) \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 1 \quad (\text{flight 101}) \\ & x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{15} = 1 \quad (\text{flight 109}) \\ & x_1 + x_2 + x_5 + x_6 = 1 \quad (\text{flight 203}) \\ & x_3 + x_4 + x_7 + x_8 + x_{11} + x_{12} = 1 \quad (\text{flight 204}) \\ & x_{12} + x_{13} + x_{14} + x_{15} = 1 \quad (\text{flight 211}) \\ & x_9 + x_{10} = 1 \quad (\text{flight 212}) \\ & x_3 + x_7 + x_9 + x_{11} = 1 \quad (\text{flight 305}) \\ & x_1 + x_4 + x_8 + x_{10} + x_{13} = 1 \quad (\text{flight 308}) \\ & x_5 + x_{12} + x_{14} = 1 \quad (\text{flight 310}) \\ & x_{11} + x_{12} = 1 \quad (\text{flight 402}) \\ & x_1 + x_5 + x_{13} + x_{14} = 1 \quad (\text{flight 406}) \\ & x_2 + x_3 + x_6 + x_7 + x_9 + x_{15} = 1 \quad (\text{flight 407}) \\ & x_1, \dots, x_{15} = 0 \text{ or } 1 \end{aligned}$$

An optimal solution makes

$$x_1^* = x_9^* = x_{12}^* = 1$$

and all other $x_j^* = 0$ at total cost \$9100.

Pairing costs are equally complex. On-duty crews are guaranteed pay for minimum periods, regardless of the part of that time they are actually flying. If the pairing requires overnight stays away from home, hotel and other expenses are added. Values for our illustration are shown in Table 11.3.

EXAMPLE 11.9: FORMING COLUMN GENERATION MODELS

A moving and storage company is allocating 5 long-distance moving loads to trucks. One feasible combination covers loads 1 and 3 in a 4525-mile route; a second combines loads 2, 3, and 4 in 2960 miles; a third hauls loads 2, 4, and 5 in 3170 miles; and a fourth covers load 1, 4, and 5 in 5230 miles. Form a set partitioning model to decide a minimum distance combination of routes covering each load exactly once.

Solution: We use the decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if route } j \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

Then the required model is

$$\begin{array}{llll} \min & 4525x_1 + 2960x_2 + 3170x_3 + 5230x_4 & & \text{(total miles)} \\ \text{s.t.} & x_1 + x_4 & = & 1 \quad \text{(load 1)} \\ & x_2 + x_3 & = & 1 \quad \text{(load 2)} \\ & x_1 + x_2 & = & 1 \quad \text{(load 3)} \\ & x_2 + x_3 + x_4 & = & 1 \quad \text{(load 4)} \\ & x_3 + x_4 & = & 1 \quad \text{(load 5)} \\ & x_1, \dots, x_4 & = & 0 \text{ or } 1 \end{array}$$

11.4 ASSIGNMENT AND MATCHING MODELS

We have already encountered **assignment problems** in network flow Section 10.6. The issue is optimal matching or pairing of objects of two distinct types—jobs to machines, sales personnel to customers, and so on. In this section we extend with a number of variations that cannot be solved by network flow methods.

Assignment Constraints

It is standard to model all assignment forms with the decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if } i \text{ of the first set is matched with } j \text{ of the second} \\ 0 & \text{otherwise} \end{cases}$$

Then corresponding assignment constraints merely require that each object of each set be paired exactly once.

Definition 11.14 Over decision variables $x_{i,j} = 1$ if i is assigned to j and $= 0$ otherwise, **assignment constraints** take the form

$$\begin{aligned} \sum_j x_{i,j} &= 1 && \text{for all } i \\ \sum_i x_{i,j} &= 1 && \text{for all } j \\ x_{i,j} &= 0 \text{ or } 1 && \text{for all } i, j \end{aligned}$$

where all sums are limited to (i, j) combinations allowed in the instance.

The first system of constraints forces every i to be assigned. The second does the same for every j .

CAM Linear Assignment Application Revisited

Section 10.6's computer-aided manufacturing (CAM) model illustrates one assignment case. Table 11.4 repeats total transportation, queueing, and processing times for 8 pending jobs on 10 workstations to which they might next be routed. Each workstation can accommodate only one job at a time. We want to find a minimum total time routing.

TABLE 11.4 Transportation and Processing Times for CAM Application

Job, i	Next Workstation, j									
	1	2	3	4	5	6	7	8	9	10
1	8	—	23	—	—	—	—	—	5	—
2	—	4	—	12	15	—	—	—	—	—
3	—	—	20	—	13	6	—	8	—	—
4	—	—	—	—	19	10	—	—	—	—
5	—	—	—	8	—	—	12	—	—	16
6	14	—	—	—	—	—	8	—	3	—
7	—	6	—	—	—	—	—	27	—	12
8	—	5	15	—	—	—	—	32	—	—

After introducing dummy jobs 9 and 10, so that we have the same number of jobs as machines, this problem clearly has the assignment form. A complete model is given in (10.5) of Section 10.6. Our interest here is in the objective function

$$\begin{aligned}
 \min \quad & 8x_{1,1} + 23x_{1,3} + 5x_{1,9} + 4x_{2,2} + 12x_{2,4} + 15x_{2,5} \\
 & + 20x_{3,3} + 13x_{3,5} + 6x_{3,6} + 8x_{3,8} + 19x_{4,5} + 10x_{4,6} \\
 & + 8x_{5,4} + 12x_{5,7} + 16x_{5,10} + 14x_{6,1} + 8x_{6,7} + 3x_{6,9} \\
 & + 6x_{7,2} + 27x_{7,8} + 12x_{7,10} + 5x_{8,2} + 15x_{8,3} + 32x_{8,8}
 \end{aligned} \tag{11.11}$$

Linear Assignment Models

CAM model (10.5) is a linear assignment model because its objective function (11.11) is linear.

Definition 11.15 **Linear assignment models** minimize or maximize a linear objective function of the form

$$\sum_i \sum_j c_{i,j} x_{i,j}$$

subject to assignment constraints [11.14], where $c_{i,j}$ is the cost (or benefit) of assigning i to j .

Costs are sums of single assignment decisions.

EXAMPLE 11.10: FORMULATING LINEAR ASSIGNMENT MODELS

A swimming coach is choosing his team for a medley relay. One swimmer will swim the back stroke leg $j = 1$, one the breast stroke leg $j = 2$, one the butterfly leg $j = 3$, and one the free-style leg $j = 4$. From previous experience the coach can estimate the time, $t_{i,j}$, that swimmer i could achieve on leg j . Formulate a linear assignment model to choose the fastest medley team.

Solution: Using the decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if swimmer } i \text{ swims leg } j \\ 0 & \text{otherwise} \end{cases}$$

the required model is

$$\begin{aligned} \min \quad & \sum_{i=1}^4 \sum_{j=1}^4 t_{i,j} x_{i,j} && \text{(team time)} \\ \text{s.t.} \quad & \sum_{j=1}^4 x_{i,j} = 1 && i = 1, \dots, 4 \quad \text{(one leg per swimmer)} \\ & \sum_{i=1}^4 x_{i,j} = 1 && j = 1, \dots, 4 \quad \text{(one swimmer per leg)} \\ & x_{i,j} = 0 \text{ or } 1 && i = 1, \dots, 4; \quad j = 1, \dots, 4 \end{aligned}$$

Swimmers are assigned to relay legs to minimize total team time.

Quadratic Assignment Models

What produces the linearity in definition [11.15](#) is that the objective function weights single decisions to pair i in one set with j in the other. For example, in objective (11.11), a decision to assign job 4 to workstation 5 adds 19 to the total time of the solution, regardless of how other decisions are resolved.

Many assignment problem circumstances do not fit the linear case because the objective function depends on combinations of decisions. That is, the impact of one decision cannot be assessed until we know how others are resolved. Such circumstances often lead to quadratic assignment models.

Definition 11.16 **Quadratic assignment models** minimize or maximize a quadratic objective function of the form

$$\sum_i \sum_j \sum_{k>i} \sum_{\ell \neq j} c_{i,j,k,\ell} x_{i,j} x_{k,\ell}$$

subject to assignment constraints [11.14](#), where $c_{i,j,k,\ell}$ is the cost (or benefit) of assigning i to j and k to ℓ .

Notice that each objective function term

$$c_{i,j,k,\ell} \cdot x_{i,j} \cdot x_{k,\ell}$$

involves two assignment decisions. Cost $c_{i,j,k,\ell}$ is realized only if both $x_{i,j} = 1$ and $x_{k,\ell} = 1$. That is, $c_{i,j,k,\ell}$ applies only if i is assigned to j and k is assigned to ℓ .

APPLICATION 11.5: MALL LAYOUT QUADRATIC ASSIGNMENT

Some of the most common cases producing quadratic assignment models arise in **facility layout**. We are given a collection of machines, offices, departments, stores, and so on, to arrange within a facility, and a set of locations within which they must fit. The problem is to decide which unit to assign to each location.

Figure 11.3 illustrates with 4 possible locations for stores in a shopping mall. Walking distances (in feet) between the shop locations are displayed in the adjacent table. The 4 prospective tenants for the shop locations are listed in Table 11.5. The table also shows the number of customers each week (in thousands) who might wish to visit various pairs of shops. For example, a projected 5 thousand customers per week will visit both 1 (Clothes Are) and 2 (Computers Aye).

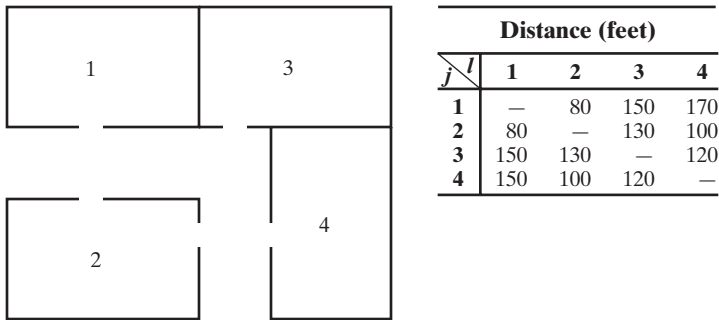


FIGURE 11.3 Mall Layout Application Locations

Mall managers want to arrange the stores in the 4 locations to minimize customer inconvenience. One very common measure is **flow-distance**, the product of flow volumes between facilities and the distances between their assigned locations. For example, if shop 1 (Clothes Are) is located in space 1, and shop 4 (Book Bazaar) is located in space 2, their 7 thousand common customers will have to walk the 80 feet between the locations. This sums to $7 \cdot 80 = 560$ thousand customer-feet to the flow-distance.

TABLE 11.5 Mall Layout Application Tenants

Store, <i>i</i>	Common Customers with <i>k</i> (000's)			
	1	2	3	4
1: Clothes Are	—	5	2	7
2: Computers Aye	5	—	3	8
3: Toy Parade	2	3	—	3
4: Book Bazaar	7	8	3	—

Mall Layout Application Model

Notice that the flow-distance for any pair of shops cannot be computed until we know where both are assigned. This is the assignment combinations characteristic that yields quadratic assignment models.

Using the decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if shop } i \text{ is assigned to location } j \\ 0 & \text{otherwise} \end{cases}$$

the required quadratic assignment model is

$$\begin{aligned}
 \min \quad & 5(80x_{1,1}x_{2,2} + 150x_{1,1}x_{2,3} + 170x_{1,1}x_{2,4} && \text{(shops 1 and 2)} \\
 & + 80x_{1,2}x_{2,1} + 130x_{1,2}x_{2,3} + 100x_{1,2}x_{2,4} \\
 & + 150x_{1,3}x_{2,1} + 130x_{1,3}x_{2,2} + 120x_{1,3}x_{2,4} \\
 & + 170x_{1,4}x_{2,1} + 100x_{1,4}x_{2,2} + 120x_{1,4}x_{2,3}) \\
 & 2(80x_{1,1}x_{3,2} + 150x_{1,1}x_{3,3} + 170x_{1,1}x_{3,4} && \text{(shops 1 and 3)} \\
 & + 80x_{1,2}x_{3,1} + 130x_{1,2}x_{3,3} + 100x_{1,2}x_{3,4} \\
 & + 150x_{1,3}x_{3,1} + 130x_{1,3}x_{3,2} + 120x_{1,3}x_{3,4} \\
 & + 170x_{1,4}x_{3,1} + 100x_{1,4}x_{3,2} + 120x_{1,4}x_{3,3}) \\
 & 7(80x_{1,1}x_{4,2} + 150x_{1,1}x_{4,3} + 170x_{1,1}x_{4,4} && \text{(shops 1 and 4)} \\
 & + 80x_{1,2}x_{4,1} + 130x_{1,2}x_{4,3} + 100x_{1,2}x_{4,4} \\
 & + 150x_{1,3}x_{4,1} + 130x_{1,3}x_{4,2} + 120x_{1,3}x_{4,4} && \text{(11.12)} \\
 & + 170x_{1,4}x_{4,1} + 100x_{1,4}x_{4,2} + 120x_{1,4}x_{4,3}) \\
 & 3(80x_{2,1}x_{3,2} + 150x_{2,1}x_{3,3} + 170x_{2,1}x_{3,4} && \text{(shops 2 and 3)} \\
 & + 80x_{2,2}x_{3,1} + 130x_{2,2}x_{3,3} + 100x_{2,2}x_{3,4} \\
 & + 150x_{2,3}x_{3,1} + 130x_{2,3}x_{3,2} + 120x_{2,3}x_{3,4} \\
 & + 170x_{2,4}x_{3,1} + 100x_{2,4}x_{3,2} + 120x_{2,4}x_{3,3}) \\
 & 8(80x_{2,1}x_{4,2} + 150x_{2,1}x_{4,3} + 170x_{2,1}x_{4,4} && \text{(shops 2 and 4)} \\
 & + 80x_{2,2}x_{4,1} + 130x_{2,2}x_{4,3} + 100x_{2,2}x_{4,4} \\
 & + 150x_{2,3}x_{4,1} + 130x_{2,3}x_{4,2} + 120x_{2,3}x_{4,4} \\
 & + 170x_{2,4}x_{4,1} + 100x_{2,4}x_{4,2} + 120x_{2,4}x_{4,3}) \\
 & 3(80x_{3,1}x_{4,2} + 150x_{3,1}x_{4,3} + 170x_{3,1}x_{4,4} && \text{(shops 3 and 4)} \\
 & + 80x_{3,2}x_{4,1} + 130x_{3,2}x_{4,3} + 100x_{3,2}x_{4,4} \\
 & + 150x_{3,3}x_{4,1} + 130x_{3,3}x_{4,2} + 120x_{3,3}x_{4,4} \\
 & + 170x_{3,4}x_{4,1} + 100x_{3,4}x_{4,2} + 120x_{3,4}x_{4,3}) \\
 \text{s.t.} \quad & x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} = 1 && \text{(1, Clothes Are)} \\
 & x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} = 1 && \text{(2, Computers Aye)} \\
 & x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1 && \text{(3, Toy Parade)} \\
 & x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} = 1 && \text{(4, Book Bazaar)} \\
 & x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} = 1 && \text{(location 1)} \\
 & x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} = 1 && \text{(location 2)} \\
 & x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} = 1 && \text{(location 3)} \\
 & x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} = 1 && \text{(location 4)} \\
 & x_{i,j} = 0 \text{ or } 1 \quad i = 1, \dots, 4; j = 1, \dots, 4
 \end{aligned}$$

The objective function computes total flow distance for all pairs of shops and all possible assigned locations. Assignment constraints assure that one shop goes to each location and each location gets one shop. An optimal assignment places shop 1 in location 1, shop 2 in location 4, shop 3 in location 3, and shop 4 in location 2, for a total flow distance of 3260 thousand customer-feet.

EXAMPLE 11.11: FORMULATING QUADRATIC ASSIGNMENT MODELS

An industrial engineer has divided a proposed machine shop’s floor area into 12 grid squares, g , each of which will be the location of a single machine m . He has also estimated the distance, $d_{g,g'}$, between all pairs of grid squares and the number of units, $f_{m,m'}$, that will have to travel between machines m and m' (in both directions) during each week of operation. Formulate a quadratic assignment model to layout the shop in a way that will minimize material handling cost (i.e., minimize the product of between machine flows and the distance between their locations). Assume $d_{g,g'} = d_{g',g}$.

Solution: Using the decision variables

$$x_{m,g} \triangleq \begin{cases} 1 & \text{if machine } m \text{ is located at grid square } g \\ 0 & \text{otherwise} \end{cases}$$

the required model is

$$\begin{aligned} \min \quad & \sum_{m=1}^{12} \sum_{g=1}^{12} \sum_{\substack{m' > m \\ g' = 1 \\ g' \neq g}}^{12} f_{m,m'} d_{g,g'} x_{m,g} x_{m',g'} \quad (\text{flow distance}) \\ \text{s.t.} \quad & \sum_{g=1}^{12} x_{m,g} = 1 \quad m = 1, \dots, 12 \quad (\text{square per machine}) \\ & \sum_{m=1}^{12} x_{m,g} = 1 \quad g = 1, \dots, 12 \quad (\text{machine per square}) \\ & x_{i,j} = 0 \text{ or } 1 \quad m = 1, \dots, 12; \quad g = 1, \dots, 12 \end{aligned}$$

Machines are assigned to grid squares to minimize the total flow distance which measures the material handling implication of a layout. Index ranges in the objective function assure that each pair of machines and locations is reflected just once.

Generalized Assignment Models

Main assignment constraints of definition [11.14](#) require, respectively, that each object i of one set is assigned to exactly one j of the other, and that each j receives one i . Suppose, instead, that each object i must be assigned to some j , but that j ’s may receive several i . Specifically, define

- $b_j \triangleq$ capacity of j
- $s_{i,j} \triangleq$ size, space, or similar amount of j ’s capacity consumed if i is assigned to j
- $c_{i,j} \triangleq$ cost (or benefit) of assigning i to j

Then finding the best way to assign all i without violating capacities is called a generalized assignment model.

Definition 11.17 | **Generalized assignment models**, which encompass cases where allocation of i to j requires fixed size or space $s_{i,j}$ within j capacity b_j , have the form

$$\begin{aligned} \text{min or max} \quad & \sum_i \sum_j c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_j x_{i,j} = 1 \quad \text{for all } i \\ & \sum_j s_{i,j} x_{i,j} \leq b_j \quad \text{for all } j \\ & x_{i,j} = 0 \text{ or } 1 \quad \text{for all } i, j \end{aligned}$$

Here $c_{i,j}$ is the cost (or benefit) of assigning i to j and all sums are limited to (i, j) combinations allowed in the instance.

APPLICATION 11.6: CDOT GENERALIZED ASSIGNMENT

The Canadian Department of Transportation encountered a problem of the generalized assignment form when reviewing their allocation of coast guard ships on Canada's Pacific coast.⁴ The ships maintain such navigational aids as lighthouses and buoys. Each of the districts along the coast is assigned to one of a smaller number of coast guard ships. Since the ships have different home bases and different equipment and operating costs, the time and cost for assigning any district varies considerably among the ships. The task is to find a minimum cost assignment.

Table 11.6 shows data for our (fictitious) version of the problem. Three ships—the Estevan, the Mackenzie, and the Skidegate—are available to serve 6 districts. Entries in the table show the number of weeks each ship would require to maintain aids in each district, together with the annual cost (in thousands of Canadian dollars). Each ship is available 50 weeks per year.

TABLE 11.6 Costs and Times for the CDOT Application

Ship, j		District, i					
		1	2	3	4	5	6
1: Estevan	Cost	130	30	510	30	340	20
	Time	30	50	10	11	13	9
2: Mackenzie	Cost	460	150	20	40	30	450
	Time	10	20	60	10	10	17
3: Skidegate	Cost	40	370	120	390	40	30
	Time	70	10	10	15	8	12

⁴Based on J. G. Debanne and Jean-Noel Lavier (1979), "Management Science in the Public Sector—the Estevan Case," *Interfaces*, 9:2, part 2, 66–77.

CDOT Application Model

Using the decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if district } i \text{ is assigned to ship } j \\ 0 & \text{otherwise} \end{cases}$$

this CDOT application can be formulated

$$\begin{aligned} \min \quad & 130x_{1,1} + 460x_{1,2} + 40x_{1,3} + 30x_{2,1} + 150x_{2,2} + 370x_{2,3} \\ & + 510x_{3,1} + 20x_{3,2} + 120x_{3,3} + 30x_{4,1} + 40x_{4,2} + 390x_{4,3} \\ & + 340x_{5,1} + 30x_{5,2} + 40x_{5,2} + 20x_{6,1} + 450x_{6,2} + 30x_{6,3} \\ \text{s.t.} \quad & x_{1,1} + x_{1,2} + x_{1,3} = 1 && (\text{district 1}) \\ & x_{2,1} + x_{2,2} + x_{2,3} = 1 && (\text{district 2}) \\ & x_{3,1} + x_{3,2} + x_{3,3} = 1 && (\text{district 3}) \\ & x_{4,1} + x_{4,2} + x_{4,3} = 1 && (\text{district 4}) \\ & x_{5,1} + x_{5,2} + x_{5,3} = 1 && (\text{district 5}) \\ & x_{6,1} + x_{6,2} + x_{6,3} = 1 && (\text{district 6}) \\ & 30x_{1,1} + 50x_{2,1} + 10x_{3,1} && (\text{Estevan}) \\ & \quad + 11x_{4,1} + 13x_{5,1} + 9x_{6,1} \leq 50 \\ & 10x_{1,2} + 20x_{2,2} + 60x_{3,2} && (\text{Mackenzie}) \\ & \quad + 10x_{4,2} + 10x_{5,2} + 17x_{6,2} \leq 50 \\ & 70x_{1,3} + 10x_{2,3} + 10x_{3,3} && (\text{Skidegate}) \\ & \quad + 15x_{4,3} + 8x_{5,3} + 12x_{6,3} \leq 50 \\ & x_{i,j} = 0 \text{ or } 1 \quad i = 1, \dots, 6; j = 1, \dots, 3 \end{aligned} \tag{11.13}$$

The objective function minimizes total cost. The first 6 constraints assure that every district is assigned to one ship, and the last 3 keep work assigned to each ship within the 50 weeks available. An optimal solution assigns districts 1, 4, and 6 to the Estevan, districts 2 and 5 to the Mackenzie, and district 3 to the Skidegate at a total cost of \$480,000.

EXAMPLE 11.12: FORMULATING GENERALIZED ASSIGNMENT MODELS

Objects $i = 1, \dots, 100$ of volume c_i cubic meters are being stored in an automated warehouse. Storage locations $j = 1, \dots, 20$ are located d_j meters from the system's input/output station, and all have capacity b cubic meters. Formulate a generalized assignment model to store all items at minimum total travel distance assuming that as many objects can be placed in any location as volume permits.

Solution: Using the decision variables

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if object } i \text{ is stored in location } j \\ 0 & \text{otherwise} \end{cases}$$

the required generalized assignment model is

$$\begin{aligned} \min \quad & \sum_{i=1}^{100} \sum_{j=1}^{20} d_j x_{i,j} && \text{(total distance)} \\ & \sum_{j=1}^{20} x_{i,j} = 1 && i = 1, \dots, 100 \quad \text{(each } i \text{ stored)} \\ & \sum_{j=1}^{100} c_j x_{i,j} \leq b && j = 1, \dots, 20 \quad \text{(} j \text{ capacity)} \\ & x_{i,j} = 0 \text{ or } 1 && i = 1, \dots, 100; \quad j = 1, \dots, 20 \end{aligned}$$

The objective function totals move distance to assigned locations, the first system of main constraints assures that every object is stored, and the second enforces capacities.

Matching Models

Assignment problems considered so far always pair objects in two distinct sets. One final variation eliminates the distinction between the sets. Decision variables of such matching models are

$$x_{i,i'} \triangleq \begin{cases} 1 & \text{if } i \text{ is paired with } i' \\ 0 & \text{otherwise} \end{cases}$$

where by convention index $i' > i$ to avoid double counting.

Definition 11.18 **Matching models**, which seek an optimal pairing of like objects i , have the form

$$\begin{aligned} \min \text{ or } \max \quad & \sum_i \sum_{i' > i} c_{i,i'} x_{i,i'} \\ \text{s.t.} \quad & \sum_{i' < i} x_{i',i} + \sum_{i' > i} x_{i,i'} = 1 \quad \text{for all } i \\ & x_{i,i'} = 0 \text{ or } 1 \quad \text{for all } i, i' > i \end{aligned}$$

Here $c_{i,i'}$ is the cost (or benefit) of pairing i with i' and all sums are limited to (i, i') combinations allowed in the instance.

The two sums in each main constraint of formulation [11.18](#) are required because any particular i will be the higher index in some pairs and the lower index in others.

Matching models include linear assignment cases [11.15](#) if allowed pairings are restricted to those matching an object in one class with an object in another. However, the matching term is usually reserved for the more general case where all objects come from a single class.

APPLICATION 11.7: SUPERFI SPEAKER MATCHING

We may illustrate matching with the task faced by fictitious high-fidelity speaker manufacturer Superfi. Superfi sells its speakers in pairs. Even though the manufacturing process maintains the most rigid quality standards, any two speakers produced will still interfere slightly with each other when connected to the same stereo system.

To improve its product quality even more, Superfi has measured the distortion $d_{i,i'}$ for each pair of speakers in the current lot. They wish to determine how to pair the speakers so that total distortion is minimized.

Notice that any two speakers may be paired. There is no distinction between large and small, or left and right.

Superfi Application Model

We may model this problem with the decision variables

$$x_{i,i'} \triangleq \begin{cases} 1 & \text{if speakers } i \text{ and } i' \text{ are paired} \\ 0 & \text{otherwise} \end{cases}$$

There is one for each pair (i, i') , $i < i'$. The corresponding matching model [11.18] is

$$\begin{aligned} \min \quad & \sum_i \sum_{i' > i} d_{i,i'} x_{i,i'} && \text{(distortion)} \\ \text{s.t.} \quad & \sum_{i' < i} x_{i',i} + \sum_{i' > i} x_{i',i} = 1 && \text{for all } i \quad \text{(each speaker paired)} \\ & x_{i,i'} = 0 \text{ or } 1 && \text{for all } i, i' > i \end{aligned} \tag{11.14}$$

The objective function sums the distortion of all selected pairs, and main constraints assure that each speaker is part of exactly one pair.

EXAMPLE 11.13: FORMULATING MATCHING MODELS

The instructor in an operations research class is assigning his students to 2-person teams for a term project. Each student s has scored his or her preference $p_{s,s'}$ for working with each other student s' . Formulate a matching model to form teams in a way that maximizes total preference.

Solution: Using the decision variables

$$x_{i,i'} \triangleq \begin{cases} 1 & \text{if } i \text{ is teamed with } i' \\ 0 & \text{otherwise} \end{cases}$$

the required matching model is

$$\begin{aligned} \max \quad & \sum_i \sum_{i' > i} (p_{i,i'} + p_{i',i}) x_{i,i'} && \text{(preference)} \\ \text{s.t.} \quad & \sum_{i' < i} x_{i',i} + \sum_{i' > i} x_{i',i} = 1 && \text{for all } i \quad \text{(each student paired)} \\ & x_{i,i'} = 0 \text{ or } 1 && \text{for all } i, i' > i \end{aligned}$$

Each term of the objective captures the two-way preference gain of teaming i with i' , and the main constraints assure that each student is assigned to one team.

Tractability of Assignment and Matching Models

The various models of the assignment family presented in this section provide a good illustration of the tremendous variation in tractability of discrete optimization models.

Principle 11.19 | Linear assignment models are highly tractable because they can be viewed as network flow problems, and thus as special cases of linear programming. Even more efficient algorithms exist (Section 10.7).

Principle 11.20 | It is very difficult to compute global optima in quadratic assignment models because the nonlinear objective function precludes even the ILP methods of Chapter 12. Improving search heuristics like those of Chapter 15 are usually applied.

Principle 11.21 | Generalized assignment models, which are ILPs, can be solved in moderate size by the methods of Chapter 12. Still, they are far less tractable than the linear assignment case.

Principle 11.22 | Matching models are more difficult to solve than linear assignment cases, but efficient special-purpose algorithms are known that can solve quite large instances.

11.5 TRAVELING SALESMAN AND ROUTING MODELS

Among the most common discrete optimization problems are those that organize a collection of customer locations, jobs, cities, points, and so on, into sequences or routes. Sometimes such **routing models** form all points into a single sequence. Other times, several routes are required.

Traveling Salesman Problem

The simplest and most famous of routing problems is known to researchers as the traveling salesman problem.

Definition 11.23 | The **traveling salesman problem (TSP)** seeks a minimum-total-length route visiting every point in a given set exactly once.

The name derives from a mythical salesperson who must make a **tour** of the **cities** in his or her territory while traveling the least possible distance. TSPs actually occur in a much wider expanse of applications. Any task of sequencing objects in minimum total cost, length, or time can be viewed as a traveling salesman problem.

APPLICATION 11.8: NCB CIRCUIT BOARD TSP

A practical scenario for thinking about traveling salesman problems occurs in the manufacture of printed circuit boards for electronics.⁵ Circuit boards have many small holes through which chips and other components are wired. In a typical

⁵Based in part on S. Danusaputro, Chung-Yee Lee, and L. A. Martin-Vega (1990), “An Efficient Algorithm for Drilling Printed Circuit Boards,” *Computers and Industrial Engineering*, 18, 145–151.

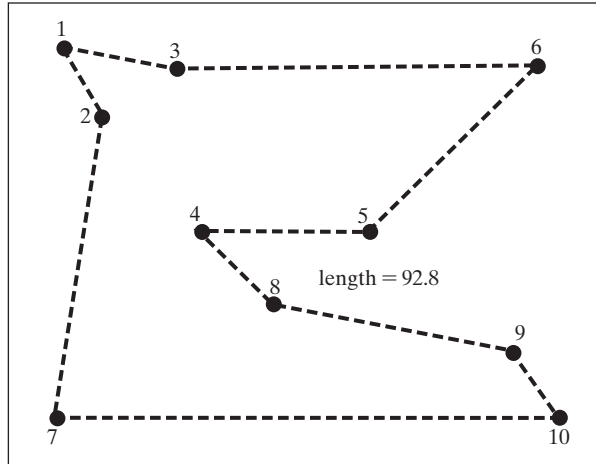


FIGURE 11.4 Board Drilling Locations for NCB Application

example, several hundred holes may have to be drilled in up to 10 different sizes. Efficient manufacture requires that these holes be completed as rapidly as possible by a moving drill. Thus for any single size, the question of finding the most efficient drilling sequence is a traveling salesman routing problem.

Figure 11.4 shows the tiny instance that we will investigate for fictional board manufacturer NCB. We seek an optimal route through the 10 hole locations indicated. Table 11.7 reports straight-line distances $d_{i,j}$ between hole locations i and j . Lines in Figure 11.4 show a fair quality solution with total length 92.8 inches. The best route is 11 inches shorter (see Section 15.2).

Symmetric versus Asymmetric Cases of the TSP

An important distinction among traveling salesman problems concerns whether distances between points are symmetric or asymmetric.

TABLE 11.7 Distances between Holes in NCB Application

$i \backslash j$	1	2	3	4	5	6	7	8	9	10
1	—	3.6	5.1	10.0	15.3	20.0	16.0	14.2	23.0	26.4
2	3.6	—	3.6	6.4	12.1	18.1	13.2	10.6	19.7	23.0
3	5.1	3.6	—	7.1	10.6	15.0	15.8	10.8	18.4	21.9
4	10.0	6.4	7.1	—	7.0	15.7	10.0	4.2	13.9	17.0
5	15.3	12.1	10.6	7.0	—	9.9	15.3	5.0	7.8	11.3
6	20.0	18.1	15.0	15.7	9.9	—	25.0	14.9	12.0	15.0
7	16.0	13.2	15.8	10.0	15.3	25.0	—	10.3	19.2	21.0
8	14.2	10.6	10.8	4.2	5.0	14.9	10.3	—	10.2	13.0
9	23.0	19.7	18.4	13.9	7.8	12.0	19.2	10.2	—	3.6
10	26.4	23.0	21.9	17.0	11.3	15.0	21.0	13.0	3.6	—

Definition 11.24 A traveling salesman problem is **symmetric** if the distance or cost of passing from any point i to any other point j is the same as the distance from j to i . Otherwise, the problem is **asymmetric**.

Our NCB application is symmetric because $d_{i,j} = d_{j,i}$ in Table 11.7. In other cases, distances are not symmetric in this way because travel from i to j is with the traffic, and travel from j to i against, or because of a host of similar asymmetric circumstances.

Formulating the Symmetric TSP

One thing that intrigues researchers about traveling salesman problems is that there are many different formulations—none of them straightforward. Most ILP models of the symmetric case employ decision variables, $i < j$,

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if the route includes a leg between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Notice that we define $x_{i,j}$ only for $i < j$. This numbering convention avoids the duplication that could result because a leg between i and j implies one between j and i , and costs are the same.

In terms of these new decision variables, the total route length now has the easy linear form

$$\min \sum_i \sum_{j>i} d_{i,j} x_{i,j} \quad (11.15)$$

What makes ILP traveling salesman models complex is their constraints. A little contemplation will make one system clear. In the symmetric case, exactly two x variables relating to any point i can be = 1 in a feasible solution. One links i to the city before it in the route, and the other links i to the city after. We can express this requirement mathematically with constraints

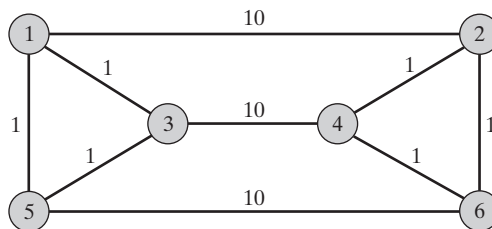
$$\sum_{j<i} x_{j,i} + \sum_{j>i} x_{i,j} = 2 \quad \text{for all } i \quad (11.16)$$

A specific instance for $i = 5$ in the NCB application of Figure 11.4 is

$$x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{5,6} + x_{5,7} + x_{5,8} + x_{5,9} + x_{5,10} = 2$$

EXAMPLE 11.14: MODELING TSPs AS ILPs

The following graph shows the available links joining 6 points, with numbers on edges indicating transit times.



We wish to find the shortest time route visiting all nodes exactly once and using only links shown in the graph.

- (a) Explain why this problem can be viewed as a symmetric traveling salesman problem.
- (b) Formulate integer linear program objective (11.15) for this instance.
- (c) Formulate constraints (11.16) for this instance.

Solution:

(a) The problem is a TSP because it requires a closed route visiting each point. It is symmetric because the time is the same whether a link is passed in the *i*-to-*j* or the *j*-to-*i* direction.

(b) The linear objective required is

$$\begin{aligned} \min \quad & 10x_{1,2} + 1x_{1,3} + 1x_{1,5} + 1x_{2,4} + 1x_{2,6} \\ & + 10x_{3,4} + 1x_{3,5} + 1x_{4,6} + 10x_{5,6} \end{aligned}$$

which minimizes total tour length.

(c) Needed constraints (11.16) are

$$\begin{aligned} x_{1,2} + x_{1,3} + x_{1,5} &= 2 && \text{(node 1)} \\ x_{1,2} + x_{2,4} + x_{2,6} &= 2 && \text{(node 2)} \\ x_{1,3} + x_{3,4} + x_{3,5} &= 2 && \text{(node 3)} \\ x_{2,4} + x_{3,4} + x_{4,6} &= 2 && \text{(node 4)} \\ x_{1,5} + x_{3,5} + x_{5,6} &= 2 && \text{(node 5)} \\ x_{2,6} + x_{4,6} + x_{5,6} &= 2 && \text{(node 6)} \end{aligned}$$

Subtours

Figure 11.5 illustrates why constraints (11.16) are not usually enough. The solution shown does have two links at each point. But it divides the 10 hole locations among three **subtours** or miniroutes. We seek a single route through all the points.

No one has any difficulty understanding subtours, but constraints to prevent them are less obvious. One form of **subtour elimination constraints** is obtained from any

$$S \triangleq \text{proper subset of the points/cities to be routed}$$

Every tour must cross between points in *S* and points outside at least twice. This leads to constraints of the form

$$\left(\begin{array}{l} \text{number of legs} \\ \text{between points in } S \\ \text{and points not in } S \end{array} \right) = \sum_{i \in S} \sum_{j \notin S} x_{i,j} = \sum_{i \notin S} \sum_{j \in S} x_{i,j} \geq 2 \quad (11.17)$$

There is one such constraint for every proper subset *S* of at least 3 cities.

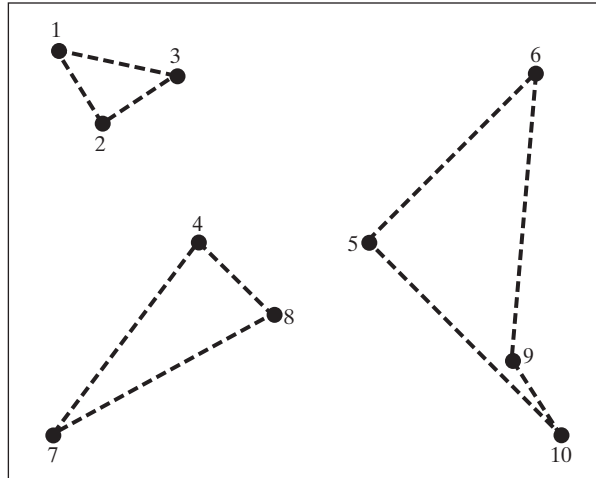


FIGURE 11.5 Subtour Solution for NCB Application

The subtour solution of Figure 11.5 violates several of these constraints. For example, pick $S \triangleq \{5, 6, 9, 10\}$. The corresponding subtour elimination constraint lists all possible tour legs passing into or out of set S :

$$\begin{aligned}
 &x_{1,5} + x_{1,6} + x_{1,9} + x_{1,10} + x_{2,5} \\
 &\quad + x_{2,6} + x_{2,9} + x_{2,10} + x_{3,5} + x_{3,6} \\
 &\quad + x_{3,9} + x_{3,10} + x_{4,5} + x_{4,6} + x_{4,9} \\
 &\quad + x_{4,10} + x_{5,7} + x_{6,7} + x_{7,9} + x_{7,10} \\
 &\quad + x_{5,8} + x_{6,8} + x_{8,9} + x_{8,10} \\
 &\geq 2
 \end{aligned}$$

EXAMPLE 11.15: FORMING SUBTOUR ELIMINATION CONSTRAINTS

Return to the TSP of Example 11.14.

- (a) Show by inspection that the least cost binary solution over $= 2$ constraints of Example 11.14(b) has subtours.
- (b) Formulate a subtour elimination constraint (11.17) not satisfied by the subtour solution of part (a).

Solution:

- (a) It is obvious that the least cost binary solution touching each node with exactly two links has

$$\begin{aligned}
 x_{1,3} &= x_{1,5} = x_{3,5} = 1 \\
 x_{2,4} &= x_{2,6} = x_{4,6} = 1 \\
 x_{1,2} &= x_{3,4} = x_{5,6} = 0
 \end{aligned}$$

It contains subtours 1–3–5–1 and 2–4–6–2.

Principle 11.26 The asymmetric traveling salesman problem can be formulated as the integer linear program

$$\begin{aligned} \min \quad & \sum_i \sum_{j \neq i} d_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_j x_{j,i} = 1 \quad \text{for all } i \\ & \sum_j x_{i,j} = 1 \quad \text{for all } i \\ & \sum_{i \in S} \sum_{j \notin S} x_{i,j} \geq 1 \quad \text{for all proper point subsets } S, |S| \geq 2 \\ & x_{i,j} = 0 \text{ or } 1 \quad \text{for all } i, j \end{aligned}$$

where $x_{i,j} = 1$ if the tour passes from i to j and $d_{i,j} \triangleq$ distance from i to j .

EXAMPLE 11.16: FORMULATING ASYMMETRIC TSPS

Return to the TSP of Examples 11.14 and 11.15, and assume that a 2-unit cost penalty is required when the tour passes from a higher to a lower node number. That is, $d_{1,2} = 10$, but $d_{2,1} = 10 + 2 = 12$, $d_{1,3} = 1$, but $d_{3,1} = 1 + 2 = 3$, and so on.

- Explain why the problem is now an asymmetric TSP.
- Formulate an objective function for the corresponding model [11.26](#).
- Formulate constraints requiring the tour to enter and leave each node exactly once.
- Formulate a subtour elimination constraint requiring the tour to leave node subset $S = \{1, 3, 5\}$ at least once.

Solution:

- The penalty of 2 makes $d_{i,j} \neq d_{j,i}$, which turns the problem asymmetric.
- Including variables for all one-way passages, the objective function of format [11.26](#) is

$$\begin{aligned} \min \quad & 10x_{1,2} + 1x_{1,3} + 1x_{1,5} \\ & + 12x_{2,1} + 1x_{2,4} + 1x_{2,6} \\ & + 3x_{3,1} + 10x_{3,4} + 1x_{3,5} \\ & + 3x_{4,2} + 12x_{4,3} + 1x_{4,6} \\ & + 3x_{5,1} + 3x_{5,3} + 10x_{5,6} \\ & + 3x_{6,2} + 3x_{6,4} + 10x_{6,5} \end{aligned}$$

- These constraints have the assignment form:

$$\begin{aligned} x_{2,1} + x_{3,1} + x_{5,1} &= 1 \\ x_{1,2} + x_{4,2} + x_{6,2} &= 1 \\ x_{1,3} + x_{4,3} + x_{5,3} &= 1 \\ x_{2,4} + x_{3,4} + x_{6,4} &= 1 \\ x_{1,5} + x_{3,5} + x_{6,5} &= 1 \\ x_{2,6} + x_{4,6} + x_{5,6} &= 1 \end{aligned}$$

plus

$$\begin{aligned} x_{1,2} + x_{1,3} + x_{1,5} &= 1 \\ x_{2,1} + x_{2,4} + x_{2,6} &= 1 \\ x_{3,1} + x_{3,4} + x_{3,5} &= 1 \\ x_{4,2} + x_{4,3} + x_{4,6} &= 1 \\ x_{5,1} + x_{5,3} + x_{5,6} &= 1 \\ x_{6,2} + x_{6,4} + x_{6,5} &= 1 \end{aligned}$$

(d) To avoid any subtour among nodes in $S = \{1, 3, 5\}$, we add subtour elimination constraint.

$$x_{1,2} + x_{3,4} + x_{5,6} \geq 1$$

Quadratic Assignment Formulation of the TSP

There are an enormous number of subtour elimination constraints for a TSP on even a modest number of points. That is why it is sometimes easier—especially in developing heuristic procedures of Chapter 15—to deal with a TSP formulation having simpler constraints. We can accomplish this, at the cost of making the objective function nonlinear, by formulating the TSP as a quadratic assignment (QAP) model [11.16](#).

Think of the tour as a sequence or permutation of the points to be visited. Decision variables in the QAP form assign positions k to points i ; that is,

$$y_{k,i} \triangleq \begin{cases} 1 & \text{if } k\text{th point visited is } i \\ 0 & \text{otherwise} \end{cases}$$

For example, the illustrative route of Figure 11.4 has

$$\begin{aligned} y_{1,1} = y_{2,3} = y_{3,6} = y_{4,5} = y_{5,4} = y_{6,8} \\ = y_{7,9} = y_{8,10} = y_{9,7} = y_{10,2} \\ = 1 \end{aligned}$$

when viewed as starting at hole 1. In terms of these variables, either the symmetric or the asymmetric case can be formulated as an INLP.

Principle 11.27 | The traveling salesman problem can be formulated as the quadratic assignment model

$$\begin{aligned} \min \quad & \sum_i \sum_j d_{i,j} \sum_k y_{k,i} y_{k+1,j} && \text{(total length)} \\ & \sum_i y_{k,i} = 1 && \text{for all } k \quad \text{(each position occupied)} \\ & \sum_k y_{k,i} = 1 && \text{for all } i \quad \text{(each point visited)} \\ & y_{k,i} = 0 \text{ or } 1 && \text{for all } k, i \end{aligned}$$

where $y_{k,i} = 1$ if the k th visited point is i , and $d_{i,j} \triangleq$ distance from point i to point j .

Constraints in [11.27] have the usual assignment format [11.14], but the objective function is less transparent. Terms

$$d_{i,j} \sum_k y_{k,i} y_{k+1,j}$$

add in distance $d_{i,j}$ exactly when j follows i somewhere in the chosen tour sequence. (Here we take $k + 1$ to mean 1 when k is the highest-numbered point.) In Figure 11.4, for example, one of the nonzero terms would be

$$d_{3,6} \cdot y_{2,3} \cdot y_{3,6}$$

because holes 3 and 6 could be second and third in sequence. Summing over all i and j recovers total distance.

Problems Requiring Multiple Routes

For many trucking and other distribution organizations, the problem is to design many routes, not just one. The task begins with a collection of stops to be serviced. Decisions first subdivide the stops into several routes and then choose the visitation sequence for each route.

APPLICATION 11.9: KI TRUCK ROUTING

Kraft Incorporated confronts such multiple-route design problems in planning truck delivery of its food products to over 100,000 commercial, industrial, and military customers in North America.⁶ Known customer requirements must be grouped into truckloads and then routes planned.

Our tiny fictitious version of the KI case is illustrated in Figure 11.6. The 20 stops shown in the figure are to be serviced from a single depot. Table 11.8 displays the requirements at each stop, expressed in fractions, f_i , of a truckload. The 7 routes depicted in Figure 11.6 show one feasible solution.

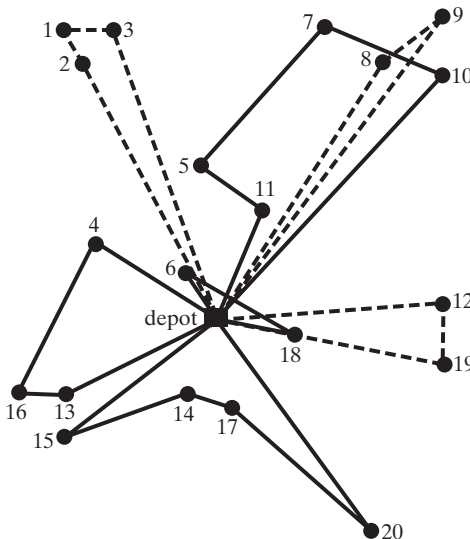


FIGURE 11.6 Depot and Delivery Locations in the KI Application

⁶Based on H. K. Chung and J. B. Norback (1991), "A Clustering and Insertion Heuristic Applied to a Large Routing Problem in Food Distribution," *Journal of the Operational Research Society*, 42, 555–564.

TABLE 11.8 Fractions of Truckloads to Be Delivered in KI Application

Stop, i	Fraction, f_i	Stop, i	Fraction, f_i	Stop, i	Fraction, f_i	Stop, i	Fraction, f_i
1	0.25	6	0.70	11	0.21	16	0.38
2	0.33	7	0.28	12	0.68	17	0.26
3	0.39	8	0.43	13	0.16	18	0.29
4	0.40	9	0.50	14	0.19	19	0.17
5	0.27	10	0.22	15	0.22	20	0.31

KI Truck Routing Application Model

With complex problems like KI’s, it is not easy to see where to begin a model.

Principle 11.28 Routing problems are characteristically difficult to represent concisely in optimization models.

We will start with the assignment of stops to routes. The decision variables

$$z_{i,j} \triangleq \begin{cases} 1 & \text{if stop } i \text{ is assigned to route } j \\ 0 & \text{otherwise} \end{cases}$$

Generalized assignment constraints (definition [11.17](#)) then manage the allocation of the 20 stops to seven routes:

$$\begin{aligned} \sum_{j=1}^7 z_{i,j} &= 1 && \text{for all } i = 1, \dots, 20 && \text{(each } i \text{ to some } j) \\ \sum_{i=1}^{20} f_i z_{i,j} &\leq 1 && \text{for all } j = 1, \dots, 7 && \text{(truck capacities)} \\ z_{i,j} &= 0 \text{ or } 1 && \text{for all } i = 1, \dots, 20; j = 1, \dots, 7 \end{aligned} \tag{11.18}$$

The first set makes sure every stop goes to some route, and the second keeps loads to assigned trucks within capacity.

The difficulty of concisely formulating routing problems becomes apparent when we try to express an objective function to go with (11.18). Certainly, it is something like

$$\sum_{j=1}^7 \theta_j(\mathbf{z}) \tag{11.19}$$

where \mathbf{z} is the vector of $z_{i,j}$ and

$$\theta_j(\mathbf{z}) \triangleq \text{length of the best route through stops assigned to truck } j \text{ by decision vector } \mathbf{z}$$

However, functions like $\theta_j(\mathbf{z})$ are highly nonlinear. In fact, $\theta_j(\mathbf{z})$ is the optimal solution value of a traveling salesman problem over the stops allocated to route j .

No concise expression of such functions exists. Still, we know what they mean in the problem context. Chapter 15 will show that good heuristic solutions can be obtained with rough approximations.

11.6 FACILITY LOCATION AND NETWORK DESIGN MODELS

Principle 11.2 and definition 11.3 of Section 11.1 demonstrated how **fixed charges** can be modeled with new binary variables and **switching constraints**. Virtually any linear program with fixed charges can be modeled in that way, but some forms are especially common. In this section we introduce the classic **facility location** and **network design** cases.

Facility Location Models

Facility location models, also called warehouse location models and plant location models, are perhaps the most frequently solved of all forms involving fixed charges.

Definition 11.29 **Facility/plant/warehouse location models** choose which of a proposed list of facilities to open in order to service specified customer demands at minimum total cost.

Costs include both the variable cost of servicing customers from chosen facilities and the fixed cost of opening the facilities.

APPLICATION 11.10: TMARK FACILITIES LOCATION

AT&T has confronted many facility location problems in recommending sites for the toll-free call-in centers of its telemarketing customers.⁷ Such centers handle telephone reservations and orders arising in many geographic zones. Since telephone rates vary dramatically depending on the zone of call origin and the location of the receiving center, site selection is extremely important. A well-designed system should minimize the total of call charges and center setup costs.

Our version of this scenario will involve fictional firm Tmark. Figure 11.7 shows the 8 sites under consideration for Tmark's catalog order centers embedded

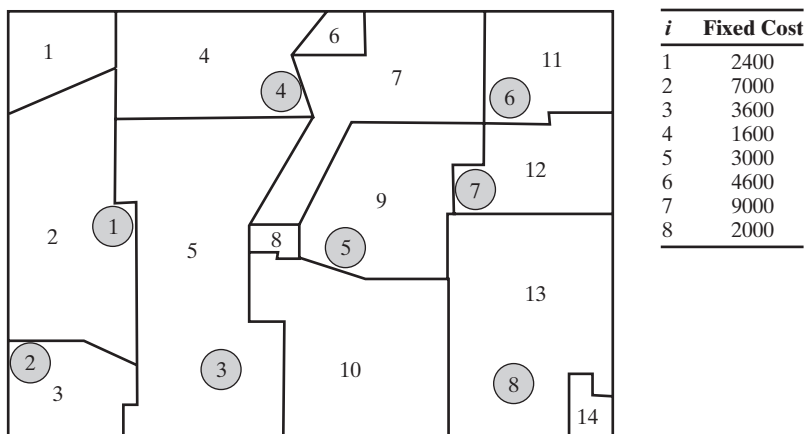


FIGURE 11.7 Customer Zones and Possible Facility Locations for Tmark Application

⁷Based on T. Spencer III, A. J. Brigandi, D. R. Dargon, and M. J. Sheehan (1990), "AT&T's Telemarketing Site Selection System Offers Customer Support," *Interfaces*, 20:1, 83-96.

TABLE 11.9 Unit Call Charges and Demands for Tmark Application

Zone, <i>j</i>	Possible Center Location, <i>i</i>								Call Demand
	1	2	3	4	5	6	7	8	
1	1.25	1.40	1.10	0.90	1.50	1.90	2.00	2.10	250
2	0.80	0.90	0.90	1.30	1.40	2.20	2.10	1.80	150
3	0.70	0.40	0.80	1.70	1.60	2.50	2.05	1.60	1000
4	0.90	1.20	1.40	0.50	1.55	1.70	1.80	1.40	80
5	0.80	0.70	0.60	0.70	1.45	1.80	1.70	1.30	50
6	1.10	1.70	1.10	0.60	0.90	1.30	1.30	1.40	800
7	1.40	1.40	1.25	0.80	0.80	1.00	1.00	1.10	325
8	1.30	1.50	1.00	1.10	0.70	1.50	1.50	1.00	100
9	1.50	1.90	1.70	1.30	0.40	0.80	0.70	0.80	475
10	1.35	1.60	1.30	1.50	1.00	1.20	1.10	0.70	220
11	2.10	2.90	2.40	1.90	1.10	2.00	0.80	1.20	900
12	1.80	2.60	2.20	1.80	0.95	0.50	2.00	1.00	1500
13	1.60	2.00	1.90	1.90	1.40	1.00	0.90	0.80	430
14	2.00	2.40	2.00	2.20	1.50	1.20	1.10	0.80	200

in a map of the 14 calling zones. Table 11.9 shows corresponding unit calling charges, $r_{i,j}$, from each zone j to various centers i , and the zone’s anticipated call load, d_j .

Any Tmark center selected can handle between 1500 and 5000 call units per day. However, their fixed costs of operation vary significantly because of differences in labor and real estate prices. Estimated daily fixed costs, f_i , for the 8 centers are displayed in Figure 11.7.

ILP Model of Facilities Location

Clearly, there are two kinds of decisions to make in facilities location: which facilities to open, and how chosen facilities should serve customer demands. We employ the decision variables

$$y_i \triangleq \begin{cases} 1 & \text{if facility } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

to determine which sites are selected. A second set,

$$x_{i,j} \triangleq \text{fraction of customer } j \text{ demand satisfied from facility } i$$

decides how service is to be allocated. Combining produces a standard model of facilities location.

Principle 11.30 Basic facilities location problems can be formulated as the integer linear program

$$\min \sum_i \sum_j c_{i,j} d_j x_{i,j} = \sum_i f_i y_i \quad (\text{total cost})$$

$$\text{s.t.} \quad \sum_i x_{i,j} = 1 \quad \text{for all } j \quad (\text{fullfill } j \text{ demand})$$

(Continued)

$$\begin{aligned} \sum_j d_j x_{i,j} &\leq u_i y_i && \text{for all } i && (i \text{ capacity switching}) \\ x_{i,j} &\geq 0 && \text{for all } i, j \\ y_i &= 0 \text{ or } 1 && \text{for all } i \end{aligned}$$

where $x_{i,j} \triangleq$ fraction of demand j fulfilled from i , $y_i = 1$ if facility i is opened, d_j is the demand at j , $c_{i,j}$ is the unit cost of fulfilling j demand from i , f_i is the nonnegative fixed cost of opening facility i , and u_i is the capacity of facility i .

This objective function sums all variable and fixed costs. The first system of constraints assures that 100% of each customer demand is fulfilled. The second set switches “on” facility capacity when the corresponding $y_i = 1$. If the problem has no true capacities on facilities, any sufficiently large value can provide a u_i —perhaps total demand.

Sometimes customers must be 100% serviced from a single facility, so that constraints

$$x_{i,j} = 0 \text{ or } 1 \quad \text{for all } i, j$$

are also enforced. Other times, demand can be split among several facilities, which leaves $x_{i,j}$ continuous.

Tmark Facilities Location Application Model

Following format [11.30](#), we can express a Tmark’s facilities location problem in the mixed-integer linear program

$$\begin{aligned} \min \quad & \sum_{i=1}^8 \sum_{j=1}^{14} (r_{i,j} d_j) x_{i,j} + \sum_{i=1}^8 f_i y_i && \text{(total cost)} \\ \text{s.t.} \quad & \sum_{i=1}^8 x_{i,j} = 1 && \text{for all } j = 1, \dots, 14 \quad \text{(carry } j \text{ load)} \\ & 1500 y_i \leq \sum_{j=1}^{14} d_j x_{i,j} && \text{for all } i = 1, \dots, 8 \quad \text{(minimum at } i) \quad (11.20) \\ & \sum_{j=1}^{14} d_j x_{i,j} \leq 5000 y_i && \text{for all } i = 1, \dots, 8 \quad \text{(maximum at } i) \\ & x_{i,j} \geq 0 && \text{for all } i = 1, \dots, 8; \quad j = 1, \dots, 14 \\ & y_i = 0 \text{ or } 1 && \text{for all } i = 1, \dots, 8 \end{aligned}$$

Here the objective function sums calling and setup costs. The total cost of call load serviced by i for zone j is

$$(\text{demand at } j)(i, j \text{ telephone rate})(\text{fraction of } j \text{ serviced from } i) = d_j r_{i,j} x_{i,j}$$

Most constraints of model (11.20) are identical to pattern [11.30](#). Variables $x_{i,j}$ are continuous because it is assumed that the load from a zone can be split among several centers.

The one new element is switching constraints dealing with minimum operating levels. Constraints

$$1500y_i \leq \sum_{j=1}^{14} d_j x_{i,j}$$

enforce the requirement that any open center i must handle at least 1500 call units per day.

Solution of model (11.20) produces

$$y_4^* = y_8^* = 1$$

$$y_1^* = y_2^* = y_3^* = y_5^* = y_6^* = y_7^* = 0$$

with a total cost of \$10,153 per day. Zones 1, 2, 4, 5, 6, and 7 are serviced from center 4, and the rest from center 8.

EXAMPLE 11.17: FORMULATING FACILITIES LOCATION MODELS

Environmental protection authorities have identified 14 possible offices around the country from which inspectors would make annual visits to each of 111 sites at high risk for oil spills. They have also measured the travel cost $c_{i,j}$ from location i to every potential spill site j . Each spill site should be under the supervision of a single inspection office.

- (a) Formulate a facilities location model to choose a minimum total cost collection of offices and inspection assignments assuming that there is an annual fixed cost of f for operating any office.
- (b) Formulate a facilities location model to choose a minimum total cost collection of offices and inspection assignments assuming that fixed costs of operating office are unknown but that it has been decided to open at most 9.

Solution: We follow integer linear programming format [11.30].

(a) For this case the model required is

$$\begin{aligned} \min \quad & \sum_{i=1}^{14} \sum_{j=1}^{111} c_{i,j} x_{i,j} + f \sum_{i=1}^{14} y_i && \text{(total cost)} \\ \text{s.t.} \quad & \sum_{i=1}^{14} x_{i,j} = 1 && j = 1, \dots, 111 \quad \text{(each } j \text{ inspected)} \\ & \sum_{j=1}^{111} x_{i,j} \leq 111y_i && i = 1, \dots, 14 \quad \text{(} i \text{ switching)} \\ & x_{i,j} = 0 \text{ or } 1 && i = 1, \dots, 14; \quad j = 1, \dots, 111 \\ & y_i = 0 \text{ or } 1 && i = 1, \dots, 14 \end{aligned}$$

Since no capacities are specified for the offices, switching constraints use $u_i = 111$.

Every spill site can be covered by any open office. Variables $x_{i,j}$ are binary because spill sites must be inspected from a single office.

(b) For this case the model required is

$$\begin{aligned}
 \min \quad & \sum_{i=1}^{14} \sum_{j=1}^{111} c_{i,j} x_{i,j} && \text{(travel cost)} \\
 \text{s.t.} \quad & \sum_{j=1}^{14} x_{i,j} = 1 && j = 1, \dots, 111 \quad \text{(each } j \text{ inspected)} \\
 & \sum_{j=1}^{111} x_{i,j} \leq 111 y_i && i = 1, \dots, 14 \quad \text{(} i \text{ switching)} \\
 & \sum_{i=1}^{14} y_i \leq 9 && \text{(max nine chosen)} \\
 & x_{i,j} = 0 \text{ or } 1 && i = 1, \dots, 14; \quad j = 1, \dots, 111 \\
 & y_i = 0 \text{ or } 1 && i = 1, \dots, 14
 \end{aligned}$$

Fixed charges have been omitted and a new constraint added that limits the number of offices to 9.

Network Design Models

Facility location models decide which nodes of a network to open. Network design or fixed-charge network flow models decide which arcs to use. A continuous network flow in variables $x_{i,j}$ is augmented with discrete variables $y_{i,j}$ implementing fixed charges for opening/constructing/setting up arcs.

Definition 11.31 The **fixed-charge network flow** or **network design model** on a digraph on nodes $k \in V$ with net demand b_k , and arcs $(i, j) \in A$ with capacity $u_{i,j}$, unit cost $c_{i,j}$, and nonnegative fixed cost $f_{i,j}$ is

$$\begin{aligned}
 \min \quad & \sum_{(i,j) \in A} c_{i,j} x_{i,j} + \sum_{(i,j) \in A} f_{i,j} y_{i,j} \\
 \text{s.t.} \quad & \sum_{(i,k) \in A} x_{i,k} - \sum_{(k,j) \in A} x_{k,j} = b_k && \text{for all } k \in V \\
 & 0 \leq x_{i,j} \leq u_{i,j} y_{i,j} && \text{for all } (i, j) \in A \\
 & y_{i,j} = 0 \text{ or } 1 && \text{for all } (i, j) \in A
 \end{aligned}$$

Main constraints in $x_{i,j}$ mirror network flow models [10.3](#) (Section 10.1) in conserving flow at each node. Switching constraints (definition [11.3](#)) turn on arc capacities $u_{i,j}$ if the fixed charge is paid. As usual, capacities must be derived from other constraints if not given explicitly—possibly by taking $u_{i,j}$ = the largest feasible flow on arc (i, j) .

APPLICATION 11.11: WASTEWATER NETWORK DESIGN

Network design applications may involve telecommunications, electricity, water, gas, coal slurry, or any other substance that flows in a network. We illustrate with an application involving regional wastewater (sewer) networks.⁸

As new areas develop around major cities, entire networks of collector sewers and treatment plants must be constructed to service growing population. Figure 11.8 displays our particular (fictional) instance.

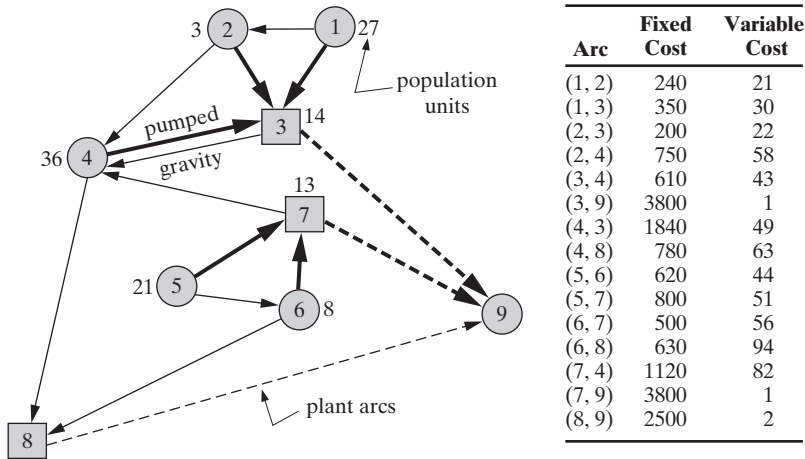


FIGURE 11.8 Wastewater Network Design Application

Nodes 1 to 8 of the network represent population centers where smaller sewers feed into the main regional network, and locations where treatment plants might be built. Wastewater loads are roughly proportional to population, so the inflows indicated at nodes represent population units (in thousands).

Arcs joining nodes 1 to 8 show possible routes for main collector sewers. Most follow the topology in gravity flow, but one pumped line (4, 3) is included. A large part of the construction cost for either type of line is fixed: right-of-way acquisition, trenching, and so on. Still, the cost of a line also grows with the number of population units carried, because greater flows imply larger-diameter pipes. The table in Figure 11.8 shows the fixed and variable cost for each arc in thousand of dollars.

Treatment plant costs actually occur at nodes—here nodes 3, 7, and 8. Figure 11.8 illustrates, however, that such costs can be modeled on arcs by introducing an artificial “supersink” node 9. Costs shown for arcs (3, 9), (7, 9), and (8, 9) capture the fixed and variable expense of plant construction as flows depart the network.

⁸Based on J. J. Jarvis, R. L. Rardin, V. E. Unger, R. W. Moore, and C. C. Schimpeler (1978), “Optimal Design of Regional Wastewater Systems: A Fixed Charge Network Flow Model,” *Operations Research*, 26, 538–550.

Wastewater Network Design Application Model

To place our wastewater network design problem in format [11.31](#), we need capacities $u_{i,j}$ on the arcs. None are explicitly provided, but it is not difficult to determine the maximum possible flow on each arc. For example, we may take

$$u_{2,3} = 27 + 3 = 30$$

because no solution would send more than the 27 thousand population units at node 1 and the 3 at node 2 along arc (2, 3).

With such derived capacities, we may model the wastewater system design problem of Figure 11.8 as the following fixed-charge network flow problem:

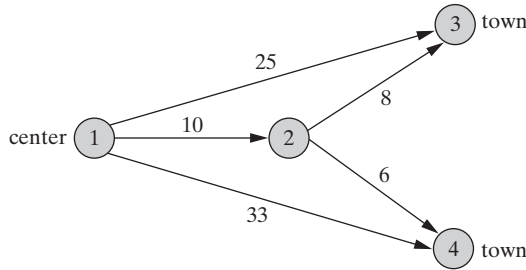
$$\begin{aligned}
 \min \quad & 21x_{1,2} + 30x_{1,3} + 22x_{2,3} + 58x_{2,4} + 43x_{3,4} && \text{(total cost)} \\
 & + 1x_{3,9} + 49x_{4,3} + 63x_{4,8} + 44x_{5,6} + 51x_{5,7} \\
 & + 56x_{6,7} + 94x_{6,8} + 82x_{7,4} + 1x_{7,9} + 2x_{8,9} \\
 & + 240y_{1,2} + 350y_{1,3} + 200y_{2,3} + 750y_{2,4} + 610y_{3,4} \\
 & + 3800y_{3,9} + 1840y_{4,3} + 780y_{4,8} + 620y_{5,6} + 800y_{5,7} \\
 & + 500y_{6,7} + 630y_{6,8} + 1120y_{7,4} + 3800y_{7,9} + 2500y_{8,9} \\
 \text{s.t.} \quad & -x_{1,2} - x_{1,3} &= & -27 && \text{(node 1)} \\
 & x_{1,2} - x_{2,3} - x_{2,4} &= & -3 && \text{(node 2)} \\
 & x_{1,3} + x_{2,3} + x_{4,3} - x_{3,4} - x_{3,9} &= & -14 && \text{(node 3)} \\
 & x_{2,4} + x_{3,4} + x_{7,4} - x_{4,3} - x_{4,8} &= & -36 && \text{(node 4)} \\
 & -x_{5,6} - x_{5,7} &= & -21 && \text{(node 5)} \\
 & x_{5,6} - x_{6,7} - x_{6,8} &= & -8 && \text{(node 6)} \\
 & x_{5,7} + x_{6,7} - x_{7,4} - x_{7,9} &= & -13 && \text{(node 7)} \\
 & x_{4,8} + x_{6,8} - x_{8,9} &= & 0 && \text{(node 8)} \\
 & x_{3,9} + x_{7,9} + x_{8,9} &= & 122 && \text{(node 9)} \\
 & 0 \leq x_{1,2} \leq 27y_{1,2}, & 0 \leq x_{1,3} \leq 27y_{1,3}, & 0 \leq x_{2,3} \leq 30y_{2,3} && \text{(switching)} \\
 & 0 \leq x_{2,4} \leq 30y_{2,4}, & 0 \leq x_{3,4} \leq 44y_{3,4}, & 0 \leq x_{3,9} \leq 122y_{3,9} \\
 & 0 \leq x_{4,3} \leq 108y_{4,3}, & 0 \leq x_{4,8} \leq 122y_{4,8}, & 0 \leq x_{5,6} \leq 21y_{5,6} \\
 & 0 \leq x_{5,7} \leq 21y_{5,7}, & 0 \leq x_{6,7} \leq 29y_{6,7}, & 0 \leq x_{6,8} \leq 29y_{6,8} \\
 & 0 \leq x_{7,4} \leq 42y_{7,4}, & 0 \leq x_{7,9} \leq 42y_{7,9}, & 0 \leq x_{8,9} \leq 122y_{8,9} \\
 & y_{i,j} = 0 \text{ or } 1 & \text{all arcs } (i,j)
 \end{aligned} \tag{11.21}$$

Bold lines in Figure 11.8 indicate an optimal design. Lines (1, 3), (2, 3), (4, 3), (5, 7), and (6, 7) should be constructed, along with the plant at node 7. Total cost is \$15,571,000.

EXAMPLE 11.18: FORMULATING NETWORK DESIGN MODELS

The following digraph shows possible routes for cable television lines from broadcasting center node 1 to towns at nodes 3 and 4. Node 2 is a connection box that may

or may not be included in the ultimate system. Numbers on arcs are the fixed cost of the corresponding cable line.



Formulate a fixed-charge network flow model to choose a least cost design providing service to both towns.

Solution: The problem provides no explicit supplies, demands, or capacities, and there are no variable costs. Thus we may assume that demand = 1 at both cities, with 2 units of flow supplied at node 1. Capacity is then 2 on arc (1,2), which might carry both flows, and 1 on all other arcs.

Substituting these values in form 11.31 produces the integer linear programming model

$$\begin{aligned}
 \min \quad & 10y_{1,2} + 25y_{1,3} + 33y_{1,4} + 8y_{2,3} + 6y_{2,4} && \text{(cost)} \\
 \text{s.t.} \quad & -x_{1,2} - x_{1,3} - x_{1,4} = -2 && \text{(node 1)} \\
 & x_{1,2} - x_{2,3} - x_{2,4} = 0 && \text{(node 2)} \\
 & x_{1,3} + x_{2,3} = 1 && \text{(node 3)} \\
 & x_{1,4} + x_{2,4} = 1 && \text{(node 4)} \\
 & 0 \leq x_{1,2} \leq 2y_{1,2}, \quad 0 \leq x_{1,3} \leq y_{1,3}, \quad 0 \leq x_{1,4} \leq y_{1,4} && \text{(switching)} \\
 & 0 \leq x_{2,3} \leq y_{2,3}, \quad 0 \leq x_{2,4} \leq y_{2,4} \\
 & y_{1,2}, y_{1,3}, y_{1,4}, y_{2,3}, y_{2,4} = 0 \text{ or } 1
 \end{aligned}$$

11.7 PROCESSOR SCHEDULING AND SEQUENCING MODELS

Scheduling is the allocation of resources over time. The enormous range of applications encompasses staffing activities such as the ONB shift planning in Section 5.4, AA air crew scheduling in Section 11.3, Purdue final exam timetabling in Section 2.4, and construction project management in Section 9.7. In this section we introduce another very broad class: **processor scheduling** models that **sequence** a collection of jobs through a given set of processing devices.

APPLICATION 11.12: NIFTY NOTES SINGLE-MACHINE SCHEDULING

We begin with the binder scheduling problem confronting a fictitious Nifty Notes copy shop. Just before the start of each semester, professors at the nearby university supply Nifty Notes with a single original of their class handouts, a projection of the class enrollment, and a due date by which copies should be available. Then the Nifty

Notes staff must rush to print and bind the required number of copies before each class begins.

During the busy period each semester, Nifty Notes operates its single binding station 24 hours per day. Table 11.10 shows **process times**, **release times**, and **due dates** for the jobs $j = 1, \dots, 6$ now pending at the binder:

$p_j \triangleq$ estimated process time (in hours) job j will require to bind

$r_j \triangleq$ release time (hour) at which job j has/will become available for processing (relative to time 0 = now)

$d_j \triangleq$ due date (hour) by which job j should be completed (relative to time 0 = now)

Notice that two jobs are already late ($d_j < 0$).

TABLE 11.10 Nifty Notes Scheduling Application Data

	Binder Job, j					
	1	2	3	4	5	6
Process time, p_j	12	8	3	10	4	18
Release time, r_j	-20	-15	-12	-10	-3	2
Due date, d_j	10	2	72	-8	-6	60

We wish to choose an optimal sequence in which to accomplish these jobs. No more than one can be in process at a time; and once started, a job must be completed before another can begin.

Single-Processor Scheduling Problems

Our Nifty Notes application is a very simple case of single-processor (or single-machine) scheduling.

Definition 11.32 | **Single-processor (or single-machine) scheduling problems** seek an optimal sequence in which to complete a given collection of jobs on a single processor that can accommodate only one job at a time.

Usually (as in the Nifty Notes case), we also assume that **preemption** is not allowed. That is, one job cannot be interrupted to work on another.

Time Decision Variables

Since scheduling means assignment of resources over time, it is natural that one set of decision variables found in most models chooses job **start** or **completion times**.

Principle 11.33 | A set of (continuous) decision variables in processor scheduling models usually determines either the start or the completion time(s) of each job on the processor(s) it requires.

In our Nifty Notes application we employ

$x_j \triangleq$ time binding starts for job j (relative to time 0 = now)

Then one set of constraints requires each job to start after the later of now (time = 0) and the hour it will be released for processing:

$$x_j \geq \max \{0, r_j\} \quad j = 1, \dots, 6 \quad (\text{earliest start}) \quad (11.22)$$

We could just as well have modeled in terms of completion times. Still, both start and completion times need not be decided because one can be computed from the other through

$$(\text{start time}) + (\text{process time}) = (\text{completion time})$$

(Recall that process times are given constants.)

Conflict Constraints and Disjunctive Variables

The central issue in processor scheduling is that only one job should be in progress on any processor at any time. If we know just start (or finish) times, it is not difficult to check for a violation or **conflict**. Still, it is not easy to write standard mathematical programming constraints that prevent conflicts.

For any pair of jobs j and j' that might conflict on a processor, the appropriate **conflict constraint** is either

$$\begin{aligned} (\text{start time of } j) + (\text{process time of } j) &\leq \text{start time of } j' \\ \text{or} \\ (\text{start time of } j') + (\text{process time of } j') &\leq \text{start time of } j \end{aligned} \quad (11.23)$$

Either j must finish before j' begins, or vice versa. But only one possibility can hold, and to determine which, we must know whether j or j' starts first on the processor.

Often, operations research analysts deal with conflict prevention (11.23) outside the usual mathematical program format. Nonetheless, conflict avoidance requirements can be modeled explicitly with the aid of additional disjunctive variables.

Principle 11.34 | A set of (discrete) **disjunctive variables** in processor scheduling models usually determines the sequence in which jobs are started on processors by specifying whether each job j is scheduled before or after each other j' with which it might conflict.

Then we can enforce conflict prevention (11.23) with pairs of linear constraints.

Principle 11.35 | A processor scheduling model with job start times x_j and process times p_j can prevent conflicts between jobs j and j' with **disjunctive constraint pairs**

$$\begin{aligned} x_j + p_j &\leq x_{j'} + M(1 - y_{j,j'}) \\ x_{j'} + p_{j'} &\leq x_j + My_{j,j'} \end{aligned}$$

where M is a large positive constant, and binary disjunctive variable $y_{j,j'} = 1$ when j is scheduled before j' on the processor and $= 0$ if j' is first.

To illustrate for our Nifty Notes application, define

$$y_{j,j'} \triangleq \begin{cases} 1 & \text{if } j \text{ binding comes before } j' \\ 0 & \text{if } j' \text{ binding comes before } j \end{cases}$$

Construction [11.35](#) yields constraints

$$\left. \begin{aligned} x_j + p_j &\leq x_{j'} + M(1 - y_{j,j'}) \\ x_{j'} + p_{j'} &\leq x_j + My_{j,j'} \end{aligned} \right\} j = 1, \dots, 6; \quad j' > j \quad (11.24)$$

As in many other cases, we consider only $j' > j$, to avoid listing the same pair twice.

To see how the constraints prevent conflicts, consider the specific case of $j = 2$, $j' = 6$. Using processing times from Table 11.10, the corresponding pair (11.24) is

$$\begin{aligned} x_2 + 8 &\leq x_6 + M(1 - y_{2,6}) \\ x_6 + 18 &\leq x_2 + My_{2,6} \end{aligned}$$

If job 2 is started before job 6, $y_{2,6} = 1$, and the first constraint keeps the start time for 6 after the finish of 2. The second constraint is also enforced. However, the big- M term $My_{2,6}$ makes it true for any x_2, x_6 . On the other hand, if job 6 is started before job 2, so that $y_{2,6} = 0$, the first constraint of the pair is discounted and the second enforced.

EXAMPLE 11.19: FORMULATING CONFLICT CONSTRAINTS

Formulate integer linear programming constraints for feasible schedules on a single processor with jobs $j = 1, \dots, 3$ having process times 14, 3, and 7, respectively.

Solution: We employ the decision variables

$$\begin{aligned} x_j &\triangleq \text{start time of job } j \\ y_{j,j'} &\triangleq \begin{cases} 1 & \text{if } j \text{ is scheduled before } j' \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Then conflict constraints [11.35](#) assure that only one job is processed at a time are

$$\begin{aligned} x_1 + 14 &\leq x_2 + M(1 - y_{1,2}) \\ x_2 + 3 &\leq x_1 + My_{1,2} \\ x_1 + 14 &\leq x_3 + M(1 - y_{1,3}) \\ x_3 + 7 &\leq x_1 + My_{1,3} \\ x_2 + 3 &\leq x_3 + M(1 - y_{2,3}) \\ x_3 + 7 &\leq x_2 + My_{2,3} \end{aligned}$$

Variable-type restrictions

$$\begin{aligned} x_1, x_2, x_3 &\geq 0 \\ y_{1,2}, y_{1,3}, y_{2,3} &= 0 \text{ or } 1 \end{aligned}$$

complete the constraints required.

Handling of Due Dates

Readers will note that we have not formulated any constraints enforcing due dates for various jobs. This could easily be done by adding conditions of the form

$$x_j + p_j \leq d_j$$

However, it is not standard to enforce such requirements because there may very well be no feasible schedule that meets all due dates. In Nifty Notes data of Table 11.10, for example, some due dates have already passed; it is far more customary to reflect due dates in the objective function as explained in the next subsection.

Principle 11.36 | **Due dates** in processor scheduling models are usually handled as goals to be reflected in the objective function rather than as explicit constraints. Dates that must be met are termed **deadlines** to distinguish.

Processor Scheduling Objective Functions

One of the intriguing features of scheduling models is the wide variety of objective functions that may be appropriate.

Principle 11.37 | Denoting the start time of job $j = 1, \dots, n$ by x_j , the process time by p_j , the release time by r_j , and the due date by d_j , processor scheduling objective functions often minimize one of the following:

Maximum completion time	$\max_j \{x_j + p_j\}$
Mean completion time	$\frac{1}{n} \sum_j (x_j + p_j)$
Maximum flow time	$\max_j \{x_j + p_j - r_j\}$
Mean flow time	$\frac{1}{n} \sum_j (x_j + p_j - r_j)$
Maximum lateness	$\max_j \{x_j + p_j - d_j\}$
Mean lateness	$\frac{1}{n} \sum_j (x_j + p_j - d_j)$
Maximum tardiness	$\max_j \{\max\{0, x_j + p_j - d_j\}\}$
Mean tardiness	$\frac{1}{n} \sum_j (\max\{0, x_j + p_j - d_j\})$

Maximum completion time is also known as **makespan**.

Total completion time, flow time, lateness, or tardiness may also be of interest, but optimization over each of these is equivalent to optimization over the corresponding mean measure because the total is constant n times the mean.

The completion time measures in principle [11.37](#) emphasize getting all jobs done as soon as possible. For example, the mean completion time version of our Nifty Notes application would have objective function

$$\min \frac{1}{6} [(x_1 + 12) + (x_2 + 8) + (x_3 + 3) + (x_4 + 10) + (x_5 + 4) + (x_6 + 18)]$$

A corresponding optimal schedule has Nifty Notes binding jobs in sequence 3–5–2–4–1–6 with start times

$$x_1^* = 25, \quad x_2^* = 7, \quad x_3^* = 0, \quad x_4^* = 15, \quad x_5^* = 3, \quad x_6^* = 37 \quad (11.25)$$

and mean completion time 23.67.

Completion time measures are particularly appropriate when there are a fixed number of jobs to complete and no more expected. Where the model relates to a more continuing operation, flow time may be more suitable. Flow time tracks the length of time that a job is in the system:

$$\text{flow time} \triangleq (\text{completion time}) - (\text{release time})$$

The idea is to minimize **work in process** so that inventory costs for partially finished goods are reduced.

The third category of measures becomes important when due dates are critical. Lateness counts both early and late jobs:

$$\text{lateness} \triangleq (\text{completion time}) - (\text{due date})$$

Tardiness considers only the late ones (positive lateness):

$$\text{tardiness} \triangleq \max\{0, (\text{lateness})\}$$

For example, minimizing maximum lateness in the Nifty Notes case produces the objective function

$$\min \max\{(x_1 + 12 - 10), (x_2 + 8 - 2), (x_3 + 3 - 72), (x_4 + 10 + 8), (x_5 + 4 + 6), (x_6 + 18 - 60)\} \quad (11.26)$$

The corresponding optimal schedule has Nifty Notes binding jobs in sequence 4–2–5–3–1–6 with start times

$$x_1^* = 22, \quad x_2^* = 14, \quad x_3^* = 52, \quad x_4^* = 0, \quad x_5^* = 10, \quad x_6^* = 34 \quad (11.27)$$

and maximum lateness $(22 + 12 - 10) = 24$ on job 1. Notice that this schedule differs significantly from the mean completion time schedule of (11.25). To decrease lateness, mean completion time has increased from 23.67 to 31.17.

EXAMPLE 11.20: UNDERSTANDING PROCESSOR SCHEDULING OBJECTIVES

The following table shows the process times, release times, due dates, and scheduled start times for three jobs.

	Job 1	Job 2	Job 3
Process time	15	6	9
Release time	5	10	0
Due date	20	25	36
Scheduled start	9	24	0

Compute the corresponding value of each of the eight objective functions in principle [11.37](#).

Solution: Completion times (start + process) are

$$9 + 15 = 24, \quad 24 + 6 = 30, \quad \text{and} \quad 0 + 9 = 9$$

Thus maximum completion time is $\max \{24, 30, 9\} = 30$ and mean completion time is

$$\frac{1}{3}(24 + 30 + 9) = 21$$

Corresponding flow times (completion–release) are

$$24 - 5 = 19, \quad 30 - 10 = 20, \quad \text{and} \quad 9 - 0 = 9$$

Thus maximum flow time is $\max \{19, 20, 9\} = 20$ and mean flow time is $\frac{1}{3}(19 + 20 + 9) = 16$.

Lateness of the three jobs (completion–due date) is

$$24 - 20 = 4, \quad 30 - 25 = 5, \quad \text{and} \quad 9 - 36 = -27$$

Thus maximum lateness is $\max \{4, 5, -27\} = 5$ and mean lateness is $\frac{1}{3}(4 + 5 - 27) = -6$.

Finally, tardiness of the jobs ($\max\{0, \text{lateness}\}$) is

$$\max\{0, 4\} = 4, \quad \max\{0, 5\} = 5, \quad \text{and} \quad \max\{0, -27\} = 0$$

Thus maximum tardiness is $\max \{4, 5, 0\} = 5$ and mean tardiness is $\frac{1}{3}(4 + 5 + 0) = 3$.

ILP Formulation of Minmax Scheduling Objectives

Disjunctive constraints of formulation [11.35](#) can be combined with any of the mean objective forms in list [11.36](#) except tardiness to obtain an integer linear programming formulation of a processor scheduling problem. When tardiness or any of the minmax objectives are being optimized, however, the problem is an integer nonlinear program (INLP).

We can convert any of these INLP forms to the more tractable ILP by using the techniques of Section 4.6 (principle [4.13](#)).

Principle 11.38 Any of the min max objective of list [11.37](#) can be linearized by introducing a new decision variable f to represent the objective function value, then minimizing f subject to new constraints of the form $f \geq$ each element in the maximize set. A similar construction can model tardiness by introducing new nonnegative tardiness variables for each job and adding constraints keeping each tardiness variable \geq the corresponding lateness.

To illustrate, return to the Nifty Notes lateness objective of expression (11.26):

$$\min \max \{(x_1 + 12 - 10), (x_2 + 8 - 2), (x_3 + 3 - 72), (x_4 + 10 + 8), \\ (x_5 + 4 + 6), (x_6 + 18 - 60)\}$$

We may convert to ILP form by introducing a new variable f and then solving

$$\begin{array}{ll}
 \min & f \\
 \text{s.t.} & f \geq x_1 + 2 \\
 & f \geq x_2 + 6 \\
 & f \geq x_3 - 69 \\
 & f \geq x_4 + 18 \\
 & f \geq x_5 + 10 \\
 & f \geq x_6 - 42 \\
 & \text{(all original constraints)}
 \end{array}$$

EXAMPLE 11.21: LINEARIZING SCHEDULING OBJECTIVES

Using x_j to represent the scheduled start time of each job j in Example 11.20, show how each of the following objective functions can be expressed in ILP format.

(a) Maximum completion

(b) Mean tardiness

Solution: We apply construction [11.38](#).

(a) To linearize maximum completion time, we introduce a new decision variable f and enforce new constraints to keep it as great as any completion time. Specifically, the formulation is

$$\begin{array}{ll}
 \min & f \\
 \text{s.t.} & f \geq x_1 + 15 \\
 & f \geq x_2 + 6 \\
 & f \geq x_3 + 9 \\
 & \text{(all original constraints)}
 \end{array}$$

(b) To model tardiness, we introduce new decision variables $t_j \geq 0$ for each job j and force it to be \geq lateness. Then the mean tardiness model is

$$\begin{array}{ll}
 \min & \frac{1}{3}(t_1 + t_2 + t_3) \\
 \text{s.t.} & t_1 \geq x_1 + 15 - 20 \\
 & t_2 \geq x_2 + 6 - 25 \\
 & t_3 \geq x_3 + 9 - 36 \\
 & t_1, t_2, t_3 \geq 0 \\
 & \text{(all original constraints)}
 \end{array}$$

If job j is late, the corresponding new main constraint makes $t_j =$ lateness. If not, nonnegativity constraints $t_j \geq 0$ force tardiness = 0.

Equivalences among Scheduling Objective Functions

Different objective functions from list [11.37](#) do not always imply different optimal schedules.

Principle 11.39 | The mean completion time, mean flow time, and mean lateness scheduling objective functions are equivalent in the sense that an optimal schedule for one is also optimal for the others.

Principle 11.40 | An optimal schedule for the maximum lateness objective function is also optimal for maximum tardiness.

For example, the optimal mean completion time schedule (11.25) of our Nifty Notes application is also optimal for the mean flow time and mean lateness objective functions (principle [11.39](#)). Schedule (11.27), which minimized maximum lateness, also minimizes maximum tardiness (principle [11.40](#)).

To see why mean completion time and mean flow time are equivalent, we need only rearrange defining sums:

$$\begin{aligned} \text{mean flow time} &= \frac{1}{n} \sum_{j=1}^n (x_j + p_j - r_j) \\ &= \frac{1}{n} \sum_{j=1}^n (x_j + p_j) - \frac{1}{n} \sum_{j=1}^n r_j \\ &= (\text{mean completion time}) - \frac{1}{n} \sum_{j=1}^n r_j \end{aligned}$$

Expressed this way, it is apparent that the objective functions differ only by the constant last term. Adding or subtracting such a constant to the objective function cannot change what solutions are optimal. Similar arguments equate mean completion time and mean lateness.

Connection [11.40](#) between maximum lateness and maximum tardiness is also straightforward. If at least one job must be late in every schedule, maximum lateness = maximum tardiness. If no job has to be late, all schedules are optimal for the maximum tardiness objective, including any optimal for maximum lateness.

Job Shop Scheduling

In contrast to the single-processor case of definition [11.32](#), job shop scheduling involves jobs that must be processed on several different machines.

Definition 11.41 | **Job shop scheduling problems** seek an optimal schedule for a given collection of jobs, each of which requires a known sequence of processors that can accommodate only one job at a time.

APPLICATION 11.13: CUSTOM METALWORKING JOB SHOP

We illustrate job shop scheduling with a fictitious Custom Metalworking company which fabricates prototype metal parts for a nearby engine manufacturer. Figure 11.9 provides details on the 3 jobs waiting to be scheduled. First is a die requiring work on a sequence of 5 workstations: 1 (forging), then 2 (machining), then 3 (grinding), then 4 (polishing), and finally, 6 (electric discharge cutting). Job 2 is a cam shaft requiring 4 stations, and job 3 a fuel injector requiring 5 steps. Numbers in boxes indicate process times

$$p_{j,k} \triangleq \text{process time (in minutes) of job } j \text{ on processor } k$$

For example, job 1 requires 45 minutes at polishing workstation 4.

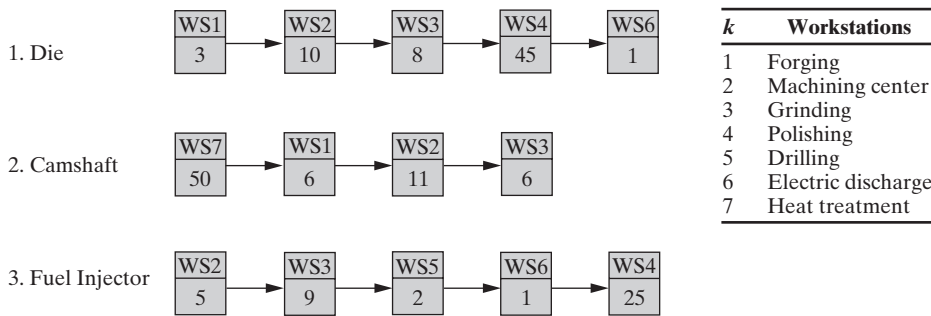


FIGURE 11.9 Custom Metalworking Application Jobs

Any of the objective function forms in list [11.37](#) could be appropriate for Custom Metalworking’s scheduling. We will assume that the company wants to complete all 3 jobs as soon as possible (minimize maximum completion), so that workers can leave for a holiday.

Custom Metalworking Application Decision Variables and Objective

Job shop scheduling involves deciding when to start each step of each job on its processor. Thus start time decision variables [11.33](#) are now indexed by both job and processor:

$$x_{j,k} \triangleq \text{start time of job } j \text{ on processor } k$$

Our assumed makespan scheduling objective can then be expressed as

$$\min \max \{x_{1,6} + 1, x_{2,3} + 6, x_{3,4} + 25\}$$

Notice that only the last step of each job is reflected. Completion of a multiprocessor job means completion of all steps.

Precedence Constraints

Steps of the various jobs being scheduled in a job shop must take place in the sequence given. That is, start times are subject to **precedence constraints**.

Principle 11.42 The precedence requirement that job j must complete on processor k before activity on k' begins can be expressed as

$$x_{j,k} + p_{j,k} \leq x_{j,k'}$$

where $x_{j,k}$ denotes the start time of job j on processor k , $p_{j,k}$ is the process time of j on k , and $x_{j,k'}$ is the start time of job j on processor k' .

Job shop models have precedence constraints [11.42](#) between each step and its successor in each job. For instance, job 1 in Figure 11.9 implies precedence constraints

$$\begin{aligned}x_{1,1} + 3 &\leq x_{1,2} \\x_{1,2} + 10 &\leq x_{1,3} \\x_{1,3} + 8 &\leq x_{1,4} \\x_{1,4} + 45 &\leq x_{1,6}\end{aligned}$$

to maintain the required processing sequence.

EXAMPLE 11.22: FORMULATING JOB SHOP PRECEDENCE CONSTRAINTS

A job shop must schedule product 1, which requires 12 minutes on machine 1 followed by 30 minutes on machine 2, and product 2, which requires 17 minutes on machine 1 followed by 29 minutes on machine 3. Formulate the implied precedence constraints in terms of the decision variables

$$x_{j,k} \triangleq \text{start time of product } j \text{ on machine } k$$

Solution: There is one precedence constraint for each job because each has only two steps. In accord with [11.42](#), those constraints are

$$\begin{aligned}x_{1,1} + 12 &\leq x_{1,2} \\x_{2,1} + 17 &\leq x_{2,3}\end{aligned}$$

Conflict Constraints in Job Shops

As in one machine case such as Nifty Notes, job shop models must also confront the possibility of conflicts—jobs scheduled simultaneously on the same processor. For example, in the Custom Metalworking application of Figure 11.9, jobs 1 and 2 may conflict at workstation 1, which both require. One must complete before the other can begin.

Paralleling [11.34](#) and [11.35](#), we may model conflicts by introducing the new discrete decision variables

$$y_{j,j',k} \triangleq \begin{cases} 1 & \text{if } j \text{ is scheduled before job } j' \text{ on processor } k \\ 0 & \text{otherwise} \end{cases}$$

Principle 11.43 Job shop models can prevent conflicts between jobs by introducing new disjunctive variables $y_{j,j',k}$ and constraint pair

$$\begin{aligned}x_{j,k} + p_{j,k} &\leq x_{j',k} + M(1 - y_{j,j',k}) \\x_{j',k} + p_{j',k} &\leq x_{j,k} + My_{j,j',k}\end{aligned}$$

for each j, j' that both require any processor k . Here $x_{j,k}$ denotes the start time of job j on processor k , $p_{j,k}$ its process time, M is a large positive constant, and binary $y_{j,j',k} = 1$ when j is scheduled before j' on k and $= 0$ if j' is first.

For instance, the possible conflict between Custom Metalworking jobs 1 and 2 at workstation 1 produces constraint pair

$$\begin{aligned}x_{1,1} + 6 &\leq x_{2,1} + M(1 - y_{1,2,1}) \\x_{2,1} + 3 &\leq x_{1,1} + My_{1,2,1}\end{aligned}$$

If job 1 uses the processor first, $y_{1,2,1} = 1$, and the first constraint is enforced. If job 2 comes first, $y_{1,2,1} = 0$, and the second constraint controls.

EXAMPLE 11.23: FORMULATING JOB SHOP CONFLICT CONSTRAINTS

Return to the job shop of Example 11.22 and formulate all constraints required to prevent conflicts.

Solution: Conflicts can occur only on machine 1, which is the only one required for both products. Thus we require only one binary variable,

$$y_{1,2,1} \triangleq \begin{cases} 1 & \text{if product 1 is first on machine 1} \\ 0 & \text{if product 2 is first on machine 1} \end{cases}$$

and the needed constraints [11.43](#) are

$$\begin{aligned}x_{1,1} + 12 &\leq x_{2,1} + M(1 - y_{1,2,1}) \\x_{2,1} + 17 &\leq x_{1,1} + My_{1,2,1}\end{aligned}$$

Here $M = 12 + 17 = 29$ would be large enough to have the desired effect of discounting whichever constraint should not really apply.

Custom Metalworking Application Model

Combining our maximum completion time objective function with all required precedence and conflict constraints produces the following complete model of the Custom Metalworking application in Figure 11.9:

$$\begin{aligned}\min \quad & \max\{x_{1,6} + 1, x_{2,3} + 6, x_{3,4} + 25\} && \text{(maximum completion)} \\ \text{s.t.} \quad & x_{1,1} + 3 \leq x_{1,2} && \text{(job 1 precedence)} \\ & x_{1,2} + 10 \leq x_{1,3}\end{aligned}$$

$$\begin{aligned}
x_{1,3} + 8 &\leq x_{1,4} \\
x_{1,4} + 45 &\leq x_{1,6} \\
x_{2,7} + 50 &\leq x_{2,1} && \text{(job 2 precedence)} \\
x_{2,1} + 6 &\leq x_{2,2} \\
x_{2,2} + 11 &\leq x_{2,3} \\
x_{3,2} + 5 &\leq x_{3,3} && \text{(job 3 precedence)} \\
x_{3,3} + 9 &\leq x_{3,5} \\
x_{3,5} + 2 &\leq x_{3,6} \\
x_{3,6} + 1 &\leq x_{3,4} \\
x_{1,1} + 6 &\leq x_{2,1} + M(1 - y_{1,2,1}) && \text{(workstation 1 conflicts)} \\
x_{2,1} + 3 &\leq x_{1,1} + My_{1,2,1} \\
x_{1,2} + 10 &\leq x_{2,2} + M(1 - y_{1,2,2}) && \text{(workstation 2 conflicts)} \\
x_{2,2} + 11 &\leq x_{1,2} + My_{1,2,2} \\
x_{1,2} + 10 &\leq x_{3,2} + M(1 - y_{1,3,2}) \\
x_{3,2} + 5 &\leq x_{1,2} + My_{1,3,2} \\
x_{2,2} + 11 &\leq x_{3,2} + M(1 - y_{2,3,2}) && (11.28) \\
x_{3,2} + 5 &\leq x_{2,2} + My_{2,3,2} \\
x_{1,3} + 8 &\leq x_{2,3} + M(1 - y_{1,2,3}) && \text{(workstation 3 conflicts)} \\
x_{2,3} + 6 &\leq x_{1,3} + My_{1,2,3} \\
x_{1,3} + 8 &\leq x_{3,3} + M(1 - y_{1,3,3}) \\
x_{3,3} + 9 &\leq x_{1,3} + My_{1,3,3} \\
x_{2,3} + 6 &\leq x_{3,3} + M(1 - y_{2,3,3}) \\
x_{3,3} + 9 &\leq x_{2,3} + My_{2,3,3} \\
x_{1,4} + 45 &\leq x_{3,4} + M(1 - y_{1,3,4}) && \text{(workstation 4 conflicts)} \\
x_{3,4} + 25 &\leq x_{1,4} + My_{1,3,4} \\
x_{1,6} + 1 &\leq x_{3,6} + M(1 - y_{1,3,6}) && \text{(workstation 6 conflicts)} \\
x_{3,6} + 1 &\leq x_{1,6} + My_{1,3,6} \\
\text{all } x_{j,k} &\geq 0 \\
\text{all } y_{j',k} &= 0 \text{ or } 1
\end{aligned}$$

An optimal solution uses start times

$$\begin{aligned}
x_{1,1}^* &= 2, & x_{1,2}^* &= 5, & x_{1,3}^* &= 15, & x_{1,4}^* &= 42, & x_{1,6}^* &= 87 \\
x_{2,7}^* &= 0, & x_{2,1}^* &= 50, & x_{2,2}^* &= 56, & x_{2,3}^* &= 67 \\
x_{3,2}^* &= 0, & x_{3,3}^* &= 5, & x_{3,5}^* &= 14, & x_{3,6}^* &= 16, & x_{3,4}^* &= 17
\end{aligned}$$

to complete all jobs in 88 minutes.

EXERCISES

11-1 A fertilizer plant can make a product by any of 3 processes. Using decision variables $x_j \triangleq$ number of units produced by process j , the following linear program computes a minimum cost way to produce 150 units with available resources:

$$\begin{aligned} \min \quad & 15x_1 + 11x_2 + 18x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 150 \\ & 2x_1 + 4x_2 + 2x_3 \leq 310 \\ & 4x_1 + 3x_2 + x_3 \leq 450 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

- (a) Explain why this LP implicitly assumes that objective function coefficients are variable costs.
- (b) Use class optimization software to solve the given LP.
- (c) Formulate a revised ILP model implementing a requirement that only one of the 3 activities may be used.
- (d) Use class optimization software to solve the ILP of part (c).
- (e) Formulate a different revised ILP implementing a fixed setup cost of 400 charged for each activity used at all.
- (f) Use class optimization software to solve the ILP of part (e).
- (g) Formulate another revised ILP implementing a requirement that an activity can be used only if a minimum of 50 units are produced.
- (h) Use class optimization software to solve the ILP of part (g).

11-2 A computer distributor can purchase workstations from any of 3 suppliers. Using decision variables $x_j \triangleq$ number of units purchased from supplier j , the following linear program computes a minimum cost way to purchase 300 workstations within applicable limits:

$$\begin{aligned} \min \quad & 5x_1 + 7x_2 + 6.5x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 = 300 \\ & 3x_1 + 5x_2 + 4x_3 \leq 1500 \\ & 0 \leq x_1 \leq 200 \\ & 0 \leq x_2 \leq 300 \\ & 0 \leq x_3 \leq 200 \end{aligned}$$

Do (a) through (h) as in Exercise 11-1 using a setup cost of 100 and a minimum purchase of 125.

11-3 A retired executive has up to \$8 million that he wishes to invest in apartment buildings. The following table shows the purchase price and the expected 10-year return (in millions of dollars) of the 4 buildings that he is considering.

	Building			
	1	2	3	4
Price	4.0	3.8	6.0	7.2
Return	4.5	4.1	8.0	7.0

The executive wishes to choose investments that maximize his total return. Assume that every option is available only on an all-or-nothing basis.

- (a) Formulate a knapsack ILP to choose an optimal investment plan.
- (b) Solve your knapsack by inspection.

11-4 The River City redevelopment authority wants to add a minimum of one thousand new parking spaces in the downtown area. The following table shows the estimated cost (in millions of dollars) of the 4 proposed projects and the number of spaces each would yield (in hundreds).

	Project			
	1	2	3	4
Cost	16	9	11	13
Spaces	8	3	6	6

The authority wants to meet its goal at minimum total cost. Assume that every project is available only on an all-or-nothing basis.

- (a) Formulate a knapsack ILP to choose an optimal parking program.
- (b) Solve your knapsack by inspection.

11-5 Silo State's School of Engineering is preparing a 5-year plan for building construction and expansion to accommodate new offices, laboratories, and classrooms. The Electrical Engineering

faculty has proposed projects for all 3 available parcels of land: a digital circuits lab on the northwest parcel at \$48 million, a faculty office annex on the southeast parcel at \$20.8 million, and a computer vision lab on the northeast parcel at \$32 million. The Mechanical Engineering faculty has 3 alternative proposals for the same northwest parcel: a large lecture room building at \$28 million, a heat transfer lab at \$44 million, and a computer-aided design expansion at \$17.2 million. The Industrial Engineering faculty has only 2 proposals: a manufacturing research center on the southeast parcel at \$36.8 million, and a tunnel from their current building to the new center for an additional \$1.2 million. The Dean of Engineering scores the impact of these projects as 9, 2, and 10 for the EE proposals, 2, 5, and 8 for the ME alternatives, 10 and 1 for the IE ideas. He wishes to allocate his available \$100 million to maximize the total impact, selecting projects on an all-or-nothing basis with at most one per land parcel.

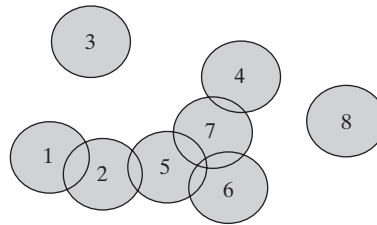
- ✔ (a) Describe verbally the budget limits, mutual exclusiveness constraints, and project dependency requirements of this capital budgeting problem.
- ✔ (b) Formulate a capital budgeting ILP to select an optimal combination of proposals.
- ✔ (c) Use class optimization software to solve your ILP model.

11-6 A small pharmaceutical research laboratory must decide which product research activities to undertake with its \$25 million available in each of the next two 5-year periods. The products optamine and feasibine are ready for field testing in the first 5-year period at a cost of \$13 million and \$14 million, respectively. Discretol and zeronex are at an earlier, development stage. Proposed development activities would require \$4 million in the first 5 years for discretol and \$3 million in the second. Corresponding values for zeronex are \$2 million and \$6 million. Field testing of the two products may also be chosen in the second 5-year period for \$10 million and \$15 million, respectively, if the corresponding development activity was undertaken in the first 5 years. The company wishes to maximize future profits from field-tested products, which it estimates at \$510 million for optamine, \$640 million for feasibine, \$580 million for discretol, and \$469 million

for zeronex. All projects must be adopted on an all-or-nothing basis, and no more than one of the two development projects may be selected.

Do (a) through (c) as in Exercise 11-5.

11-7 The map that follows shows the locations of 8 applicants for low-power radio station licences and the approximate range of their signals.



Regulators have scored the quality of applications on a scale of 0 to 100 as 45, 30, 84, 73, 80, 70, 61, and 91, respectively. They wish to select the highest-quality combination of applications that has no overlap in signal ranges.

- ✔ (a) Formulate this problem as a set packing ILP.
- ✔ (b) Use class optimization software to solve your ILP.

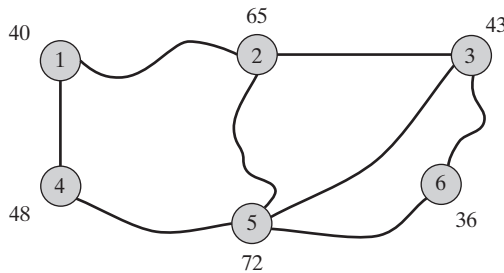
11-8 Time Sink Incorporated is about to begin selling its computer game software to college students in the midwest. The following table shows the states that could be covered by salespersons based at 4 possible locations, together with the annual sales (in thousands of dollars) expected if all those states were covered from that base.

	Base			
	Ames	Beloit	Normal	Avon
MN	×	×	—	—
IA	×	×	×	—
MO	×	—	×	—
WI	—	×	×	—
IN	—	—	×	×
KY	—	—	—	×
Sales	115	90	150	126

Time Sink wants to choose a collection of bases that maximizes total sales without assigning the same state to more than one base.

Do (a) and (b) as in Exercise 11-7.

11-9 The following map shows the 8 intersections at which automatic traffic monitoring devices might be installed. A station at any particular node can monitor all the road links meeting that intersection. Numbers next to nodes reflect the monthly cost (in thousands of dollars) of operating a station at that location.



- (a) Formulate the problem of providing full coverage at minimum total cost as a set covering ILP.
- (b) Use class optimization software to solve your ILP of part (a).
- (c) Revise your formulation of part (a) to obtain an ILP minimizing the number of uncovered road links while using at most 2 stations.
- (d) Use class optimization software to solve your ILP of part (c).

11-10 Top Tool Company wishes to hire part-time models to pass out literature about its machine tools at an upcoming trade show. Each of the 6 available models can work 2 of the 5 show days: Monday–Tuesday, Monday–Wednesday, Monday–Friday, Tuesday–Wednesday, Tuesday–Friday, and Thursday–Friday. If there is no more than one day separating their duty days, models will be paid \$300, but the charge is \$500 if two or more days intervene. Top Tool seeks a minimum cost way to have a least one model on all 5 days.

Do (a) through (d) as in Exercise 11-9 with the revised model minimizing the number of days uncovered using just 2 models.

11-11 Air Anton is a small commuter airline running 6 flights per day from New York City to surrounding resort areas. Flight crews are all based in New York, staffing flights to various locations and then returning on the next flight home. Taking into account complex work rules and pay incentives, Air Anton schedulers

have constructed the 8 possible work patterns detailed in the following table. Each row of the table marks the flights that could be covered in a particular pattern and the daily cost per crew (in thousands of dollars).

Work Pattern	Flight						Cost
	1	2	3	4	5	6	
1	–	×	–	×	–	–	1.40
2	×	–	–	–	–	×	0.96
3	–	×	–	×	×	–	1.52
4	–	×	–	–	×	×	1.60
5	×	–	×	–	–	×	1.32
6	–	–	×	–	×	–	1.12
7	–	–	–	×	–	×	0.84
8	×	–	×	×	–	–	1.54

The company wants to choose a minimum total cost collection of work patterns that covers all flights exactly once.

- (a) Formulate this problem as a set partitioning ILP.
- (b) Use class optimization software to solve your ILP.

11-12 A special court commission appointed to resolve a bitter fight over legislative redistricting has proposed 6 combinations of the 5 disputed counties that could form new districts. The following table marks the counties composing each proposed district and shows the district’s deviation from the equal-population norm.

Country	District					
	1	2	3	4	5	6
1	×	–	–	–	×	×
2	×	–	×	–	–	×
3	–	–	×	×	×	–
4	–	×	–	×	–	–
5	–	×	–	–	–	×
Deviation	0.5	0.5	0.6	1.3	0.7	1.2

The court wants to select a minimum total deviation collection of districts that includes each county exactly once.

Do (a) and (b) as in Exercise 11-11.

11-13 Mogul Motors is planning a major overhaul of its automobile assembly plants as it introduces 4 new models. Exactly one existing plant

must be converted to assemble each model. The following table shows for each model and plant the cost (in millions of dollars) of modifications at the plant to produce the model. Those marked with a dash reflect plants not large enough to accommodate the needed activity.

Model	Plant			
	1	2	3	4
1	18	26	—	31
2	—	50	22	—
3	40	29	52	39
4	—	—	43	46

Mogul seek a minimum total cost way to make the conversion.

- ✓ (a) Formulate this problem as a linear assignment LP.
- ✓ (b) Explain why a binary optimal solution is guaranteed even though your model is a linear program.
- ✓ (c) Use class optimization software to solve your assignment LP.

11-14 The sister communities program pairs cities in Russia with cities in the United States that have a similar size and economic base. Visits are then exchanged between the sister communities to improve international understanding. The following table shows the program’s compatibility scores (0 to 100) for the 4 U.S. and 4 Russian cities about to join the program.

U.S.	Russian			
	1	2	3	4
1	80	65	83	77
2	54	87	61	66
3	92	45	53	59
4	70	61	81	76

Sister communities seeks a maximum total compatibility pairing.

Do (a) through (c) as in Exercise 11-13.

11-15 The following table shows the unit price and the minimum quantity at which suppliers $j = 1 \dots, 5$ have to bid to supply State University (SU) with office desk chairs. SU wishes to find the lowest total cost combination of purchases that will procure at least the minimums from each

supplier used, and will obtain a total of at least 400 chairs.

Supplier	1	2	3	4	5
Unit Price	200	400	325	295	260
Min Quantity	500	50	100	100	250

- (a) Although SU must obviously buy integer numbers of chairs from each supplier, explain why it makes sense to model the number of chairs purchased from each supplier as a nonnegative, continuous variable. Also indicate what issues are left that make the problem discrete.
- (b) Using nonnegative continuous x_j for the number of chairs purchased from supplier j , and whatever other decision variables are required, formulate SU’s problem as a Mixed-Integer Linear Program. Be sure to define any additional decision variables and annotate each objective function and constraint with its meaning.

11-16. Focus Inc. wants to consolidate its camera manufacturing operations at 2 of the current 4 plants. At the same time, each surviving plant will begin full 3-shift operations running 168 hours per week. The following table shows the estimated cost (in \$ million) of moving operations from each current plant to each other, along with the projected number of production hours per week the moved operation would add at the new site. For example, closing operation 1 and moving it to site 3 would cost \$320 million and add 70 hours per week at site 3.

From Site		To Site			
		1 = Omaha	2 = Denver	3 = Muncie	4 = Kent
1 = Omaha	Cost	0	450	320	550
	Hours	56	56	70	56
2 = Denver	Cost	770	0	640	690
	Hours	82	82	70	70
3 = Muncie	Cost	810	770	0	660
	Hours	40	40	60	60
4 = Kent	Cost	580	610	490	0
	Hours	56	56	56	56

Formulate Focus Inc.’s problem of choosing which two sites to retain and how to move the others to minimize total moving cost as an Integer Linear

Program. Use decision variables $x_{ij} = 1$ if plant i is moved to site j and $= 0$ otherwise, so that $x_{ii} = 1$ means the plant will be retained. Be sure to annotate objectives and constraints to indicate their meaning.

11-17 Channel 999 TV has staff to provide on-scene coverage of up to 4 high school football games this Friday. The following table shows 3 possibilities are in the town where Channel 999 is located. At least 2 of them must be covered, as well as at least 1 out of town. The table also shows that 4 of the games involve a team likely to compete for the state championship, at least 2 of which must be covered. Games must be fully covered or not. Within these requirements Channel 999 wants to maximize its total audience across the ratings points show for possible game choices.

Game Number	1	2	3	4	5	6	7	8
In Town?	Y		Y		Y			
State Champ?		Y		Y	Y			Y
Ratings Points	3.0	1.7	2.6	1.8	1.5	5.3	1.6	2.0

(a) Formulate a pure ILP to compute optimal choice of games to cover. Be sure to define your decision variables and briefly annotate the objective function, and each (main or variable-type) constraint with a few words indicating its meaning.

(b) Use class optimization software to compute an optimal choice.

11-18 Erika Entrepreneur assembles laptop computer systems in her home to finance her graduate education. She makes and sells two types of units, both using the same frame. The deluxe model has 1024 megabytes of RAM memory, a 16 gigabyte hard drive, and a communication card; it sells for \$1400 per unit. The cheaper basic model, which sells for \$1000, has only 512 megabytes of RAM memory, a 4 gigabyte hard drive, and no communication card. RAM in both models is built up by installing the requisite number of 256 megabyte chips.

Component	256 MB		16 GB	4 GB	Comm
	Frames	Chips	Drives	Drives	Cards
On Hand	18	72	7	11	3
Min Purchase	5	48	10	8	3
Max Purchase	40	182	30	64	25
Price/Unit	\$700	\$75	\$300	\$110	\$250

The table shows the number on hand at the beginning of the current month for each of the components Erika uses, along with the minimum and maximum she can buy if she makes a new purchase, and the unit price that would apply. Note that her suppliers do not allow Erika to buy fewer than the minimum nor more than the maximum shown, and only one purchase per month is allowed. Within these limits Erika wants to decide a production and procurement plan that will maximize her gross profit (sales - cost) for the coming month.

- (a) Although the numbers of components used and purchased must obviously be integer, explain why it would be good OR modeling to represent them as nonnegative continuous decision variables. Also indicate what issues are left that make the problem discrete.
- (b) Formulate Erika’s problem as a Mixed-Integer Linear Program. Be sure to define all your decision variables and annotate objectives and constraints to show their meaning.

11-19 Sandbox State University is rearranging the locations of 3 equal-sized academic departments to provide for better faculty communication. The following tables show the estimated number of person-to-person contacts per month between members of the various faculties, and the distances (in thousands of feet) between available office locations.

	Interaction	
	English	Math
History	20	12
English	—	14

	Distance	
	2	3
1	3	6
2	—	1

Sandbox State wants to place one department in each location in a way that minimizes total distance traveled for faculty interactions.

- (a) Formulate this problem as a quadratic assignment INLP.

- ✔ (b) Explain why the objective function in this problem must be quadratic rather than linear.
- ✔ (c) Compute an optimal assignment in your INLP by inspection.

11-20 The River City Operations Research society is planning a meeting that will have 2 morning and 2 afternoon sessions running at the same time. The 4 sessions will be on LP, NLP, ILP, and INLP, respectively, but times have not yet been fixed. The following table shows the estimated number of attendees who would like to be able to attend both of each combination of sessions.

	NLP	ILP	INLP
LP	10	30	14
NLP	—	5	8
ILP	—	—	18

The society would like to arrange sessions to minimize the number of persons who cannot attend a desired pair of sessions because they occur simultaneously.

Do (a) through (c) as in Exercise 11-19.

11-21 A warehouse facility has packing stations at both its front and back entrances. The following table shows the number of ton-feet (in thousands) of materials handling that would be required to move each of the 6 pending jobs to either of the 2 stations, along with the number of hours packing that would be required at whichever station does the work.

	Job					
	1	2	3	4	5	6
Front	21	17	10	30	40	22
Back	13	18	29	24	33	29
Time	44	60	51	80	73	67

Schedulers seek a minimum handling plan that completes all packing within the 200 hours available at the front station and 190 hours available at the back. Assume that jobs must go entirely to a single packing station.

- ✔ (a) Formulate this problem as a generalized assignment ILP.

- ✔ (b) Explain why this is a generalized rather than an ordinary assignment problem.
- ✔ (c) Use class optimization software to solve your ILP.

11-22 Three professional baseball teams are trying to find places for 6 available players within their remaining salary limits of \$35 million, \$20 million, and \$26 million, respectively. The following table shows how valuable each player would be to each team on a scale of 0 to 10, and the player’s current annual salary (in millions of dollars).

Player	Value			Salary
	1	2	3	
1	8	7	10	10
2	7	8	6	13
3	5	4	6	8
4	6	3	3	6
5	8	7	6	15
6	10	9	10	22

The teams want to find a maximum total score allocation of players to teams that fits with in salary limits.

Do (a) through (c) as in Exercise 11-21.

11-23 Small business Cool Room Furniture (CRF) sells whole-room packages of furniture in the region around its single store and warehouse. Each day it must decide how to assign needed deliveries $i = 1, \dots, m$ to available trucks $j = 1, \dots, n$ to fulfill commitments of timely delivery to customers at minimum total cost. Each delivery i requires a whole-truck round trip from warehouse to customer and back taking t_i hours and costing $c_{j,i}$ dollars. But the same truck j may make multiple deliveries i if the total of their trip times is at most the trucks available for regular work hours a_j . It is also possible for any truck to work up to an additional 4 hours of overtime at q_j per hour.

The following table shows values of these parameters for a typical day with $m = 9$ loads and $n = 5$ trucks.

- (a) Formulate CRF’s problem as a generalized assignment type ILP with extensions to allow buying overtime.

	Deliveries i									Avail a_j	Otime q_j
	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$	$i = 8$	$i = 9$		
Time t_i	4	6	2	3	7	1	4	3	9		
Trucks j	Trip Costs $c_{j,i}$										
$j = 1$	210	50	89	115	151	77	40	160	145	8	50
$j = 2$	150	40	69	95	131	57	30	120	125	6	70
$j = 3$	210	50	89	115	151	77	40	160	145	8	50
$j = 4$	150	40	69	95	131	57	30	120	125	6	70
$j = 5$	190	45	79	105	141	67	35	140	135	8	60

(b) Use class optimization software to compute an optimal plan.

11-24 Military commanders are planning the command structure for 6 new radar stations. Three commanders will each be in charge of two of the stations. The following table shows the projected cost (in millions of dollars) of building the necessary communication links to connect jointly commanded locations.

	2	3	4	5	6
1	42	65	29	31	55
2	—	20	39	40	21
3	—	—	68	55	22
4	—	—	—	30	39
5	—	—	—	—	47

Planners seek a minimum cost way to organize the command.

- (a) Formulate this problem as a matching ILP.
- (b) Explain why this is not an assignment problem.
- (c) Use class optimization software to solve your ILP.

11-25 Awesome Advertising manages the television promotion of a variety of products. In the next few months they are planning to cross-advertise six of their items by running interlocking television ads that mention both products. The following table shows Awesome’s estimate of the number of viewers (in millions) who might be interested jointly in each pair of products.

	2	3	4	5	6
1	7	8	6	14	15
2	—	18	20	5	8
3	—	—	19	9	10
4	—	—	—	6	11
5	—	—	—	—	16

Awesome wants to find a product pairing that maximizes the appeal, with each product in exactly one pair.

Do (a) through (c) as in Exercise 11-24.

11-26 Engineers are designing a fixed route to be followed by automatic guided vehicles in a large manufacturing plant. The following table shows the east–west and north–south coordinates of the 6 stations to be served by vehicles moving continuously around the same route.

	1	2	3	4	5	6
E/W	20	40	180	130	160	50
N/S	90	70	20	40	10	80

Since traffic must move along east–west or north–south aisles, designers seek a route of shortest total rectilinear length (see Section 4.6).

- (a) Explain why this problem can be viewed as a traveling salesman problem.
- (b) Explain why distances in this problem are symmetric, and compute a matrix of rectilinear distances between all pairs of points.
- (c) Formulate this problem (incompletely) as an ILP with main constraints requiring only that every point be touched by 2 links of the route.
- (d) Use class optimization software to show that your ILP of part (c) produces a subtour 1–2–6–1.
- (e) Formulate a subtour elimination constraint that precludes the solution of part (d).
- (f) Use class optimization software to show that an optimal route results when your subtour elimination constraint is added to the formulation of part (c).
- (g) Formulate this problem as a quadratic assignment INLP.

11-27 An oil company currently has 5 platforms drilling off the Gulf coast of the United States. The following table shows the east–west and north–south coordinates of their shore base at point 0 and all the platform locations.

	0	1	2	3	4	5
E/W	80	10	60	30	85	15
N/S	95	15	70	10	75	30

Each day a helicopter delivers supplies by flying from the base to all platforms and then returning to base. Supervisors seek the route of shortest total length.

Do (a) through (g) as in Exercise 11-26 using straightline (Euclidean) distance and subtour 0–2–4–0.

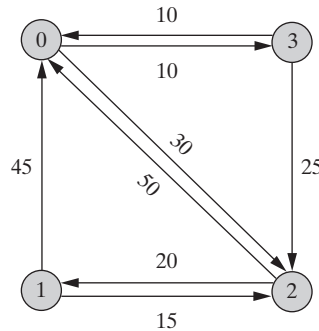
11-28 Every week Mighty Mo Manufacturing makes one production run of each of its 4 different kinds of metal cookware. Setup times to make any particular product vary depending on what was produced most recently. The following table shows the time (in hours) required to convert from any product to any other.

	1	2	3	4
1	—	4.2	1.5	6.5
2	5.0	—	8.5	1.0
3	1.2	7.7	—	8.0
4	5.5	1.8	6.0	—

Mighty Mo would like to find the production sequence that minimizes total setup time.

- ✓ (a) Explain why this problem can be viewed as an asymmetric traveling salesman problem.
- ✓ (b) Formulate this problem (incompletely) as a linear assignment problem to choose a successor for each product.
- ✓ (c) Use class optimization software to show that your ILP of part (b) produces a subtour 1–3–1.
- ✓ (d) Formulate a subtour elimination constraint that precludes the solution of part (c).
- ✓ (e) Use class optimization software to show that an optimal route results when your subtour elimination constraint is added to the formulation of part (b).
- ✓ (f) Formulate this problem as a quadratic assignment INLP.

11-29 Every weekday afternoon, at the height of the rush hour, a bank messenger drives from the a bank’s central office to its 3 branches and returns with noncash records of the day’s activity. The following figure shows the freeway routes that are not hopelessly clogged by traffic at that hour and the estimated driving time for each (in minutes).



The bank would like to find a route that minimizes total travel time.

Do (a) through (f) as in Exercise 11-28 with subtour 0–3–0.

11-30 Gotit Grocery Company is considering 3 locations for new distribution centers to serve its customers in 4 nearby cities. The following table shows the fixed cost (in millions of dollars) of opening each potential center, the number (in thousands) of truckloads forecasted to be demanded at each city over the next 5 years, and the transportation cost (in millions of dollars) per thousand truckloads moved from each center location to each city.

Center	Fixed Cost	City			
		1	2	3	4
1	200	6	5	9	3
2	400	4	3	5	6
3	225	5	8	2	4
Demand	—	11	18	15	25

Gotit seeks a minimum cost distribution system assuming any distribution center can meet any or all demands.

- ✓ (a) Formulate this problem as a facilities location ILP.
- ✓ (b) Use class optimization software to solve your ILP.

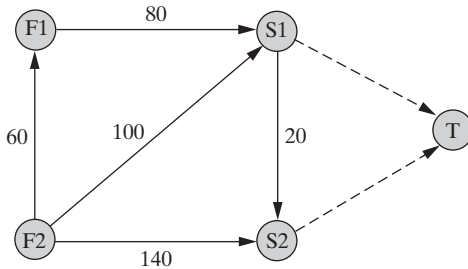
11-31 Basic Box Company is considering 5 new box designs of different sizes to package 4 upcoming lines of computer monitors. The following table shows the wasted space that each box would have if used to package each monitor. Missing values indicate a box that cannot be used for a particular monitor.

Box	Monitor			
	1	2	3	4
1	5	—	10	—
2	20	—	—	25
3	40	—	40	30
4	—	10	70	—
5	—	40	80	—

Basic wants to choose the smallest number of box designs needed to pack all products and to decide which box design to use for each monitor, to minimize waste.

Do (a) and (b) as in Exercise 11-30. (*Hint:* Use a large positive constant for fixed charges.)

11-32 The figure that follows shows 5 pipelines under consideration by a natural gas company to move gas from its 2 fields to its 2 storage areas. The numbers on the arcs show the number of miles of line that would have to be constructed at \$100,000 per mile.



The figure also shows that storage facilities are both already connected to the company's main terminal through existing lines. An estimated 800 million cubic feet must be shipped each year from field 1 to the terminal, and 600 million from field 2. Variable shipping cost is \$2000 per million cubic feet on each link of the network, and all links have an annual capacity of 1 billion cubic feet. The company wants a minimum total annual cost system for the required shipping.

☑ (a) Formulate this problem as a network design ILP.

☑ (b) Use class optimization software to solve you ILP.

11-33 Dandy Diesel manufacturing company assembles diesel engines for heavy construction equipment. Over the next 4 quarters the company expects to ship 40, 20, 60, and 15 units, respectively, but no more than 50 can be assembled in any quarter. There is a fixed cost of \$2000 each time the line is setup for production, plus \$200 per unit assembled. Engines may be held over in inventory at the plant for \$100 per unit per month. Dandy seeks a minimum total cost production plan for the 4 quarters, assuming that there is no beginning or ending inventory.

Do (a) and (b) as in Exercise 11-32. (*Hint:* Create nodes for each quarter and a common source node for production arcs.)

11-34 Top-T shirt company imprints T-shirts with cartoons and celebrity photographs. For each of their 4 pending contracts, the following table shows the number of days of production required, the earliest day the order can begin, and the day the order is due.

	1	2	3	4
Production	10	3	16	8
Earliest	0	20	1	12
Due Date	12	30	20	21

The company wants to design an optimal schedule assuming that contracts can be processed in any sequence but that production cannot be interrupted once a job has started.

- ☑ (a) Ignoring objective functions for the moment, formulate constraints of a single-machine ILP to select an optimal start time for each contract.
- ☑ (b) Evaluate each of the 8 objective function in principle [11.37] for the schedule with start times 2, 20, 23, and 12 for the four contracts, respectively.
- ☑ (c) Extend your constraints of part (a) to formulate an ILP to compute a minimum mean completion time schedule.
- ☑ (d) Use class optimization software to solve your ILP of part (c).
- ☑ (e) Without actually solving, list the other objective functions of [11.37] for which your schedule of part (d) must be optimal.

- ✔ (f) Extend your constraints of part (a) to formulate an ILP to compute a minimum maximum lateness schedule.
- ✔ (g) Use class optimization software to solve your ILP of part (f).
- ✔ (h) Without actually solving, list the other objective functions of 11.37 for which your schedule of part (g) must be optimal.

11-35 Sarah is a graduate student who must make 4 large experimental runs on her personal computer as part of her thesis research. The jobs require virtually all the computer’s resources, so only one can be processed at a time and none can be interrupted once it has begun. The following table shows the number of days of computing each job will require, the earliest that all data will be available, and the day Sarah has promised the result to her thesis advisor.

	1	2	3	4
Time	15	8	20	6
Earliest	0	0	10	10
Promise	20	20	30	20

Before beginning any work, Sarah wants to compute an optimal schedule. Assume promised times are only targets.

Do (a) through (h) as in Exercise 11-34, evaluating the schedule with start times 8, 0, 23, and 43, respectively.

11-36 Three new jobs have just arrived at Fancy Finishing’s main furniture restoration shop. The following table shows the sequence that each must follow through the company’s 3 finish removal processes and the time required for each.

Job	Sequence	Process Time		
		1	2	3
1	1–2–3	10	3	14
2	1–3–2	2	4	1
3	2–1–3	12	6	8

Once a process is begun, it cannot be interrupted. Although the shop was empty when the new jobs arrived, Fancy expects more in the next few days. To maintain efficiency, they seek a schedule that minimizes the average time that a job is in the shop.

- ✔ (a) Ignoring objective functions for the moment, formulate constraints of a job shop ILP to select an optimal start time for each job on each machine.
- ✔ (b) Which of the 8 objective functions in 11.37 is appropriate for this problem?
- ✔ (c) Complete an ILP model by introducing that objective.
- ✔ (d) Use class optimization software to solve your ILP model for an optimal schedule.

11-37 A team of auditors has divided itself into 3 groups, each to examine one category of records. Each group will review their speciality area for all 3 subsidiaries of the client being audited, but the required sequence and times differ, as shown in the following table.

Subsidiary	Sequence	Group Time		
		1	2	3
1	1–3–2	4	5	12
2	2–1–3	6	18	3
3	3–2–1	5	7	3

Once a group starts on a subsidiary, it should finish all work either before it moves to another or before a different group begins on theirs. The team seeks a schedule that will complete all work at the earliest possible time.

Do (a) through (d) as in Exercise 11-36.

11-38 With the addition of a new plant, Monsanto⁹ now has more capacity than it needs to manufacture its main chemical product. Numerous reactors $i = 1, \dots, m$ can be operated at a variety $j = 1, \dots, n$ of discrete combinations of settings for feed rate, reactor velocity, and reactor pressure. Both the production yield $p_{i,j}$ and the operating cost $c_{i,j}$ vary with reactor and setting. Formulate an ILP model to find the least cost way to fulfill total production target b in terms of the decision variables ($i = 1, \dots, m; j = 1, \dots, n$)

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if reactor } i \text{ operates at setting } j \\ 0 & \text{otherwise} \end{cases}$$

11-39 W.R. Grace¹⁰ strip mines phosphates in strata numbered from $i = 1$ at the top to $i = n$ at the deepest level. Each stratum must be removed

⁹Based on R. R. Boykin (1985), “Optimizing Chemical Production at Monsanto,” *Interfaces*, 15:1, 88–95.

¹⁰Based on D. Klingman and N. Phillips (1988), “Integer Programming for Optimal Phosphate Mining Strategies,” *Journal of the Operational Research Society*, 9, 805–809.

before the next can be mined, but only some of the layers contain enough suitable minerals to justify processing into the company's three products: pebble, concentrate, and flotation feed ($j = 1, 2, 3$). The company can estimate from drill samples the quantity $a_{i,j}$ of product j available in each stratum i , the fraction $b_{i,j}$ of BPL (a measure of phosphate content) in the part of i suitable for j , and the corresponding fraction $p_{i,j}$ of pollutant chemicals. They wish to choose a mining plan that maximizes the product output while keeping the average fraction BPL of material processed for each product j at least b_j and the average pollution fraction at most p_j . Formulate an ILP model of this mining problem using the decision variables ($i = 1, \dots, n$)

$$x_i \triangleq \begin{cases} 1 & \text{if stratum } i \text{ is removed} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i \triangleq \begin{cases} 1 & \text{if stratum } i \text{ is processed} \\ 0 & \text{otherwise} \end{cases}$$

11-40 Ault Food Limited¹¹ is planning the production and distribution system for its new line of food products. Plants may be opened at any of sites $i = 1, \dots, 7$, and warehouses at locations $j = 1, \dots, 13$, to meet demands d_k at customer regions $k = 1, \dots, 219$. Each plant costs \$50 million to open and produces up to 30 thousand cases per year. Warehouses cost \$12 million to open and handle up to 10 cases per year. Transportation costs are $r_{i,j}$ per case for rail shipment from plant i to warehouse j , and $t_{j,k}$ per case for trucking from warehouse j to customer k . No direct shipments from the plants are allowed. Formulate ILP model to decide which facilities to open and how to service customers using the decision variables ($i = 1, \dots, 7; j = 1, \dots, 13; k = 1, \dots, 219$)

$x_{i,j,k} \triangleq$ thousand of cases produced at plant i and shipped to customer k via warehouse j

$$y_i \triangleq \begin{cases} 1 & \text{if plant } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

$$w_j \triangleq \begin{cases} 1 & \text{if warehouse } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

11-41 Space structures¹² designed for zero gravity have no structural weight to support and nothing to which a foundation can be attached. The structure needs only to withstand vibrations in space. This is accomplished by replacing truss members with dampers at a given number of p sites among $j = 1, \dots, n$ candidate locations throughout the structure. Engineering analysis can identify the principal strain modes $i = 1, \dots, m$, and estimate fractions $d_{i,j}$ of total modal strain energy imparted in mode i to truss site j . The best design places dampers to absorb as much energy as possible. Specifically we want to maximize the minimum total of $d_{i,j}$ reaching chosen sites over all modes i . Formulate an ILP model to choose an optimal design in terms of p , the $d_{i,j}$, and the decision variables ($j = 1, \dots, n$)

$$x_j \triangleq \begin{cases} 1 & \text{if a damper is placed at } j \\ 0 & \text{otherwise} \end{cases}$$

$$z \triangleq \text{smallest modal } d\text{-total}$$

(Hint: Maximize z .)

11-42 The National Cancer Institute¹³ has received proposals from 22 states to participate in its newest smoking intervention study. The first $j = 1, \dots, 5$ are from the Northeast region, the next $j = 6, \dots, 11$ from the Southeast, numbers $j = 12, \dots, 17$ from the Midwest, and the last $j = 18, \dots, 22$ from the West. At least 3 are to be selected from each region. Each proposal has been evaluated and rated with a merit score r_j based on rankings by a panel of experts. Selected project budgets b_j (in millions of dollars) must total no more than the \$15 million available for the study, and the number of smokers s_j (in millions) living in chosen states must sum to at least 11 million. Proposals $j = 2, 7, 11, 19$ come from states in the highest quartile with respect to the fraction of the population that smokes; proposals $j = 1, 4, 13, 14, 21$ come from the lowest quartile. At least 2 states must be selected from each of these outlier groups. Formulate an ILP model to choose

¹¹Based on J. Pooley (1994), "Integrated Production and Distribution Facility Planning for Ault Foods," *Interfaces*, 24:4, 113–121.

¹²Based on R. K. Kincaid and R. T. Berger (1993), "Damper Placement for the CSI-Phase I Evolutionary Model," 34th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, part 6, 3086–3095.

¹³Based on N. G. Hall, J. C. Hershey, L. G. Kessler, and R. C. Spotts (1992), "A Model for Making Project Funding Decisions at the National Cancer Institute," *Operations Research*, 40, 1040–1052.

a maximum merit feasible set of proposals to fund using the decision variables ($j = 1, \dots, 22$)

$$x_j \triangleq \begin{cases} 1 & \text{if proposal } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

11-43 Every year the city of Montreal¹⁴ must remove large quantities of snow from sectors $i = 1, \dots, 60$ of the city. Each sector is assigned to one of sites $j = 1, \dots, 20$ as its primary disposal point. From prior-year history, planners have been able to estimate the expected volume of snowfall f_i (in cubic meters) in each sector and the capacity u_j (in cubic meters) of each disposal site. They also know the travel distance $d_{i,j}$ from each sector to each disposal site. Removal rates also have to be considered. The total of hourly removal rates r_i (in m^3/hr) associated with sectors assigned to any disposal site must not exceed its receiving rate b_j (in m^3/hr). Formulate an ILP model to select an assignment that minimizes total distance times volume moved using the decision variables ($i = 1, \dots, 60; j = 1, \dots, 20$)

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if sector } i \text{ is assigned to disposal } j \\ 0 & \text{otherwise} \end{cases}$$

11-44 Highway maintenance in Australia¹⁵ is performed by crews operating out of maintenance depots. The Victoria region is planning a major realignments of its depots to provide more effective service to the $i = 1, \dots, 276$ highway segments in the region. A total of 14 sites must be selected from among the possible depot locations $j = 1, \dots, 36$. Review of the possibilities has determined the indicators

$$a_{i,j} \triangleq \begin{cases} 1 & \text{if a depot at } j \text{ is close enough} \\ & \text{to provide good service to segment } i \\ 0 & \text{otherwise} \end{cases}$$

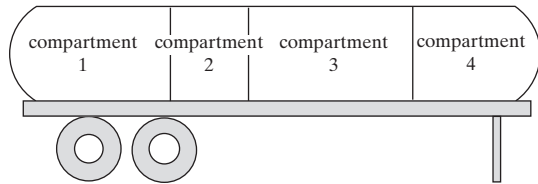
Not all segments will be able to receive this good service, so planners wish to select 14 locations to minimize to sum of service requirements s_i at

segments with inadequate service. Formulate an ILP model to select an optimal collection of depots using the decision variables ($i = 1, \dots, 276; j = 1, \dots, 36$)

$$x_i \triangleq \begin{cases} 1 & \text{if depot } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i \triangleq \begin{cases} 1 & \text{if segment } i \text{ goes inadequately covered} \\ 0 & \text{otherwise} \end{cases}$$

11-45 Mobil Oil Corporation¹⁶ serves 600,000 customers with 430 tanktrucks operating out of 120 bulk terminals. As illustrated in the following figure, tanktrucks have several compartments $c = 1, \dots, n$ of varying capacity u_c .



The final stage of distribution planning is to allocate outgoing gasoline products $p = 1, \dots, m$ to compartments. The various products ordered are placed in one or more compartments, but each compartment can contain only one product. To avoid overfilling, the total gallons of any product loaded can be reduced from the ordered volume v_p by up to b_p gallons. The loading procedure seeks to minimize the sum of these underloadings while meeting all other requirements. Formulate an ILP model of this loading problem using the decision variables ($p = 1, \dots, m; c = 1, \dots, n$)

$$x_{p,c} \triangleq \text{gallons of product } p \text{ loaded in compartment } c$$

$$y_{p,c} \triangleq \begin{cases} 1 & \text{if product } p \text{ is loaded in} \\ & \text{compartment } c \\ 0 & \text{otherwise} \end{cases}$$

$$z_p \triangleq \text{gallons underloading of product } p$$

¹⁴Based on J. F. Campbell and A. Langevin (1995), "The Snow Disposal Assignment Problem," *Journal of the Operational Research Society*, 46, 919–929.

¹⁵Based on G. Rose, D. W. Bennett, and A. T. Evans (1992), "Locating and Sizing Road Maintenance Depots," *European Journal of Operational Research*, 63, 151–163.

¹⁶Based on G. G. Brown, C. J. Ellis, G. W. Graves, and D. Ronen (1987), "Real-Time, Wide Area Dispatch of Mobil Tank Trucks," *Interfaces*, 17:1, 107–120.

11-46 Each of the 11 nurses¹⁷ at Rosey Retirement Home works a total of 10 days within each 2-week period, alternating between a compatible pair of weekly schedules taken from those depicted in the following table (1 = work, 0 = off).

Day	Shift					
	1	2	3	4	5	6
1	1	1	0	1	1	0
2	1	0	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	0	1
5	0	1	1	0	1	1
6	0	0	0	1	1	1
7	0	0	0	1	1	1

Compatible weekly schedule pairs are $C \triangleq \{(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6), (4, 1), (4, 2), (5, 2), (5, 3), (6, 3)\}$, and the number working each shift should be constant from week to week. At least r_d nurses must be on duty on days $d = 1, \dots, 7$, but excesses are allowed if mandated by the shift patterns. Rosey’s objective is to minimize the sum of any such over-staffing. Formulate an ILP model to decide an optimal cyclic schedule for the staff of 11 using the decision variables $((i, j) \in C, d = 1, \dots, 7)$,

$x_{i,j} \triangleq$ number of nurses working shift i then shift j
 $z_d \triangleq$ excess number of nurses scheduled on day d

11-47 Regional Bell telephone operating companies,¹⁸ which buy many products from a much smaller number of suppliers, often solicit discounts based on the total dollar volume of business awarded to particular suppliers. For example, suppliers $i = 1, \dots, 25$ might be required to quote base prices $p_{i,j}$ for needed products $j = 1, \dots, 200$, together with upper limits $b_{i,k}, k = 1, \dots, 5$ for ranges on total dollar volume and corresponding discount fractions $d_{i,k}$ that increase with k . Then the actual cost to the telephone company of supplier i ’s goods will be $(1 - d_{i,1})$ times the total

base price of those goods if that total dollar value falls within the interval $[u_{i,0}, u_{i,1}]$, $(1 - d_{i,2})$ if the total dollar volume falls within $[u_{i,1}, u_{i,2}]$, and so on (assume that $u_{i,0} \triangleq 0, u_{i,5} \geq$ any feasible dollar volume). The company wants to choose bids to equal or exceed all required product quantities r_j at least total discounted cost. Formulate an ILP model of this volume discount problem using the decision variables ($i = 1, \dots, 25; j = 1, \dots, 200; k = 1, \dots, 5$)

$x_{i,j} \triangleq$ quantity of product j purchased from supplier i

$w_{i,k} \triangleq$ dollar volume of goods from supplier i when discount range k applies

$y_{i,k} \triangleq \begin{cases} 1 & \text{if discount range } k \text{ applies for} \\ & \text{supplier } i \\ 0 & \text{otherwise} \end{cases}$

11-48 The small Adele¹⁹ textile company knits products $p = 1, \dots, 79$ on a variety of machines $m = 1, \dots, 48$ to meet known output quotas q_p pounds for the next week. The variable cost per pound of making product p on machine m is known $c_{m,p}$. Machines operate with changeable cylinder types $j = 1, \dots, 14$, which have different combinations of knitting needles, and thus yield different quantities $a_{m,j,p}$ (pounds) of product p per hour on machine m . A total of 100 hours is available on each machine over the next week, but a setup time $s_{m,j}$ must be deducted for each cylinder type j used on each machine m . Adele wants to find a minimum total variable cost schedule that conforms to all constraints. Formulate an ILP model of this production scheduling problem using decision variables ($m = 1, \dots, 48; j = 1, \dots, 14; p = 1, \dots, 79$)

$x_{m,j,p} \triangleq$ pounds of product p made on machine m with cylinder type j

$y_{m,j} \triangleq \begin{cases} 1 & \text{if cylinder type } j \text{ is used on} \\ & \text{machine } m \\ 0 & \text{otherwise} \end{cases}$

¹⁷Based on E. S. Rosenbloom and N. F. Goertzen (1987), “Cyclic Nurse Scheduling,” *European Journal of Operational Research*, 31, 19–23.

¹⁸Based on P. Katz, A. Sadrian, and P. Tendick (1994), “Telephone Companies Analyze Price Quotations with Bellcore’s PDSS Software,” *Interfaces*, 24:1, 50–63.

¹⁹Based on U. Akinc (1993), “A Practical Approach to Lot and Setup Scheduling at a Textile Firm,” *IIE Transactions*, 25, 54–64.

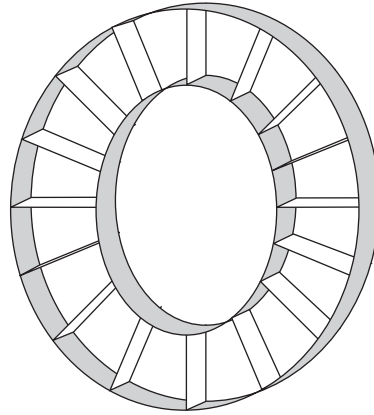
11-49 Mail Order Mart (MOM)²⁰ will ship q_j pounds of small-order novelty goods over the next week to regions $j = 1, \dots, 27$ of the United States. MOM's distribution facility is located in New England region 1. Orders can be directly shipped from the distribution center by small parcel carriers at cost $p_{1,j}$ per pound. An often cheaper alternative is to drop-ship (i.e., group) the week's orders for a region j into a bulk quantity that can be sent by common carrier freight to an intermediate point in region i for c_i per pound ($c_1 \triangleq 0$), and then be directly shipped from i on to j at small-parcel cost $p_{i,j}$ per pound. However, common carriers require a minimum of 1000 pounds per shipment. MOM wants to identify the least total cost way to meet this week's shipping needs. Formulate an ILP model of this shipping problem using the decision variables ($i, j = 1, \dots, 27$)

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if goods bound for region } j \text{ are drop} \\ & \text{are drop shipped via region } i \\ 0 & \text{otherwise} \end{cases}$$

($x_{1,j} = 1$ implies direct shipping to j .)

11-50 Gas turbine engines²¹ have the following radial assembly of nozzle guides located immediately upstream from each rotor. The purpose of the vanes is to spread flow uniformly over the rotor, which improves its efficiency materially. During engine maintenance, the 55 old vanes are removed and replaced by new and refurbished ones $i = 1, \dots, 55$ marked with previously assessed performance measures a_i and b_i of the two faces. The effect of each nozzle slot is greatly impacted by the sum of the a -value for the vane placed on one side and the b -value for the vane installed on the other. Maintenance personnel want to choose an (counterclockwise) arrangement of vanes around the assembly to balance this performance by minimizing the total of squared deviations between each resulting $a + b$ sum and known target value t . Show that this vane arrangement task can be

viewed as a traveling salesman problem and write an expression for the corresponding point i to point j costs.



11-51 A new freight airline²² is designing a hub-and-spoke system for its operations. From a total of 34 airports to be served, 3 will be selected as hubs. Then (one-way) airport-to-airport freight quantities $f_{i,j}$ will be routed via the hubs ($f_{i,i} = 0$ for all i). That is, flow from i to j will begin at i , go to the unique hub k for i , then pass to the (possibly same) hub ℓ for j before being shipped on to j . The goal is to minimize the total of flow time unit transportation costs $c_{i,j}$ taking into account a 30% savings for flows between hubs that results from economies of scale ($c_{i,i} = 0$ for all i).

- (a) Explain why appropriate decision variables for an integer programming model of this hub design problem are ($i, k = 1, \dots, 34$)

$$x_{i,k} \triangleq \begin{cases} 1 & \text{if airport } i \text{ is assigned to a hub at } k \\ 0 & \text{otherwise} \end{cases}$$

$$y_k \triangleq \begin{cases} 1 & \text{if a hub opens at } k \\ 0 & \text{otherwise} \end{cases}$$

- (b) Use only the $x_{i,k}$ (i.e., disregard whether hubs are open) to formulate a quadratic objective function summing origin-to-hub,

²⁰Based on L. S. Franz and J. Woodmansee (1993), "Zone Skipping vs. Direct Shipment of Small Orders: Integrating Order Processing and Optimization," *Computers and Operations Research*, 20, 467–475.

²¹Based on R. D. Plante, T. J. Lowe, and R. Chandrasekaran (1987), "The Product Matrix Traveling Salesman Problem: An Application and Solution Heuristic," *Operations Research*, 35, 772–783.

²²Based on M. E. O'Kelly (1987), "A Quadratic Integer Program for the Location of Interacting Hub Facilities," *European Journal of Operational Research*, 32, 393–404.

hub-to-hub, and hub-to-destination transportation costs for each pair of nodes.

- (c) Complete an INLP model of the problem by adding linear main constraints and appropriate variable-type constraints.

11-52 In an area with many suburban communities, telephone listings are usually grouped into several different books.²³ Patrons in each community $i = 1, \dots, n$ are covered in exactly one directory $k = 1, \dots, m$. Numbers of patrons p_i are known for each community, as well

as (one-way) calling traffic levels $t_{i,j}$ between communities. Engineers seek to design books that maximize the traffic among patrons sharing a common telephone book without listing more than capacity q patrons in any single directory. Formulate an INLP model to select an optimal collection of telephone books in terms of the decision variables ($i = 1, \dots, n, k = 1, \dots, m$)

$$x_{i,k} \triangleq \begin{cases} 1 & \text{if community } i \text{ goes in book } k \\ 0 & \text{otherwise} \end{cases}$$

REFERENCES

Chen, Der-San, Robert G. Batson, and Yu Dang (2010), *Applied Integer Programming - Modeling and Solution*. Wiley, Hoboken, New Jersey.

Hillier, Fredrick S. and Gerald J. Lieberman (2001), *Introduction to Operations Research*. McGraw-Hill, Boston, Massachusetts.

Parker, R. Gary and Ronald L. Rardin (1988), *Discrete Optimization*. Academic Press, San Diego, California.

Taha, Hamdy (2011), *Operations Research - An Introduction*. Prentice-Hall, Upper Saddle River, New Jersey.

Winston, Wayne L. (2003), *Operations Research - Applications and Algorithms*. Duxbury Press, Belmont, California.

Wolsey, Laurence (1998), *Integer Programming*. John Wiley, New York, New York.

²³Based on S. Chen and C. J. McCallum (1977), "The Application of Management Science to the Design of Telephone Directories," *Interfaces*, 8:1, 58–69.

This page intentionally left blank

Exact Discrete Optimization Methods

In Chapter 11 we illustrated the wide range of integer and combinatorial optimization models encountered in operations research practice. Some are linear programs with a few discrete side constraints; others are still linear but involve only combinatorial decision variables; still others are both nonlinear and combinatorial. Each one includes logical decisions that just cannot be modeled validly as continuous, so most lack the elegant tractability of the LP and network models studied in earlier chapters.

Diminished tractability does not imply diminished importance. Discrete optimization models such as those presented in Chapter 11 all represent critical decision problems in engineering and management that must somehow be confronted. Even partial analysis can prove enormously valuable.

It should not surprise that discrete optimization methods span a range as wide as the models they address. In contrast to, say, linear programming, where a few prominent algorithms have proved adequate for the overwhelming majority of models, success in discrete optimization often requires methods cleverly specialized to an individual application. Still, there are common themes. In this chapter we introduce the best known of those seeking—at least nominally—**exact optimal** solutions. Chapter 15 treats expressly **heuristic methods** satisfied with approximate optima.

12.1 SOLVING BY TOTAL ENUMERATION

Beginning students often find counterintuitive the idea that discrete optimization problems are more difficult than their continuous analogs. The algebra of LP algorithms in Chapters 5 and 6 is rather daunting. By comparison, a discrete model, which has only a finite number of choices for decision variables, can seem refreshingly easy. Why not just try them all and keep the best feasible solution as optimal?

Although naive, this point of view contains a kernel of wisdom.

Principle 12.1 If a model has only a few discrete decision variables, the most effective method of analysis is often the most direct: enumeration of all the possibilities.

Total Enumeration

To be more specific, total or complete enumeration requires checking all possibilities implied by discrete variable values.

Definition 12.2 **Total enumeration** solves a discrete optimization by trying all possible combinations of discrete variable values, computing for each the best corresponding choice of any continuous variables. Among combinations yielding a feasible solution, those with the best objective function value are optimal.

Swedish Steel All-or-Nothing Application

We can illustrate with the discrete version of our Swedish Steel application formulated in model (11.2) (Section 11.1):

$$\begin{aligned}
 \min \quad & 16(75)y_1 + 10(250)y_2 + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 \\
 \text{s.t.} \quad & 75y_1 + 250y_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 \\
 & 0.0080(75)y_1 + 0.0070(250)y_2 + 0.0085x_3 + 0.0040x_4 \geq 0.0065(1000) \\
 & 0.0080(75)y_1 + 0.0070(250)y_2 + 0.0085x_3 + 0.0040x_4 \leq 0.0075(1000) \\
 & 0.180(75)y_1 + 0.032(250)y_2 + 1.0x_5 \geq 0.030(1000) \\
 & 0.180(75)y_1 + 0.032(250)y_2 + 1.0x_5 \leq 0.035(1000) \\
 & 0.120(75)y_1 + 0.011(250)y_2 + 1.0x_6 \geq 0.010(1000) \\
 & 0.120(75)y_1 + 0.011(250)y_2 + 1.0x_6 \leq 0.012(1000) \\
 & 0.001(250)y_2 + 1.0x_7 \geq 0.011(1000) \\
 & 0.001(250)y_2 + 1.0x_7 \leq 0.013(1000) \\
 & x_3, \dots, x_7 \geq 0 \\
 & y_1, y_2 = 0 \text{ or } 1
 \end{aligned} \tag{12.1}$$

In this version the first two sources of scrap iron have to be entered on an all-or-nothing basis modeled with discrete variables. The other five sources can be employed in any nonnegative amount.

There are 2 possible values for y_1 and 2 for y_2 , or a total of $2 \cdot 2 = 4$ combinations to enumerate. Table 12.1 provides details. Third option $y_1 = 1, y_2 = 0$ yields the optimal solution with objective value 9540.3.

TABLE 12.1 Enumeration of the Swedish Steel All-or-Nothing Model

Discrete Combination	Corresponding Continuous Solution						Objective Value
$y_1 = 0, y_2 = 0$	$x_3 = 814.3,$	$x_4 = 114.6,$	$x_5 = 30.0,$	$x_6 = 10.0,$	$x_7 = 1.1$		9914.1
$y_1 = 0, y_2 = 1$	$x_3 = 637.9,$	$x_4 = 82.0,$	$x_5 = 22.0,$	$x_6 = 7.3,$	$x_7 = 0.9$		9877.3
$y_1 = 1, y_2 = 0$	$x_3 = 727.6,$	$x_4 = 178.8,$	$x_5 = 16.5,$	$x_6 = 1.0,$	$x_7 = 1.1$		9540.3
$y_1 = 1, y_2 = 1$	$x_3 = 552.8,$	$x_4 = 112.9,$	$x_5 = 8.5,$	$x_6 = 0.0,$	$x_7 = 0.9$		9591.1

Since this model has both discrete and continuous variables, each case enumerated requires solving a continuous optimization over variables x_3, \dots, x_7 to find the best continuous values to go with the choice of discrete variables selected. For example, fixing $y_1 = y_2 = 0$ in model (12.1) leaves the linear program

$$\begin{aligned}
 \min \quad & 16(75)(0) + 10(250)(0) + 8x_3 + 9x_4 + 48x_5 + 60x_6 + 53x_7 \\
 \text{s.t.} \quad & 75(0) + 250(0) + x_3 + x_4 + x_5 + x_6 + x_7 = 1000 \\
 & 0.0080(75)(0) + 0.0070(250)(0) + 0.0085x_3 + 0.0040x_4 \geq 0.0065(1000) \\
 & 0.0080(75)(0) + 0.0070(250)(0) + 0.0085x_3 + 0.0040x_4 \leq 0.0075(1000) \\
 & 0.180(75)(0) + 0.032(250)(0) + 1.0x_5 \geq 0.030(1000) \\
 & 0.180(75)(0) + 0.032(250)(0) + 1.0x_5 \leq 0.035(1000) \\
 & 0.120(75)(0) + 0.011(250)(0) + 1.0x_6 \geq 0.010(1000) \\
 & 0.120(75)(0) + 0.011(250)(0) + 1.0x_6 \leq 0.012(1000) \\
 & 0.001(250)(0) + 1.0x_7 \geq 0.011(1000) \\
 & 0.001(250)(0) + 1.0x_7 \leq 0.013(1000) \\
 & x_3, \dots, x_7 \geq 0
 \end{aligned}$$

Optimal solution $x_3 = 814.3$, $x_4 = 144.6$, $x_5 = 30.0$, $x_6 = 10.0$, $x_7 = 1.1$, completes the first case in Table 12.1.

EXAMPLE 12.1: SOLVING BY TOTAL ENUMERATION

Solve the following discrete optimization model by total enumeration [\[12.2\]](#).

$$\begin{aligned}
 \max \quad & 7x_1 + 4x_2 + 19x_3 \\
 \text{s.t.} \quad & x_1 + x_3 \leq 1 \\
 & x_2 + x_3 \leq 1 \\
 & x_1, x_2, x_3 = 0 \text{ or } 1
 \end{aligned}$$

Solution: Checking the $2^3 = 8$ combinations produces the following table:

Case	Objective	Case	Objective
$\mathbf{x} = (0, 0, 0)$	0	$\mathbf{x} = (1, 0, 0)$	7
$\mathbf{x} = (0, 0, 1)$	19	$\mathbf{x} = (1, 0, 1)$	Infeasible
$\mathbf{x} = (0, 1, 0)$	4	$\mathbf{x} = (1, 1, 0)$	11
$\mathbf{x} = (0, 1, 1)$	Infeasible	$\mathbf{x} = (1, 1, 1)$	Infeasible

Solution $\mathbf{x} = (0, 0, 1)$ is the feasible one with best objective value 19, so it is optimal.

Exponential Growth of Cases to Enumerate

Our Swedish Steel application has two discrete decision variables, each with two possible values 0 and 1. A total of

$$2 \cdot 2 = 2^2 = 4$$

combinations result.

Similar thinking shows that a model with k binary decision variables would have

$$\underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{k \text{ times}} = 2^k$$

cases to enumerate. This is **exponential growth**, with every additional 0–1 variable doubling the number of combinations. Even $k = 100$ implies $\approx 10^{10}$.

Principle 12.3 Exponential growth makes total enumeration impractical for models having more than a few discrete variables.

12.2 RELAXATIONS OF DISCRETE OPTIMIZATION MODELS AND THEIR USES

Because analysis of discrete optimization models is usually hard, it is natural to look for related but easier formulations that can aid in the analysis. **Relaxations** are auxiliary optimization models of this sort formed by weakening either the constraints or the objective function or both of the given discrete model.

APPLICATION 12.1: BISON BOOSTERS

Before considering relaxation in the more realistic circumstances of models in Chapter 11, it will help to develop a more compact (albeit highly artificial) example. Consider the dilemma of the Bison Boosters club supporting the local athletic team.

The Boosters are trying to decide what fundraising projects to undertake at the next country fair. One option is customized T-shirts, which will sell for \$20 each; the other is sweatshirts selling for \$30. History shows that everything offered for sale will be sold before the fair is over.

Materials to make the shirts are all donated by local merchants, but the Boosters must rent the equipment for customization. Different processes are involved, with the T-shirt equipment renting at \$550 for the period up to the fair, and the sweatshirt equipment for \$720. Display space presents another consideration. The Boosters have only 300 square feet of display wall area at the fair, and T-shirts will consume 1.5 square feet each, sweatshirts 4 square feet each. What plan will net the most income?

Certainly this problem centers on making shirts, so decision variables will include

$$\begin{aligned} x_1 &\triangleq \text{number of T-shirts made and sold} \\ x_2 &\triangleq \text{number of sweatshirts made and sold} \end{aligned}$$

However, the Boosters also confront discrete decisions on whether to rent equipment:

$$\begin{aligned} y_1 &\triangleq 1 \text{ if T-shirt equipment is rented and } = 0 \text{ otherwise} \\ y_2 &\triangleq 1 \text{ if sweatshirt equipment is rented and } = 0 \text{ otherwise} \end{aligned}$$

Using these decision variables, the Boosters' dilemma can be modeled:

$$\begin{aligned} \max \quad & 20x_1 + 30x_2 - 550y_1 - 720y_2 \quad (\text{net income}) \\ \text{s.t.} \quad & 1.5x_1 + 4x_2 \leq 300 \quad (\text{display space}) \\ & x_1 \leq 200y_1 \quad (\text{T-shirts if equipment}) \end{aligned} \tag{12.2}$$

$$\begin{aligned}
 x_2 &\leq 75y_2 && \text{(sweatshirts if equipment)} \\
 x_1, x_2 &\geq 0 \\
 y_1, y_2 &= 0 \text{ or } 1
 \end{aligned}$$

The objective function maximizes net income, and the first main constraint enforces the display space limit. The next two constraints provide the switching we have seen in other models. Any sufficiently large big- M could be used as the y_j coefficient in these constraints. Values in (12.2) derive from the greatest production possible within the 300 square feet display limit. Coefficients $300/1.5 = 200$ for T-shirts and $300/4 = 75$ for sweatshirts introduce no limitation if y 's equal 1, yet switch off all production if y 's equal 0.

Enumeration of the 4 combinations of y_1 and y_2 values easily establishes that the Boosters should make only T-shirts. The unique optimal solution is $x_1^* = 200$, $x_2^* = 0$, $y_1^* = 1$, $y_2^* = 0$, with net income \$3450.

Constraint Relaxations

Relaxations may weaken either the objective function or the constraints, but the elementary ones we explore in this book mostly focus on constraints. A constraint relaxation produces an easier model by dropping or easing some constraints.

Definition 12.4 Model (\bar{P}) is a **constraint relaxation** of model (P) if every feasible solution to (P) is also feasible in (\bar{P}) and both models have the same objective function.

New feasible solutions may be allowed, but none should be lost.

Table 12.2 shows several constraint relaxations of the tiny Bison Boosters model (12.2). The first simply doubles capacities. The result is certainly a relaxation, because every solution fitting within the true capacity of 300 square feet will also fit within twice as much area. Still, this relaxation gains us little.

TABLE 12.2 Constraint Relaxations of Bison Boosters Model

Revised Constraints	Discussion
$1.5x_1 + 4x_2 \leq 600$ $x_1 \leq 400y_1$ $x_2 \leq 150y_2$ $x_1, x_2 \geq 0$ $y_1, y_2 = 0 \text{ or } 1$	Doubled capacities. Relaxation optimum: $\bar{x}_1 = 400, \bar{x}_2 = 0, \bar{y}_1 = 1, \bar{y}_2 = 0$, net income \$7450
$x_1 \leq 200y_1$ $x_2 \leq 75y_2$ $x_1, x_2 \geq 0$ $y_1, y_2 = 0 \text{ or } 1$	Dropped first constraint. Relaxation optimum: $\bar{x}_1 = 200, \bar{x}_2 = 75, \bar{y}_1 = 1, \bar{y}_2 = 1$, net income \$4980
$1.5x_1 + 4x_2 \leq 300$ $x_1 \leq 200y_1$ $x_2 \leq 75y_2$ $x_1, x_2 \geq 0$ $0 \leq y_1 \leq 1$ $0 \leq y_2 \leq 1$	Linear programming relaxation with discrete variables treated as continuous. Relaxation optimum: $\bar{x}_1 = 200, \bar{x}_2 = 0, \bar{y}_1 = 1, \bar{y}_2 = 0$, net income \$3450

Principle 12.5 Relaxations should be significantly more tractable than the models they relax, so that deeper analysis is practical.

Doubling capacities fails this requirement because the character of the model is unchanged.

The second relaxation of Table 12.2 is more on track. Dropping the first constraint delinks decisions about the two types of shirts. It then becomes much easier to compute a (relaxation) optimal solution. We need only decide one by one whether the maximum production now allowed each x_j when its $y_j = 1$ justifies the fixed cost of equipment rental. Both do.

EXAMPLE 12.2: RECOGNIZING CONSTRAINT RELAXATIONS

Determine whether or not each of the following mixed-integer programs is a constraint relaxation of

$$\begin{array}{ll} \min & 3x_1 + 6x_2 + 7x_3 + x_4 \\ \text{s.t.} & 2x_1 + x_2 + x_3 + 10x_4 \geq 100 \\ & x_1 + x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \\ & x_4 \geq 0 \end{array}$$

- | | |
|---|---|
| <p>(a) $\min \quad 3x_1 + 6x_2 + 7x_3 + x_4$
 s.t. $2x_1 + x_2 + x_3 + 10x_4 \geq 100$
 $x_1, x_2, x_3 = 0 \text{ or } 1$
 $x_4 \geq 0$</p> | <p>(b) $\min \quad 3x_1 + 6x_2 + 7x_3 + x_4$
 s.t. $2x_1 + x_2 + x_3 + 10x_4 \geq 200$
 $x_1 + x_2 + x_3 \leq 1$
 $x_1, x_2, x_3 = 0 \text{ or } 1$
 $x_4 \geq 0$</p> |
| <p>(c) $\min \quad 3x_1 + 6x_2 + 7x_3 + x_4$
 s.t. $2x_1 + x_2 + x_3 + 10x_4 \geq 100$
 $x_1 + x_2 + x_3 \leq 1$
 $x_1, x_2, x_3, x_4 \geq 0$</p> | <p>(d) $\min \quad 3x_1 + 6x_2 + 7x_3 + x_4$
 s.t. $2x_1 + x_2 + x_3 + 10x_4 \geq 100$
 $x_1 + x_2 + x_3 \leq 1$
 $1 \geq x_1 \geq 0, 1 \geq x_2 \geq 0,$
 $1 \geq x_3 \geq 0, x_4 \geq 0$</p> |

Solution: We apply definition [12.4](#).

(a) This model is a constraint relaxation because it is formed by dropping the second main constraint. Certainly, every solution feasible in the original model remains so with fewer constraints.

(b) This model is not a relaxation. The only change, which is increasing the righthand side by 100, to 200, eliminates previously feasible solutions. One example is $\mathbf{x} = (0, 0, 0, 10)$.

(c) This model is a relaxation. Allowing x_1 , x_2 , and x_3 to take on any nonnegative value—rather than just 0 or 1—cannot eliminate previously feasible solutions.

(d) This model is also a relaxation. Allowing x_1 , x_2 , and x_3 to take on any values in the interval $[0, 1]$ precludes none of their truly feasible values.

Linear Programming Relaxations

The third case in Table 12.2 illustrates the best known and most used of all constraint relaxation forms: linear programming, or more generally, continuous relaxations.

Definition 12.6 | **Continuous relaxations (linear programming relaxations** if the given model is an integer linear program) are constraint relaxations formed by treating any discrete variables as continuous while retaining all other constraints.

In the real Bison Boosters model, each y_j must equal 0 or 1. In the continuous relaxation we also admit fractions, replacing each

$$y_j = 0 \text{ or } 1 \quad \text{by} \quad 1 \geq y_j \geq 0$$

Certainly, no feasible solutions are lost by allowing both fractional and integer choices for discrete variables, so the process does produce a valid relaxation. More important, the relaxed model usually proves significantly more tractable.

Our Bison Boosters model is an integer linear program (ILP), linear in all aspects except the discreteness of y_1 and y_2 . Thus, relaxing discrete variables to continuous leaves a linear program to solve—the linear programming relaxation; we have already expended several chapters of this book showing how effectively linear programs can be analyzed.

Principle 12.7 | LP relaxations of integer linear programs are by far the most used relaxation forms because they bring all the power of linear programming to bear on analysis of the given discrete models.

EXAMPLE 12.3: FORMING LINEAR PROGRAMMING RELAXATIONS

Form the linear programming relaxation of the following mixed-integer program:

$$\begin{aligned} \min \quad & 15x_1 + 2x_2 - 4x_3 + 10x_4 \\ \text{s.t.} \quad & x_3 - x_4 \leq 0 \\ & x_1 + 2x_2 + 4x_3 + 8x_4 = 20 \\ & x_2 + x_4 \leq 1 \\ & x_1 \geq 0 \\ & x_2, x_3, x_4 = 0 \text{ or } 1 \end{aligned}$$

Solution: Following definition [12.6](#), we replace 0–1 constraints $x_j = 0$ or 1 by $x_j \in [0, 1]$ to obtain the LP relaxation

$$\begin{aligned} \min \quad & 15x_1 + 2x_2 - 4x_3 + 10x_4 \\ \text{s.t.} \quad & x_3 - x_4 \leq 0 \\ & x_1 + 2x_2 + 4x_3 + 8x_4 = 20 \\ & x_2 + x_4 \leq 1 \\ & x_1 \geq 0 \\ & 1 \geq x_2 \geq 0, 1 \geq x_3 \geq 0, 1 \geq x_4 \geq 0 \end{aligned}$$

Relaxations Modifying Objective Functions

Not all relaxations are constraint relaxations. Sometimes the objective function is also weakened. A full definition includes both possibilities.

Definition 12.8 Optimization problem (R) is a **relaxation** of optimization problem (P) if (i) every feasible solution in (P) is feasible in (R), and (ii) the objective value in (R) of every feasible solution in (P) is equal to or better than its objective value in (P) (\leq for a min problem, \geq for a max problem).

EXAMPLE 12.4: RELAXATIONS WITH CHANGING OBJECTIVES

Return to the ILP of Example 12.3, and determine whether each of the following changes in the model produces a valid relaxation.

(a) Replacing the given objective function by

$$\min \quad 3x_1 - 6x_2 + 2x_3 + x_4$$

(b) Dropping the second main constraint as in part (a) of Example 12.3 and replacing the given objective function by

$$\min \quad 3x_1 + 6x_2 + 19x_3 + 5x_4$$

(c) Dropping the second main constraint as in part (a) of Example 12.3 and replacing the given objective function by

$$\min \quad 3x_1 + 6x_2 + 7x_3 + x_4 - 5(1 - x_1 - x_2 - x_3)$$

Solution:

(a) This change has not modified the original constraints, but it has reduced two objective function coefficients. Since all the variables of the problem are nonnegative, reducing their objective coefficients can only produce a lower objective value. This is consistent with [12.8] (ii) for a min problem, and the new model is a valid relaxation.

(b) This change has widened the feasible set consistent with [12.8] (i) by dropping a constraint, but it has increased two objective function coefficients as well. Over nonnegative variables this may produce a violation of [12.8] (ii) for a min problem. The new model is not a valid relaxation.

(c) This change has again widened the feasible set consistent with [12.8] (i) by dropping a constraint. Furthermore, the change in the objective function has subtracted a multiple of the slack in the relaxed constraint. Feasible solutions in the original model will have nonnegative slack in the constraint, and thus the effect is to lower-bound the true objective value which satisfies [12.8] (ii) for a min problem. The model is a valid relaxation.

Proving Infeasibility with Relaxations

Exactly what do relaxations add to our analysis of discrete optimization models? One thing is to prove infeasibility.

Suppose that a relaxation comes out infeasible. Then it has no solutions at all. Since every solution to the full model must also be feasible in the relaxation, it follows that the original model was also infeasible. By analyzing the relaxation we have learned a critical fact about the model of real interest.

Principle 12.9 If a relaxation is infeasible, so is the full model it relaxes.

EXAMPLE 12.5: PROVING INFEASIBILITY WITH RELAXATIONS

Use linear programming relaxation to establish that the following discrete optimization model is infeasible:

$$\begin{array}{ll} \min & 8x_1 + 2x_2 \\ \text{s.t.} & x_1 - x_2 \geq 2 \\ & -x_1 + x_2 \geq -1 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{array}$$

Solution: The linear programming relaxation of this model is

$$\begin{array}{ll} \min & 8x_1 + 2x_2 \\ \text{s.t.} & x_1 - x_2 \geq 2 \\ & -x_1 + x_2 \geq -1 \\ & x_1, x_2 \geq 0 \end{array}$$

It is clearly infeasible, because the two main constraints can be written

$$\begin{array}{l} x_1 - x_2 \geq 2 \\ x_1 - x_2 \leq 1 \end{array}$$

Thus by principle [12.9](#), the given integer program is also infeasible. Any solutions satisfying all constraints would also have to be feasible in the relaxation.

It is important to understand that principle [12.9](#) works in only one direction. An infeasible relaxation does prove the full model also has no solutions. But a feasible relaxation does not assure full model feasibility. Consider, for example, the following tiny ILP

$$\begin{array}{ll} \min & 5x \\ \text{s.t.} & \frac{1}{4} \leq x \leq \frac{1}{2} \\ & x \text{ integer} \end{array}$$

Its LP-relaxation clearly has solutions in the interval $[\frac{1}{4}, \frac{1}{2}]$, but none of them are integer. The full ILP is infeasible.

Solution Value Bounds from Relaxations

Figure 12.1 illustrates how relaxations also give us bounds on optimal solution values. Constraint relaxations expand the feasible set, allowing more candidates

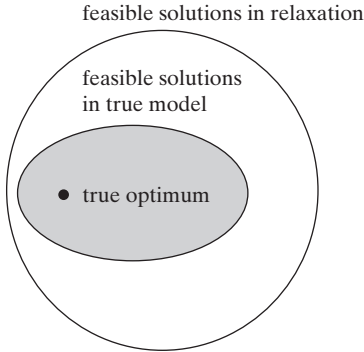


FIGURE 12.1 Relaxations and Optimality

for relaxation optimum. The relaxation optimal value, which is the best over the expanded set of solutions, must then equal or improve on the best feasible solution value to the true model.

Principle 12.10 | The optimal value of any relaxation of a maximize model yields an upper bound on the optimal value of the full model. The optimal value of any relaxation of a minimize model yields a lower bound.

All three constraint relaxations in Table 12.2 illustrate the maximize case. The optimal solution value of the Bison Boosters model (12.2) is \$3450. One of the cases in Table 12.2 yields exactly this value. The others produce higher estimates of net income. All provide the upper bound guaranteed in principle 12.10.

For relaxations with changed objective functions, the issue is a bit more complex, but solution value bounds are still produced. Take, for example, a minimizing problem (P) and valid relaxation (R). By 12.8 (i), every feasible solution to (P) is also feasible in (R), and the objective value for each in (P) is lower-bounded by its objective value in (R) (12.8 (ii)). In particular, the value of an optimal solution in (P) is \geq its value in (R) which is in turn \geq the value of the optimal solution in (R), assuring that relaxation optimal value provides a valid bound on the optimum in (P) as required in principle 12.10.

With the Bison Boosters model, which is so small that it is easily solved optimally, relaxation bounds offer little new insight. A better sense of their value comes from considering the somewhat larger EMS model (11.8) of Section 11.3 that minimizes the number of stations to cover the 20 metropolitan districts:

$$\begin{array}{ll}
 \min & \sum_{j=1}^{10} x_j \qquad \qquad \qquad \text{(number of sites)} \\
 \text{s.t.} & x_2 \qquad \qquad \qquad \geq 1 \quad \text{(district 1)} \\
 & x_1 + x_2 \qquad \qquad \geq 1 \quad \text{(district 2)} \\
 & x_1 + x_3 \qquad \qquad \geq 1 \quad \text{(district 3)} \\
 & x_3 \qquad \qquad \qquad \geq 1 \quad \text{(district 4)}
 \end{array}$$

$$\begin{aligned}
x_3 &\geq 1 && \text{(district 5)} \\
x_2 &\geq 1 && \text{(district 6)} \\
x_2 + x_4 &\geq 1 && \text{(district 7)} \\
x_3 + x_4 &\geq 1 && \text{(district 8)} \\
x_8 &\geq 1 && \text{(district 9)} \\
x_4 + x_6 &\geq 1 && \text{(district 10)} \\
x_4 + x_5 &\geq 1 && \text{(district 11)} \\
x_4 + x_5 + x_6 &\geq 1 && \text{(district 12)} \\
x_4 + x_5 + x_7 &\geq 1 && \text{(district 13)} \\
x_8 + x_9 &\geq 1 && \text{(district 14)} \\
x_6 + x_9 &\geq 1 && \text{(district 15)} \\
x_5 + x_6 &\geq 1 && \text{(district 16)} \\
x_5 + x_7 + x_{10} &\geq 1 && \text{(district 17)} \\
x_8 + x_9 &\geq 1 && \text{(district 18)} \\
x_9 + x_{10} &\geq 1 && \text{(district 19)} \\
x_{10} &\geq 1 && \text{(district 20)} \\
x_1, \dots, x_{10} &= 0 \text{ or } 1
\end{aligned} \tag{12.3}$$

How many stations does this model imply? Even with just 10 discrete variables, the answer is hardly obvious. But if we replace each $x_j = 0$ or 1 constraint by $0 \leq x_j \leq 1$, the resulting linear programming relaxation can be solved quickly with say, the simplex algorithm. An optimal solution is

$$\begin{aligned}
\tilde{x}_1 &= \tilde{x}_7 = 0 \\
\tilde{x}_2 &= \tilde{x}_3 = \tilde{x}_8 = \tilde{x}_{10} = 1 \\
\tilde{x}_4 &= \tilde{x}_5 = \tilde{x}_6 = \tilde{x}_9 = \frac{1}{2}
\end{aligned} \tag{12.4}$$

with optimal value 6.0. Without looking any further into the discrete model, we can conclude that at least 6 EMS sites will be required because this LP relaxation value provides a lower bound (principle [12.10](#)).

EXAMPLE 12.6: COMPUTING BOUNDS FROM RELAXATIONS

Compute (by inspection) the optimal solution value and the LP relaxation bound for each of the following integer programs.

- (a) $\max \quad x_1 + x_2 + x_3$
s.t. $x_1 + x_2 \leq 1$
 $x_1 + x_3 \leq 1$
 $x_2 + x_3 \leq 1$
 $x_1, x_2, x_3 = 0$ or 1
- (b) $\min \quad 20x_1 + 9x_2 + 7x_3$
s.t. $10x_1 + 4x_2 + 3x_3 \geq 7$
 $x_1, x_2, x_3 = 0$ or 1

Solution:

(a) Clearly, only one of the variables in this model can = 1, so the optimal solution value is 1. Corresponding linear programming relaxation

$$\begin{aligned} \max \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & 1 \geq x_1, x_2, x_3 \geq 0 \end{aligned}$$

yields optimal solution $\tilde{\mathbf{x}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ with objective value $\frac{3}{2}$. In accord with principle [12.10], relaxation value $\frac{3}{2}$ is an upper bound on the true optimal value 1 of this maximize model.

(b) Total enumeration shows that an optimal solution to this minimizing ILP is $\mathbf{x} = (0, 1, 1)$ with value 16. Its linear programming relaxation is

$$\begin{aligned} \min \quad & 20x_1 + 9x_2 + 7x_3 \\ \text{s.t.} \quad & 10x_1 + 4x_2 + 3x_3 \geq 7 \\ & 1 \geq x_1, x_2, x_3 \geq 0 \end{aligned}$$

with optimal solution $\tilde{\mathbf{x}} = (\frac{7}{10}, 0, 0)$ and value 14. Demonstrating principle [12.10], relaxation value 14 provides a lower bound on true optimal value 16.

Optimal Solutions from Relaxations

Sometimes relaxations not only bound the optimal value of the corresponding discrete model but produce an optimal solution.

Principle 12.11 If an optimal solution to a constraint relaxation is also feasible in the model it relaxes, the solution is optimal in that original model.

Another look at Figure 12.1 will show why. All (shaded-area) feasible solutions to the original discrete model must also belong to the larger relaxation feasible set. If the relaxation optimum happens to be one of them, it has as good an objective function value as any feasible solution to the relaxation. In particular, it has as good an objective function value as any feasible solution in the original model. It must be optimal in the full model.

The third, linear programming relaxation of the Bison Boosters model in Table 12.2 illustrates. Even though integrality was not required of y -components in the relaxation optimal solution

$$\tilde{x}_1 = 200, \quad \tilde{x}_2 = 0, \quad \tilde{y}_1 = 1, \quad \tilde{y}_2 = 0$$

it happened anyway. This relaxation optimum is feasible in the full discrete model and so optimal there.

This easy justification above, for constraint relaxations, requires a bit of refinement when dealing with relaxations that modify objective functions.

Principle 12.12 A relaxation optimum is optimal if it satisfies the requirements of [12.10](#), and in addition, its objective value in the relaxation is equal to its objective value in the full model.

To see why this additional requirement is needed, imagine adding a new point in the gray region of Figure 12.1 that is the optimal solution to the relaxation. With different objective functions being used, there is no reason to expect that relaxation optimum to coincide with the one for the full model.

Assuming the minimize case, the full-model objective value of the relaxation optimum provides an upper bound on the full-model's optimal value because the relaxation optimum is feasible there. We also know from [12.10](#) that its objective value in the relaxation lower-bounds the full-model optimal value. The only way to be sure these upper and lower bounds coincide is to require it. Full-model and relaxation solution values must match if the relaxation optimum is to be taken as an optimal solution to the underlying problem.

EXAMPLE 12.7: OBTAINING OPTIMAL SOLUTIONS FROM RELAXATIONS

Compute (by inspection) optimal solutions to each of the following relaxations, and determine whether we can conclude that the relaxation optimum is optimal in the original model.

(a) The linear programming relaxation of

$$\begin{array}{ll} \max & 20x_1 + 8x_2 + 2x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

(b) The linear programming relaxation of

$$\begin{array}{ll} \max & x_1 + x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

(c) The relaxation obtained by dropping the first main constraint and reducing the objective coefficient from 8 to 5 of

$$\begin{array}{ll} \min & 2x_1 + 4x_2 + 8x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 2 \\ & 10x_1 + 3x_2 + x_3 \geq 8 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

(d) The linear programming relaxation of

$$\begin{array}{ll} \max & 20x_1 + 8x_2 + 2x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

with objective function coefficient 2 increased to 40.

Solution:

(a) The linear programming relaxation of this model is

$$\begin{array}{ll} \max & 20x_1 + 8x_2 + 2x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 1 \\ & 1 \geq x_1, x_2, x_3 \geq 0 \end{array}$$

with obvious optimal solution $\tilde{\mathbf{x}} = (1, 0, 0)$. Since this solution is also feasible in the original model, it follows from principle [12.11](#) that it is optimal there.

(b) The linear programming relaxation of this model is

$$\begin{array}{ll} \max & x_1 + x_2 + x_3 \\ \text{s.t.} & x_1 + x_2 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x_1 + x_3 \leq 1 \\ & 1 \geq x_1, x_2, x_3 \geq 0 \end{array}$$

with optimal solution $\tilde{\mathbf{x}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Since this solution violates integrality requirements in the original model, it is infeasible there. It could not be optimal.

(c) The indicated relaxation is

$$\begin{array}{ll} \min & 2x_1 + 4x_2 + 5x_3 \\ \text{s.t.} & 10x_1 + 3x_2 + x_3 \geq 8 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

with obvious optimal solution $\tilde{\mathbf{x}} = (1, 0, 0)$. This relaxation optimum satisfies relaxed constraint

$$x_1 + x_2 + x_3 \leq 2$$

and so is feasible in the original model. Furthermore, relaxation and full-problem objective values agree. It follows from principle [12.12](#) that the relaxation optimum solves the full model.

(d) The linear programming relaxation of this model is

$$\begin{array}{ll} \max & 20x_1 + 8x_2 + 40x_3 \\ \text{s.t.} & x_1 + x_2 + x_3 \leq 1 \\ & 1 \geq x_1, x_2, x_3 \geq 0 \end{array}$$

with obvious optimal solution $\tilde{\mathbf{x}} = (0, 0, 1)$ and objective value 40. This solution is indeed feasible in the original model, and its objective value of 40 is a valid upper bound on the full model solution of $\mathbf{x}^* = (1, 0, 0)$ with value 20 (see part (a)). But the two solution values are not equal, principle [12.12](#) does not apply, and we cannot accept the relaxation optimum as a solution to the full model.

Rounded Solutions from Relaxations

When principle [12.12](#) applies, relaxation completely solves a hard discrete optimization model. More commonly, things are not that simple. As with the EMS solution (12.4) above, relaxation optima usually violate some constraints of the true model.

All is hardly lost. First, we have the bound of principle [12.10](#). We may also have a starting point for constructing a good heuristic solution to the full discrete model.

Principle 12.13 Many relaxations produce optimal solutions that are easily “rounded” to good feasible solutions for the full model.

Consider, for example, the EMS solution (12.4). The nature of model constraints (12.3), \geq form with nonnegative coefficients on the left-hand side, means that feasibility of a solution is not lost if we increase some of its components. Beginning from the LP relaxation optimum and rounding up produces the approximate optimal solution

$$\begin{aligned}
 \hat{x}_1 &= \lceil \tilde{x}_1 \rceil = \lceil 0 \rceil = 0 \\
 \hat{x}_2 &= \lceil \tilde{x}_2 \rceil = \lceil 1 \rceil = 1 \\
 \hat{x}_3 &= \lceil \tilde{x}_3 \rceil = \lceil 1 \rceil = 1 \\
 \hat{x}_4 &= \lceil \tilde{x}_4 \rceil = \lceil \frac{1}{2} \rceil = 1 \\
 \hat{x}_5 &= \lceil \tilde{x}_7 \rceil = \lceil \frac{1}{2} \rceil = 1 \\
 \hat{x}_6 &= \lceil \tilde{x}_6 \rceil = \lceil \frac{1}{2} \rceil = 1 \\
 \hat{x}_7 &= \lceil \tilde{x}_7 \rceil = \lceil 0 \rceil = 0 \\
 \hat{x}_8 &= \lceil \tilde{x}_8 \rceil = \lceil 1 \rceil = 1 \\
 \hat{x}_9 &= \lceil \tilde{x}_9 \rceil = \lceil \frac{1}{2} \rceil = 1 \\
 \hat{x}_{10} &= \lceil \tilde{x}_{10} \rceil = \lceil 1 \rceil = 1
 \end{aligned} \tag{12.5}$$

with value $\sum_{j=1}^{10} \hat{x}_j = 8$. Here **ceiling** notation

$$\lceil x \rceil \triangleq \text{least integer greater than or equal to } x$$

The corresponding **floor** notation

$$\lfloor x \rfloor \triangleq \text{greatest integer less than or equal to } x$$

Heuristic optimum $\hat{\mathbf{x}}$ may not be truly optimal, but it does satisfy all constraints. Where time permits no deeper analysis, this rounded relaxation solution might well suffice. Also, feasible solutions provide bounds to complement those obtained from the optimal relaxation solution value (principle [12.10](#)).

Principle 12.14 The objective function value of any (integer) feasible solution to a maximizing discrete optimization problem provides a lower bound on the integer optimal value, and any (integer) feasible solution to a minimizing discrete optimization problem provides an upper bound.

Set covering relaxation optima like (12.4) are particularly easy to round, because of the unusually simple form of the constraints. Many other forms admit similar rounding. Some round infeasible relaxation solutions up, some round down, and some do other straightforward patching. Details vary with model form.

Unfortunately, there are some discrete models that just do not round. For an example, return to our AA airline crew scheduling model (11.10) (Section 11.3). Its set partitioning form closely resembles the set covering case we just rounded easily. But set partitioning involves equality constraints. Each time we round some infeasible \tilde{x}_j up to 1 or down to 0, other variables sharing constraints with that x_j will also have to be adjusted if feasibility is to be preserved. Much more complex rounding schemes are required, and success cannot be guaranteed.

EXAMPLE 12.8: ROUNDING RELAXATION OPTIMA

In each of the following integer linear programs, develop and apply a scheme for rounding the indicated LP relaxation optimum to an approximate solution for the full model. Also, indicate the best lower and upper bounds on the optimal integer solution value available from relaxation and rounding.

$$\begin{aligned} \text{(a) } \min \quad & 10x_1 + 8x_2 + 18x_3 && \text{with LP relaxation optimum } \tilde{\mathbf{x}} = (0, 1, \frac{1}{7}) \\ \text{s.t.} \quad & 2x_1 + 4x_2 + 7x_3 \geq 5 \\ & x_1 + x_2 + x_3 \geq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

$$\begin{aligned} \text{(b) } \max \quad & 40x_1 + 2x_2 + 18x_3 && \text{with LP relaxation optimum } \tilde{\mathbf{x}} = (1, 0, \frac{3}{7}) \\ \text{s.t.} \quad & 2x_1 + 11x_2 + 7x_3 \leq 5 \\ & x_1 + x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

$$\begin{aligned} \text{(c) } \min \quad & 3x_1 + 5x_2 + 20x_3 + 14x_4 && \text{with LP relaxation optimum} \\ & && \tilde{\mathbf{x}} = (\frac{16}{3}, \frac{17}{3}, \frac{16}{33}, \frac{17}{33}) \\ \text{s.t.} \quad & x_1 + x_2 = 11 \\ & 3x_1 + 6x_2 = 50 \\ & x_1 \leq 11x_3 \\ & x_2 \leq 11x_4 \\ & x_1, x_2 \geq 0 \\ & x_3, x_4 = 0 \text{ or } 1 \end{aligned}$$

Solution:

(a) All main constraints of this model are \geq form, and coefficients on the left-hand side are nonnegative. Thus increasing feasible variable values cannot cause a violation. We may round up to integer-feasible solution

$$\lceil \tilde{\mathbf{x}} \rceil = (\lceil 0 \rceil, \lceil 1 \rceil, \lceil \frac{1}{7} \rceil) = (0, 1, 1)$$

Substituting this solution in the objective function gives an upper bound (principle [12.14](#)) of 26 on the optimal value. The corresponding lower bound, which is obtained by substituting the relaxation optimal solution (principle [12.10](#)), is 10.57.

(b) All main constraints of this model are \leq form, and coefficients on the left-hand side are nonnegative. Thus decreasing feasible variable values cannot cause a violation. We may round down to integer-feasible solution

$$[\tilde{\mathbf{x}}] = ([1], [0], [\frac{3}{7}]) = (1, 0, 0)$$

Substituting this solution in the objective function gives a lower bound (principle 12.14) of 40 on the optimal value. The corresponding upper bound, which is obtained by substituting the relaxation optimal solution (principle 12.10), is 47.71.

(c) Each of the discrete variables in this mixed-integer linear program occurs in only one \leq constraint on the right-hand side. Thus increasing x_3 and x_4 from their relaxation values cannot lose feasibility. We may round up to

$$(\frac{16}{3}, \frac{17}{3}, [\frac{16}{33}], [\frac{17}{33}]) = (\frac{16}{3}, \frac{17}{3}, 1, 1)$$

Notice that continuous variable values were not changed.

Substituting this solution in the objective function gives an upper bound (principle 12.14) of 78.33 on the optimal value. The corresponding lower bound, which is obtained by substituting the relaxation optimal solution (principle 12.10), is 61.24.

Stronger LP Relaxations

It should be obvious that we can detect infeasibility quicker (principle 12.9) obtain sharper bounds (principles 12.10 and 12.14), have a better chance of discovering an optimal solution (principles 12.11 and 12.12), and find rounding easier (principle 12.13) if the relaxations we employ more closely approximate the full model of interest. **Strong relaxations** do exactly that.

The Telemark facilities location model of Section 11.6 illustrates a classic case. The model formulated there is:

$$\begin{aligned} \min \quad & \sum_{i=1}^8 \sum_{j=1}^{14} (d_j r_{i,j}) x_{i,j} + \sum_{i=1}^8 f_i y_i && \text{(total fixed cost)} \\ \text{s.t.} \quad & \sum_{i=1}^8 x_{i,j} = 1 && \text{for all } j = 1, \dots, 14 \quad \text{(carry } j \text{ load)} \\ & 1500 y_i \leq \sum_{j=1}^{14} d_j x_{i,j} && \text{for all } i = 1, \dots, 8 \quad \text{(minimum at } i) \\ & \sum_{j=1}^{14} d_j x_{i,j} \leq 5000 y_i && \text{for all } i = 1, \dots, 8 \quad \text{(maximum at } i) \\ & x_{i,j} \geq 0 && \text{for all } i = 1, \dots, 8; \quad j = 1, \dots, 14 \\ & y_i = 0 \text{ or } 1 && \text{for all } i = 1, \dots, 8 \end{aligned} \tag{12.6}$$

where $x_{i,j}$ is the fraction of region j 's call traffic handled by center i , y_i decides whether or not center i is opened, d_j is the anticipated call demand from region j , $r_{i,j}$ is the unit cost of calls from region j to center i , and f_i is the fixed cost of opening center i .

Focus on the third, maximum capacity set of constraints. Each forces discrete variable y_i to take on a value in the relaxation satisfying

$$y_i \geq \frac{\sum_{j=1}^{14} d_j x_{i,j}}{5000} \triangleq \frac{\text{capacity used}}{\text{total available}}$$

For discrete modeling, these constraints do fine. Each y_i must equal 1 if corresponding x -variables are to use facility i at any level. In the LP relaxation, however, if x -variables use only a small part of the capacity, the corresponding y_i will take on a small fractional value.

The numerical values of Section 11.6 confirm this behavior. The LP relaxation of formulation (12.6) has

$$\begin{aligned} \tilde{y}_1 &= 0.230, & \tilde{y}_2 &= 0.000, & \tilde{y}_3 &= 0.000, & \tilde{y}_4 &= 0.301 \\ \tilde{y}_5 &= 0.115, & \tilde{y}_6 &= 0.000, & \tilde{y}_7 &= 0.000, & \tilde{y}_8 &= 0.650 \\ \text{total cost} &= \$8036.60 \end{aligned} \tag{12.7}$$

with many of the \tilde{y}_j small.

Compare the optimal mixed-integer solution

$$\begin{aligned} y_1^* &= 0, & y_2^* &= 0, & y_3^* &= 0, & y_4^* &= 1 \\ y_5^* &= 0, & y_6^* &= 0, & y_7^* &= 0, & y_8^* &= 1 \\ \text{total cost} &= \$10,153 \end{aligned} \tag{12.8}$$

Bound \$8036 of (12.7) is only 79% of true optimal value \$10,153. Also, (12.7) suggests that 4 centers may be needed, while the optimum opens only 2.

Even when a center is used only fractionally, it may fulfill the whole demand for some single district. Such thinking suggests inequalities

$$x_{i,j} \leq y_i \quad \text{for all } i = 1, \dots, 8; \quad j = 1, \dots, 14 \tag{12.9}$$

which require that the fraction a center is opened be as great as the fraction of any region's demand satisfied from the center.

Adding these valid inequalities improves the LP relaxation dramatically. The strengthened model has optimal solution

$$\begin{aligned} \tilde{y}_1 &= 0.000, & \tilde{y}_2 &= 0.000, & \tilde{y}_3 &= 0.000, & \tilde{y}_4 &= 0.537 \\ \tilde{y}_5 &= 0.000, & \tilde{y}_6 &= 0.000, & \tilde{y}_7 &= 0.000, & \tilde{y}_8 &= 1.000 \\ \text{total cost} &= \$10,033.68 \end{aligned}$$

Its bound \$10,033 is almost 99% of optimal value \$10,153, and only one discrete variable comes out fractional. Addition of inequalities (12.9) has produced a much stronger relaxation, which provides much better information about the form of a discrete optimum.

Principle 12.15 Equally correct integer linear programming formulations of a discrete problem may have dramatically different linear programming relaxation optima.

EXAMPLE 12.9: UNDERSTANDING STRONGER LP RELAXATIONS

Show (by inspection) that even though the two following integer linear programming models have the same feasible solutions, the second yields a stronger linear programming relaxation.

$$\begin{array}{ll}
 \max & x_1 + x_2 + x_3 \\
 \text{s.t.} & x_1 + x_2 \leq 1 \\
 & x_1 + x_3 \leq 1 \\
 & x_2 + x_3 \leq 1 \\
 & x_1, x_2, x_3 = 0 \text{ or } 1
 \end{array}
 \qquad
 \begin{array}{ll}
 \max & x_1 + x_2 + x_3 \\
 \text{s.t.} & x_1 + x_2 \leq 1 \\
 & x_1 + x_3 \leq 1 \\
 & x_2 + x_3 \leq 1 \\
 & x_1 + x_2 + x_3 \leq 1 \\
 & x_1, x_2, x_3 = 0 \text{ or } 1
 \end{array}$$

Solution: Both ILPs have the same feasible solutions,

$$\begin{aligned}
 \mathbf{x}^{(1)} &= (1, 0, 0) \\
 \mathbf{x}^{(2)} &= (0, 1, 0) \\
 \mathbf{x}^{(3)} &= (0, 0, 1)
 \end{aligned}$$

Thus they are both valid models of the same problem. Still, the first has LP relaxation optimum $\tilde{\mathbf{x}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and the second has relaxation optimum $\tilde{\mathbf{x}} = (1, 0, 0)$ (among others). The corresponding relaxation bounds are $\frac{3}{2}$ and 1, making the second relaxation stronger. In this simple case, in fact, it yields a discrete optimum (via principle [12.12](#)).

Choosing Big- M Constants

The “sufficiently large” big- M constants needed in so many models offer one easy family where details of ILP modeling affect the LP relaxation. Return, for instance, to the tiny Bison Boosters model of (12.2) and Table 12.2. In formulating switching constraints $x_1 \leq 400y_1$ and $x_2 \leq 75y_2$, we constructed values 400 and 75 with a back-of-envelope computation. Any sufficiently large M would yield a correct integer linear programming model.

Suppose that we had used 10,000 for both. The new model is

$$\begin{array}{ll}
 \max & 20x_1 + 30x_2 - 550y_1 - 720y_2 \quad (\text{net income}) \\
 \text{s.t.} & 1.5x_1 + 4x_2 \leq 300 \quad (\text{display space}) \\
 & x_1 \leq 10,000y_1 \quad (\text{T-shirts if equipment}) \\
 & x_2 \leq 10,000y_2 \quad (\text{sweatshirts if equipment}) \\
 & x_1, x_2 \geq 0 \\
 & y_1, y_2 = 0 \text{ or } 1
 \end{array} \tag{12.10}$$

Recall that the original model (12.2) had relaxation optimum

$$\tilde{x}_1 = 200, \quad \tilde{x}_2 = 0, \quad \tilde{y}_1 = 1, \quad \tilde{y}_2 = 0$$

matching perfectly the discrete optimal solution with value \$3450. Its LP relaxation was indeed strong.

Revision (12.10) is every bit as correct as the original (12.2) in the sense that it has exactly the same (discrete) feasible set. However, the LP relaxation of (12.10) yields optimum

$$\tilde{x}_1 = 200, \quad \tilde{x}_2 = 0, \quad \tilde{y}_1 = 0.02, \quad \tilde{y}_2 = 0 \quad (12.11)$$

with value \$3989. The value bound \$3989 now differs significantly from the true optimal value \$3450. Also, the relaxation optimal solution has component \tilde{y}_1 at a tiny fractional value. With only (12.11) at hand, it would be hard to tell whether to rent or not the T-shirt equipment.

This contrast between LP relaxations of integer-equivalent models (12.2) and (12.10) highlights an important and easy-to-implement principle for strengthening relaxations.

Principle 12.16 | Whenever a discrete model requires sufficiently large big- M 's, the strongest relaxations will result from models employing the smallest valid choice of those constants.

EXAMPLE 12.10: CHOOSING SMALLEST BIG- M 'S

We wish to decide which combination of two pharmaceutical facilities should be used to produce 80 units of a needed product. One costs \$5000 to setup and has variable cost \$20 unit. The other cost \$7000 to setup and has variable cost \$15. Both have capacity of 200 units.

- Formulate a mixed-integer linear programming model using capacities for needed big- M 's.
- Strengthen the linear programming relaxation of your model in part (a) by reducing big- M 's to their smallest valid value.

Solution:

- Using decision variables x_1 and x_2 for the amount produced in each facility, and switching variables x_3 and x_4 to track setups, a valid formulation is

$$\begin{aligned} \min \quad & 20x_1 + 15x_2 + 5000x_3 + 7000x_4 \\ \text{s.t.} \quad & x_1 + x_2 = 80 \\ & x_1 \leq 200x_3 \\ & x_2 \leq 200x_4 \\ & x_1, x_2 \leq 0 \\ & x_3, x_4 = 0 \text{ or } 1 \end{aligned}$$

Full capacity is available whenever setup cost is paid.

- Although capacities are 200, the problem calls for only 80 units to be produced. Thus neither x_1 nor x_2 will ever exceed 80 in an optimal solution. We may strengthen the model by reducing big- M constants from 200 to 80, to produce

$$\begin{aligned}
 \min \quad & 20x_1 + 15x_2 + 5000x_3 + 7000x_4 \\
 \text{s.t.} \quad & x_1 + x_2 = 80 \\
 & x_1 \leq 80x_3 \\
 & x_2 \leq 80x_4 \\
 & x_1, x_2 \geq 0 \\
 & x_3, x_4 = 0 \text{ or } 1
 \end{aligned}$$

The reader can verify that this new formulation has relaxation optimum $\tilde{\mathbf{x}} = (80, 0, 1, 0)$ with value \$6600, versus the original model's $\tilde{\mathbf{x}} = (80, 0, 4, 0)$ at value \$3600.

12.3 BRANCH AND BOUND SEARCH

Total enumerations of Section 12.1 are impractical for all but the simplest models because every one of an explosively growing number of discrete solutions must be considered explicitly. The process would become much more manageable if we could deal with those solutions in large classes, determining for each whole class whether it is likely to contain optimal solutions, and doing so without explicit enumeration of all its members. Only the most promising classes would have to be searched in detail.

Branch and bound algorithms combine such a partial or subset enumeration strategy with the relaxations of Section 12.2. They systematically form classes of solutions and investigate whether the classes can contain optimal solutions by analyzing associated relaxations. More detailed enumeration ensues only if the relaxations fail to be definitive.

APPLICATION 12.2: RIVER POWER

As with so many other topics, an artificially small example will aid in our development of branch and bound ideas. Here we consider an operations problem at River Power Company.

River Power has 4 generators currently available for production and wishes to decide which to put on line to meet the expected 700-megawatt peak demand over the next several hours. The following table shows the cost to operate each generator (in thousands of dollars per hour) and their outputs (in megawatts).

	Generator, j			
	1	2	3	4
Operating cost	7	12	5	14
Output power	300	600	500	1600

Units must be completely on or completely off.

We can formulate River Power's problem as a knapsack problem like those of Section 11.2. Decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if generator } j \text{ is turned on} \\ 0 & \text{otherwise} \end{cases}$$

Then a model is

$$\begin{aligned}
 \min \quad & 7x_1 + 12x_2 + 5x_3 + 14x_4 && \text{(total cost)} \\
 \text{s.t.} \quad & 300x_1 + 600x_2 + 500x_3 + 1600x_4 \geq 700 && \text{(demand)} \\
 & x_1, x_2, x_3, x_4 = 0 \text{ or } 1 && (12.12)
 \end{aligned}$$

The objective function minimizes total operating costs, and the main constraint assures that the chosen combination of generators will fulfill demand. Total enumeration establishes that an optimal solution use generators 1 and 3 and cost \$12,000.

Partial Solutions

Much like the improving searches of most of this book, branch and bound searches iterate through a sequence of solutions until we are ready to conclude optimality or stop with the best fully feasible solution found so far. What is new is that branch and bound searches through partial solutions.

Definition 12.17 | A **partial solution** has some decision variables fixed, with others left **free** or undetermined. We denote free components of a partial solution by the symbol #.

For example, in the River Power model (12.12), $\mathbf{x} = (1, \#, 0, \#)$ specifies a partial solution with $x_1 = 1$, $x_3 = 0$, while x_2 and x_4 remain free.

Completions of Partial Solutions

Each partial solution implicitly defines a class of full solutions called its completions.

Definition 12.18 | The **completions** of a partial solution to a given model are the possible full solutions agreeing with the partial solution on all fixed components.

For instance, the completions of partial solution $\mathbf{x} = (1, \#, \#, 0)$ in our River Power model are

$$(1, 0, 0, 0), \quad (1, 0, 1, 0), \quad (1, 1, 0, 0), \quad \text{and} \quad (1, 1, 1, 0)$$

Every solution with $x_1 = 1$ and $x_4 = 0$ is among these 4. The last 3 are **feasible completions** because they satisfy all constraints of model (12.12).

EXAMPLE 12.11: UNDERSTANDING PARTIAL SOLUTIONS AND COMPLETIONS

Suppose an integer program as decision variables $x_1, x_2, x_3 = 0$ or 1. List all completions of each of the following partial solutions.

- (a) $(1, \#, \#)$
- (b) $(1, \#, 0)$

Solution: We apply definitions [12.17] and [12.18]

- (a) Completions of this partial solution consist of all full solutions with $x_1 = 1$ [i.e., $(1, 0, 0)$, $(1, 0, 1)$, $(1, 1, 0)$, and $(1, 1, 1)$].
- (b) Completions of this partial solution consist of all full solutions with $x_1 = 1$ and $x_3 = 0$ [i.e., $(1, 0, 0)$ and $(1, 1, 0)$].

Tree Search

Branch and bound investigates classes of solutions corresponding to completions of partial solutions in a treelike fashion that gives it the “branch” part of its name. Figure 12.2 provides a full example for River Power model (12.12). Nodes of this **branch and bound tree** represent partial solutions, with numbers indicating the sequence in which they are investigated. Edges or links of the tree specify how variables are fixed in partial solutions. For example, partial solution $\mathbf{x}^{(6)}$ in the sequence has $x_4 = 0$ and $x_2 = 0$.

The process begins at root node 0.

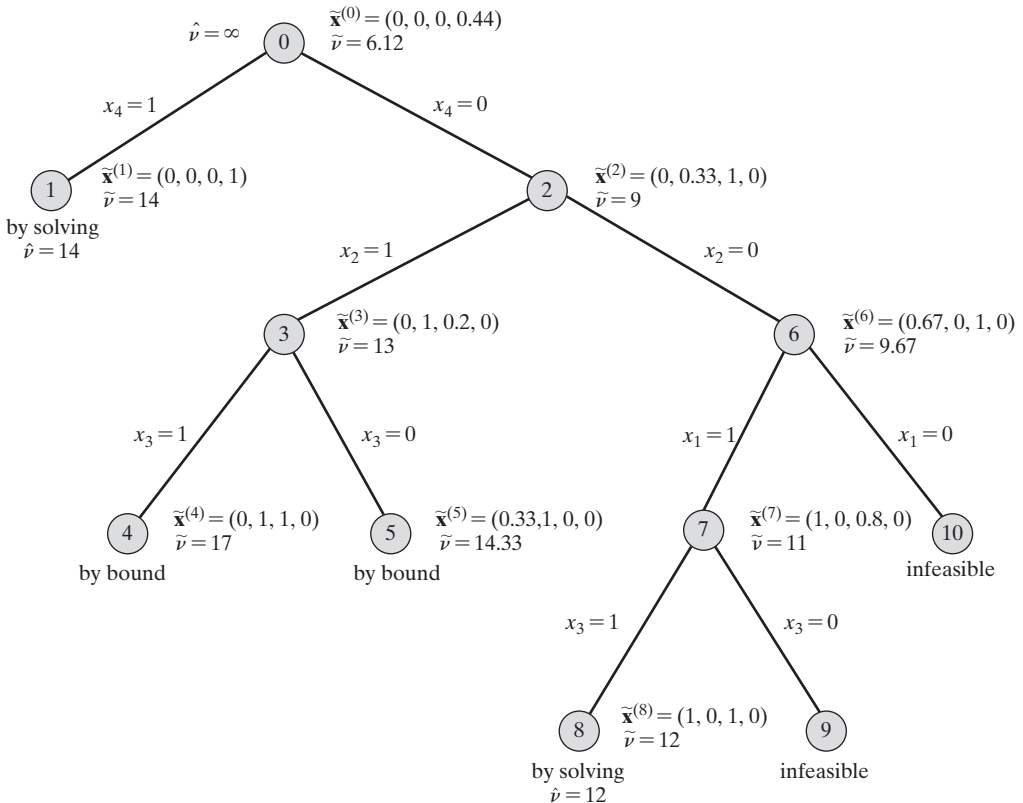


FIGURE 12.2 Branch and Bound Tree for River Power Application

Principle 12.19 | Branch and bound search begins at initial or **root** partial solution $\mathbf{x}^{(0)} = (\#, \dots, \#)$ with all variables free.

This provides the first **active** or unanalyzed partial solution.

At any stage of the search one or more active nodes (distinguished by not yet having a number) remain in the tree. Analysis of each node or partial solution attempts to decide which, if any, completions warrant consideration as an overall optimal solution. Sometimes we can either find a best completion or conclude that none is worth further investigation. Then we **terminate** or **fathom** the entire class of (completion) solutions represented by the node. That is, we give it no further consideration.

Principle 12.20 | Branch and bound searches terminate or fathom a partial solution when they either identify a best completion or prove that none can produce a feasible solution in the overall model with objective value better than the best so far known.

Node 1 in Figure 12.2 illustrates termination. It has no subsidiary nodes because analysis of partial solution $\mathbf{x}^{(1)} = (\#, \#, \#, 1)$, which we detail below, established that the best possible completion is $\mathbf{x} = (0, 0, 0, 1)$. No further investigation of any solution with $x_4 = 1$ is required.

Observe that this termination dealt, in a single step, with fully half the possible solutions to the River Power model. That is, we enumerated the half of all solutions with $x_4 = 1$ as a class. In this way, the exponentially growing effort to totally enumerate every member was avoided.

Unfortunately, it often happens that analysis is not definitive. In such cases the node or partial solution must be branched.

Principle 12.21 | When a partial solution cannot be terminated in a branch and bound search of a 0–1 discrete optimization model, it is **branched** by creating 2 subsidiary partial solutions derived by fixing a previously free binary variable. One of these partial solutions matches the current except that the variable chosen is fixed = 1, and the other is identical except that the variable is fixed = 0.

Node 2 of Figure 12.2 illustrates the need for branching. Analysis (detailed below) was unable either to find the best completion of partial solution $\mathbf{x}^{(2)} = (\#, \#, \#, 0)$ or to prove that none could be optimal. Thus the node was branched into those numbered 3 and 6. Both have $x_4 = 0$, as in the $\mathbf{x}^{(2)}$. However, previously free variable x_2 has now been fixed. In partial solution 3, it is fixed = 1. In partial solution 6, it is fixed = 0.

Notice that this branching process loses no solutions. Every completion of node 2 has either $x_2 = 0$ or $x_2 = 1$. We have simply constructed 2 smaller classes of solutions in the hope that our analysis will now be strong enough to permit termination.

Since no solutions are lost, the enumeration is complete when all partial solutions have been resolved definitively.

Principle 12.22 | Branch and bound search stops when every partial solution in the tree has been either branched or terminated.

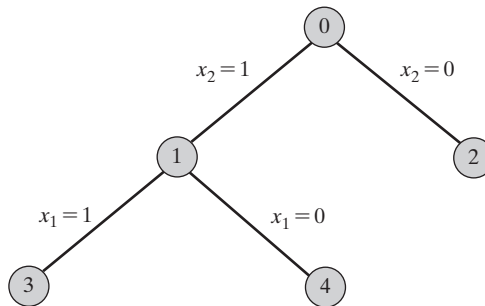
As long as partial solutions do remain, branch and bound search must select an active one to explore next. The simplest such scheme is known as depth first.

Definition 12.23 | **Depth first** search selects at each iteration an active partial solution with the most components fixed (i.e., one deepest in the search tree).

The River Power enumeration of Figure 12.2 employs this depth first rule. For example, after node 3 had been explored, the partial solutions corresponding to nodes 4, 5, and 6 were active in the tree. In accord with depth first rule [12.23], one of the deeper nodes 4 and 5 was selected to investigate next.

EXAMPLE 12.12: UNDERSTANDING BRANCH AND BOUND TREES

The following is the branch and bound tree for a discrete optimization model with decision variables $x_1, x_2, x_3 = 0$ or 1.



- List the sequence of partial solutions explored.
- Identify which of the partial solutions in part (a) were terminated and which branched.
- Determine which nodes would have been active just after processing of node 1, and explain which could have been the next explored under the depth-first enumeration rule.
- Demonstrate that all possible solutions were implicitly enumerated by specifying for each the node at which it was terminated.

Solution:

(a) Under principle [12.19], the first partial solution $\mathbf{x}^{(0)}$ of a branch and bound search is always the all-free $(\#, \#, \#)$. From variable restrictions on branches, we can see subsequent partial solutions visited were $\mathbf{x}^{(1)} = (\#, 1, \#)$, $\mathbf{x}^{(2)} = (\#, 0, \#)$, $\mathbf{x}^{(3)} = (1, 1, \#)$, and $\mathbf{x}^{(4)} = (0, 1, \#)$.

(b) A lack of subsidiary nodes shows that partial solutions 2, 3, and 4 were terminated. Remaining nodes 0 and 1 were branched.

(c) After processing node 1, 3 active partial solutions remained in the tree. None would then have had a number, but they eventually became node 2 created when partial 0 was branched, together with nodes 3 and 4 formed at 1. Depth first enumeration would have taken the search to either of the deeper nodes 3 and 4. However, the search in this example proceeded, instead, to node 2.

(d) A full solution is implicitly enumerated when a completion class to which it belongs is terminated. The following table shows the node at which each of the 8 solutions was terminated in this case.

Solution	Node	Solution	Node	Solution	Node	Solution	Node
(0, 0, 0)	2	(0, 1, 0)	4	(1, 0, 0)	2	(1, 1, 0)	3
(0, 0, 1)	2	(0, 1, 1)	4	(1, 0, 1)	2	(1, 1, 1)	3

Incumbent Solutions

Whether explicit or implicit, the goal of any enumeration is to identify an optimal (or at least a good feasible) solution to some optimization model. To that end, it is essential to keep track of the best known or incumbent solution.

Definition 12.24 The **incumbent solution** at any stage in a search of a discrete model is the best (in terms of objective value) feasible solution known so far. We denote the incumbent solution $\hat{\mathbf{x}}$ and its objective function value \hat{v} .

The incumbent solution may derive from experience prior to the search, or it may have been discovered as the search evolved.

When the search stops, the last incumbent solution is its output. Assuming that the given model has an optimal solution, the final incumbent at least provides an approximate optimum, $\hat{\mathbf{x}}$, with corresponding incumbent solution value \hat{v} . If the search was fully carried out, the optimum is exact.

Principle 12.25 If a branch and bound search stops as in [12.22](#), with all partial solutions having been either branched or terminated, the final incumbent solution is a global optimum if one exists. Otherwise, the model is infeasible.

EXAMPLE 12.13: UNDERSTANDING INCUMBENT SOLUTIONS

Return to the branch and bound tree of Example 12.12 and assume (i) that we were maximizing, (ii) that from prior experience we knew a feasible solution with objective value 10, and (iii) that analysis leading to terminations at nodes 2, 3, and 4 showed the best feasible completions of the corresponding partial solutions have objective values 8, 14, and 12, respectively.

- (a) Show the sequence of incumbent solution objective values.
 (b) Determine the optimal solution value.

Solution:

(a) From assumption (ii) the initial incumbent solution value would have been $\hat{v} = 10$. This value held until node 3 because node 1 was branched and the best completion of node 2 did not improve on 10. At node 3, the search uncovered a feasible solution with better value 14, so this became the incumbent solution value to $\hat{v} = 14$. Node 4 produced no change.

(b) This search implicitly enumerated all possible solutions. Thus, by the principle [12.25], the final incumbent solution value $\hat{v} = 14$ is optimal.

Candidate Problems

Having introduced the tree search underlying branch and bound, we are now ready to see how relaxations of Section 12.2 can make it efficient (and justify the “bound” part of its name). Candidate problems provide the linkage.

Definition 12.26 | The **candidate problem** associated with any partial solution to an optimization model is the restricted version of the model obtained when variables are fixed as in the partial solution.

We may illustrate with partial solution $\mathbf{x}^{(3)} = (\#, 1, \#, 0)$ in River Power Figure 12.2. The corresponding candidate problem is

$$\begin{aligned} \min \quad & 7x_1 + 12x_2 + 5x_3 + 14x_4 \\ \text{s.t.} \quad & 300x_1 + 600x_2 + 500x_3 + 1600x_4 \geq 700 \\ & x_1, x_3 = 0 \text{ or } 1 \\ & x_2 = 1, x_4 = 0 \end{aligned}$$

It derives from original model (12.12) by restricting x_2 and x_4 to their partial solution values.

Thinking about candidate problems aids branch and bound search because of the close connection with completions of the corresponding partial solution.

Principle 12.27 | The feasible completions of any partial solution are exactly the feasible solutions to the corresponding candidate problem, and thus the objective value of the best feasible completion is the optimal objective value of the candidate problem.

That is, we can search for a best completion of any partial solution, or at least learn something about a solution’s objective value, by trying to optimize the corresponding candidate problem.

EXAMPLE 12.14: UNDERSTANDING CANDIDATE PROBLEMS

Consider the ILP

$$\begin{aligned} \max \quad & 10w_1 + 3w_2 + 9w_3 \\ \text{s.t.} \quad & 6w_1 + 4w_2 + 3w_3 \leq 10 \\ & w_1 - w_3 \geq 0 \\ & w_1, w_2, w_3 = 0 \text{ or } 1 \end{aligned}$$

- (a) State the candidate problem corresponding to partial solution $\mathbf{w} = (1, \#, 0)$.
 (b) State the LP relaxation of the candidate problem corresponding to partial solution $\mathbf{w} = (1, \#, 0)$.

Solution:

- (a) Following definition [12.26](#), the required candidate problem is the restricted version with $w_1 = 1, w_3 = 0$:

$$\begin{aligned} \max \quad & 10w_1 + 3w_2 + 9w_3 \\ \text{s.t.} \quad & 6w_1 + 4w_2 + 3w_3 \leq 10 \\ & w_2 - w_3 \geq 0 \\ & w_1 = 1, w_3 = 0 \\ & w_2 = 0 \text{ or } 1 \end{aligned}$$

An optimal solution provides a best completion of the partial $\mathbf{w} = (1, \#, 0)$ (principle [12.27](#)).

- (b) Applying definition [12.6](#), the LP relaxation of the candidate problem in part (a) is

$$\begin{aligned} \max \quad & 10w_1 + 3w_2 + 9w_3 \\ \text{s.t.} \quad & 6w_1 + 4w_2 + 3w_3 \leq 10 \\ & w_2 - w_3 \geq 0 \\ & w_1 = 1, w_3 = 0 \\ & 0 \leq w_2 \leq 1 \end{aligned}$$

Notice that it is a relaxation of the candidate problem in part (a), not a relaxation of the full model.

Terminating Partial Solutions with Relaxations

Now we are ready to take advantage of the relaxation principles in Section 12.2. We analyze nodes in a branch and bound tree by solving relaxations of the corresponding candidate problems. For example, relaxation solutions and solution values shown next to the nodes in River Power Figure 12.2 (denoted $\tilde{\mathbf{x}}$) come from the linear programming relaxation of the corresponding candidate problem.

Begin with infeasibility principle [12.9](#). If any relaxation of a candidate problem is infeasible, so is the full candidate.

Principle 12.28 | If any relaxation of a candidate problem proves infeasible, the associated partial solution can be terminated because it has no feasible completions.

Node 10 of Figure 12.2 illustrates. The LP relaxation of the corresponding candidate problem is infeasible. It follows that partial solution $\mathbf{x}^{(10)} = (0, 0, \#, 0)$ has no feasible completions, and we may terminate infeasible.

Now consider relaxation bound principle [12.10]. Relaxation optimal values bound those of the problem relaxed. Thus in the context of candidate problems, relaxation optimal values bound the objective value of the best possible completion. Comparison to the incumbent solution value (definition [12.24]) can lead to termination.

Principle 12.29 | If any relaxation of a candidate problem has optimal objective value no better than the current incumbent solution value, the associated partial solution can be terminated because no feasible completion can improve on the incumbent.

Node 5 of Figure 12.2 illustrates such termination by bound. When the search reached partial solution $\mathbf{x}^{(5)} = (\#, 1, 0, 0)$, incumbent solution $\hat{\mathbf{x}} = (0, 0, 0, 1)$ with objective value $\hat{v} = 14$ was already in hand (from node 1). The linear programming relaxation of the candidate problem for node 5 had optimal value 14.33, which means that no feasible completion can do better than that value. It follows that none can improve on the incumbent solution in this minimizing problem, and we terminate by bound.

The third way to terminate with relaxations derives from optimality principle [12.12].

Principle 12.30 | If an optimal solution to any constraint relaxation of a candidate problem is feasible in the full candidate, it is a best feasible completion of the associated partial solution. After checking whether a new incumbent has been discovered, the partial solution can be terminated.

Consider node 1 of River Power Figure 12.2. The corresponding LP relaxation fixed $x_4 = 1$ but allowed free variables to take on any value between 0 and 1. Still, relaxation optimum $\tilde{\mathbf{x}} = (0, 0, 0, 1)$ meets integrality requirements on all components. It is optimal in the corresponding candidate problem and the best possible completion of partial solution $\mathbf{x}^{(1)} = (\#, \#, \#, 1)$. It is also the first fully feasible solution encountered in the search, so it provides the first incumbent solution. After saving it as the incumbent, node 1 was terminated by solving.

EXAMPLE 12.15: TERMINATING PARTIAL SOLUTIONS WITH RELAXATIONS

Suppose that a maximizing branch and bound search over $y_1, \dots, y_4 = 0$ or 1 reaches partial solution $\mathbf{y}^{(3)} = (\#, 0, \#, \#)$ with incumbent solution value $\hat{v} = 100$. Explain how the search should proceed assuming each of the following outcomes from an

attempt to solve the linear programming relaxation of the corresponding candidate problem.

- (a) Relaxation optimum $\tilde{\mathbf{y}} = (\frac{1}{3}, 0, 1, 0)$ with objective value $\tilde{\nu} = 85$.
- (b) Relaxation optimum $\tilde{\mathbf{y}} = (1, 0, \frac{1}{2}, 0)$ with objective value $\tilde{\nu} = 100$.
- (c) Relaxation optimum $\tilde{\mathbf{y}} = (0, 0, 1, 1)$ with objective value $\tilde{\nu} = 120$.
- (d) Relaxation infeasible.
- (e) Relaxation optimum $\tilde{\mathbf{y}} = (0, \frac{1}{4}, 1, 0)$ with objective value $\tilde{\nu} = 111$.

Solution:

(a) The relaxation bound demonstrates no feasible completion can do better than 85 in objective value, which is worse than the known incumbent. The partial solution should be terminated as in principle [12.29](#).

(b) The relaxation bound demonstrates that no feasible completion can do better than 100 in objective value, which is the same as the incumbent. The partial solution should be terminated as in principle [12.29](#) (unless alternative optimal solutions are of interest).

(c) This relaxation optimum is feasible in the full candidate problem and thus optimal. Having found the best possible completion, we terminate by principle [12.30](#) after updating the incumbent solution value to improved value $\hat{\nu} \leftarrow 120$.

(d) There are no feasible completions. The partial solution should be terminated as in principle [12.28](#).

(e) Here none of principles [12.28](#) to [12.30](#) lead to termination because the relaxation optimum is better in objective value than the incumbent but still fractional in some components. The partial solution must be branched as in principle [12.21](#).

LP-Based Branch and Bound

The discrete models most frequently solved by branch and bound are ILPs with 0–1 variables. Linear programming relaxations of candidate problems usually provide the basis for analysis.

Algorithm 12A details a formal algorithm for this **LP-based branch and bound** case. For simplicity we assume that the model is not unbounded.

Initialization begins as in principle [12.19](#) with all binary variables free. If no incumbent solution $\hat{\mathbf{x}}$ is known, the worst possible value $\hat{\nu} \leftarrow \pm \infty$ is assumed.

Each main iteration begins by selecting some active partial solution to pursue. Any one can be selected, although sequence does make a difference (see Section 12.4).

Processing begins with an attempt to solve the LP relaxation. Then we check to see if any of termination rules [12.28](#) to [12.30](#) apply. If so, the current partial is terminated and the process repeats. Partial solutions that cannot be terminated must be branched (principle [12.21](#)).

ALGORITHM 12A: LP-BASED BRANCH AND BOUND (0-1 ILPS)

Step 0: Initialization. Make the only active partial solution the one with all discrete variables free, and initialize solution index $t \leftarrow 0$. If any feasible solutions are known for the model, also choose the best as incumbent solution $\hat{\mathbf{x}}$ with objective value \hat{v} . Otherwise, set $\hat{v} \leftarrow -\infty$ if the model maximizes and $\hat{v} \leftarrow +\infty$ if it minimizes.

Step 1: Stopping. If active partial solutions remain, select one as $\mathbf{x}^{(t)}$, and proceed to Step 2. Otherwise, stop. If there is an incumbent solution $\hat{\mathbf{x}}$, it is optimal, and if not, the model is infeasible.

Step 2: Relaxation. Attempt to solve the linear programming relaxation of the candidate problem corresponding to $\mathbf{x}^{(t)}$.

Step 3: Termination by Infeasibility. If the LP relaxation proved infeasible, there are no feasible completions of partial solution $\mathbf{x}^{(t)}$. Terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 4: Termination by Bound. If the model maximizes and LP relaxation optimal value \tilde{v} satisfies $\tilde{v} \leq \hat{v}$, or it minimizes and $\tilde{v} \geq \hat{v}$, the best feasible completion of partial solution $\mathbf{x}^{(t)}$ cannot improve on the incumbent. Terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 5: Termination by Solving. If the LP relaxation optimum $\tilde{\mathbf{x}}^{(t)}$ satisfies all binary constraints of the model, it provides the best feasible completion of partial solution $\mathbf{x}^{(t)}$. After saving it as new incumbent solution

$$\begin{aligned}\hat{\mathbf{x}} &\leftarrow \tilde{\mathbf{x}}^{(t)} \\ \hat{v} &\leftarrow \tilde{v}\end{aligned}$$

terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 6: Branching. Choose some free binary-restricted component x_p that was fractional in the LP relaxation optimum, and branch $\mathbf{x}^{(t)}$ by creating two new actives. One is identical to $\mathbf{x}^{(t)}$ except that x_p is fixed = 0, and the other the same except that x_p is fixed = 1. Then increment $t \leftarrow t + 1$ and return to Step 1.

Branching Rules for LP-Based Branch and Bound

One peculiarity of the LP-based form of branch and bound is that this is done on a fractional free variable.

Principle 12.31 | LP-based branch and bound algorithms always branch by fixing an integer-restricted decision variable that had a fractional value in the associated candidate problem relaxation.

For example, at node 0 of in Figure 12.2, branching occurred by fixing x_4 , which had fractional value $\tilde{x}_4 = 0.44$ in the LP relaxation.

The motivation behind this fractional variable branching rule is avoiding duplicate computation. The alternative of branching on a free variable that came out integer in the LP relaxation produces one new candidate problem guaranteed to have the same relaxation optimum as the one just solved. For example, branching on x_1 at node 0 of Figure 12.2 would have created new partial solutions

$$\mathbf{x}^{(1)} = (1, \#, \#, \#) \quad \text{and} \quad \mathbf{x}^{(2)} = (0, \#, \#, \#)$$

But the relaxation optimum for the second would be exactly the same as $\tilde{\mathbf{x}}^{(0)} = (0, 0, 0, 0.438)$ because added constraint $x_1 = 0$ can affect solution value only if it is violated by the previous optimum. We prefer to get new information from both candidates.

Somewhat similar thinking applies when more than one integer variable is fractional in the relaxation optimum.

Principle 12.32 | When more than one integer-restricted variable is fractional in the relaxation optimum, LP-based branch and bound algorithms often branch by fixing the one closest to an integer value.

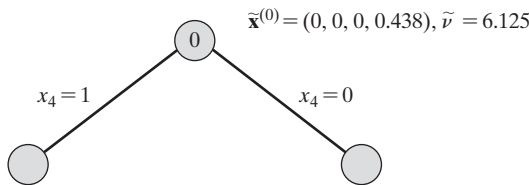
For example, if the relaxation optimum were $\tilde{\mathbf{x}} = (0.3, 1, 0.5, 0.9)$, and all components were supposed to be binary, rule [12.32](#) would next fix x_4 because it is the fractional variable closest to an integer.

The motivation behind rule [12.32](#) is to take the most obvious decision, hoping that the farthest away of the two resulting partial solutions can be terminated before it has to be explored. Still, many other schemes are possible. Commercial codes often give the modeler an opportunity to explicitly designate priority variables for branching.

LP-Based Branch and Bound Solution of the River Power Application

We are now ready to fully trace Figure 12.2’s branch and bound solution of River Power model (12.12).

The process begins with all-free partial solution $\mathbf{x}^{(0)} = (\#, \#, \#, \#)$ and $\hat{v} = +\infty$. The corresponding LP relaxation optimum is fractional on x_4 , so the node is branched (principle [12.31](#)) as



There are now 2 active partial solutions, and we must choose one to process next. Any active partial solution could be chosen, but all Figure 12.2 computation employs the depth first rule [12.23](#). That is, it chooses an active partial solution deepest in the tree.

At the moment both active candidates have equal depth because both have 1 fixed variable. Figure 12.2 adopts the simple tie-breaking rule of always selecting the partial solution with the branching variable fixed = 1. (See definition [12.40] of Section 12.4 for another possibility.)

That tie-breaking rule leads us to $\mathbf{x}^{(1)} = (\#, \#, \#, 1)$. Here the relaxation solution is binary on all components. Thus we have found the best completion, and we terminate (principle [12.30]) after saving incumbent solution

$$\hat{\mathbf{x}} \leftarrow (0, 0, 0, 1) \quad \text{with} \quad \hat{\nu} \leftarrow 14$$

The only remaining active is now partial solution $\mathbf{x}^{(2)} = (\#, \#, \#, 0)$. Relaxation optimum $\tilde{\mathbf{x}}^{(2)} = (0, 0.333, 1, 0)$ fails all termination tests [12.28] to [12.30] because it is feasible, fractional, and has solution value 9 strictly better than $\hat{\nu} = 14$. We must branch on fractional x_2 .

Processing continues in this way until we reach $\mathbf{x}^{(4)} = (\#, 1, 1, 0)$. There the relaxation bound $\tilde{\nu} = 17 \geq \hat{\nu} = 14$, so we terminate (principle [12.29]). Something similar happens at node 5.

After several more steps, we encounter a new incumbent solution at node 8,

$$\hat{\mathbf{x}} \leftarrow (1, 0, 1, 0) \quad \text{with} \quad \hat{\nu} \leftarrow 12$$

Termination of nodes 9 and 10 by infeasibility (principle [12.28]) completes the search. This final incumbent solution is optimal (principle [12.25]).

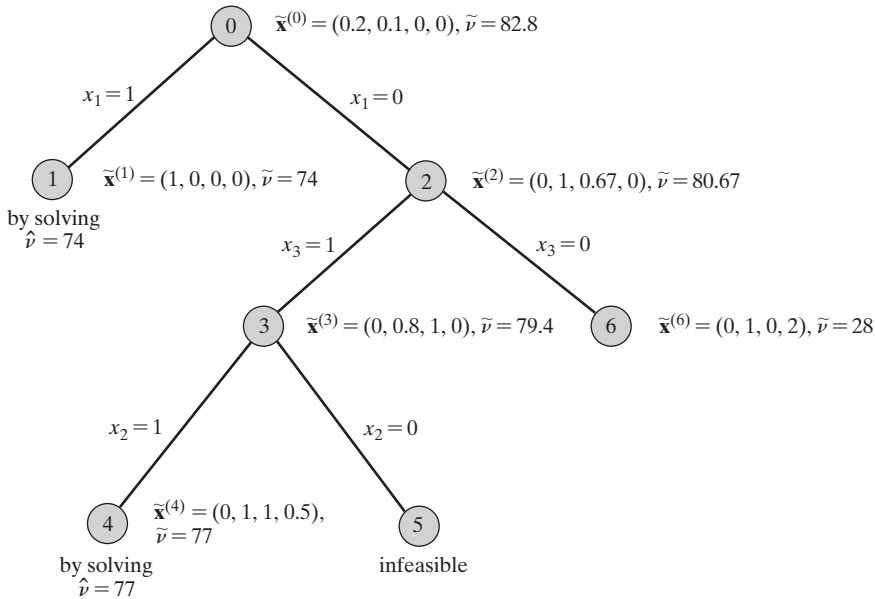
EXAMPLE 12.16: PERFORMING LP-BASED BRANCH AND BOUND

The following table shows candidate problem LP relaxation optima for all possible combinations of fixed and free values in a maximizing mixed-integer linear program over $x_1, x_2, x_3 = 0$ or $1, x_4 \geq 0$.

x_1	x_2	x_3	$\tilde{\mathbf{x}}$	$\tilde{\nu}$	x_1	x_2	x_3	$\tilde{\mathbf{x}}$	$\tilde{\nu}$
#	#	#	(0.2, 1, 0, 0)	82.80	0	0	1	Infeasible	—
#	#	0	(0.2, 1, 0, 0)	82.80	0	1	#	(0, 1, 0.67, 0)	80.67
#	#	1	(0, 0.8, 1, 0)	79.40	0	1	0	(0, 1, 0, 2)	28.00
#	0	#	(0.7, 0, 0, 0)	81.80	0	1	1	(0, 1, 1, 0.5)	77.00
#	0	0	(0.7, 0, 0, 0)	81.80	1	#	#	(1, 0, 0, 0)	74.00
#	0	1	(0.4, 0, 1, 0)	78.60	1	#	0	(1, 0, 0, 0)	74.00
#	1	#	(0.2, 1, 0, 0)	82.80	1	#	1	(1, 0, 1, 0)	63.00
#	1	0	(0.2, 1, 0, 0)	82.80	1	0	#	(1, 0, 0, 0)	74.00
#	1	1	(0, 1, 1, 0.5)	77.00	1	0	0	(1, 0, 0, 0)	74.00
0	#	#	(0, 1, 0.67, 0)	80.67	1	0	1	(1, 0, 1, 0)	63.00
0	#	0	(0, 1, 0, 2)	28.00	1	1	#	(1, 1, 0, 0)	62.00
0	#	1	(0, 0.8, 1, 0)	79.40	1	1	0	(1, 1, 0, 0)	62.00
0	0	#	Infeasible	—	1	1	1	(1, 1, 1, 0)	51.00
0	0	0	Infeasible	—					

Solve the model by LP-based branch and bound Algorithm 12A, applying the same depth first rule for selecting among actives (ties broken in favor of $x_j = 1$) that was implemented in the River Power application of Figure 12.2.

Solution: Applying Algorithm 12A produces the following branch and bound tree:



Processing stops with optimal solution $x^* = (0, 1, 1, 0.5)$ at objective value 77.

Most of the processing is similar to our River Power application. One exception is that this model maximizes. For example, we terminate by bound at node 6 because

$$28 = \tilde{v} \leq \hat{v} = 77$$

The other new element is that this model has continuous variable x_4 as well as 3 binary ones. This makes relaxation optimum $\tilde{x}^{(4)} = (0, 1, 1, 0.5)$ feasible (and so optimal) in the full candidate, even though its last component is fractional. We terminate by having solved the candidate problem.

12.4 REFINEMENTS TO BRANCH AND BOUND

Algorithm 12A contains all the main elements of at least LP-based branch and bound, but it omits many details. In this section we briefly introduce some refinements.

Branch and Bound Solution of NASA Capital Budgeting Application

It will help to have a more serious example to illustrate. For this purpose we employ the NASA mission selection model formulated in Section 11.2. There, the decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if mission } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

The complete formulation of our version was

$$\begin{aligned}
 \max \quad & 200x_1 + 3x_2 + 20x_3 + 50x_4 + 70x_5 && \text{(total value)} \\
 & + 20x_6 + 5x_7 + 10x_8 + 200x_9 + 150x_{10} \\
 & + 18x_{11} + 8x_{12} + 300x_{13} + 185x_{14} \\
 \text{s.t.} \quad & 6x_1 + 2x_2 + 3x_3 + 1x_7 + 4x_9 + 5x_{12} &\leq 10 && \text{(Stage 1)} \\
 & 3x_2 + 5x_3 + 5x_5 + 8x_7 + 5x_9 + 8x_{10} &&& \text{(Stage 2)} \\
 & + 7x_{12} + 1x_{13} + 4x_{14} &\leq 12 \\
 & 8x_5 + 1x_6 + 4x_{10} + 2x_{11} + 4x_{13} + 5x_{14} &\leq 14 && \text{(Stage 3)} \\
 & 8x_6 + 5x_8 + 7x_{11} + 1x_{13} + 3x_{14} &\leq 14 && \text{(Stage 4)} \\
 & 10x_4 + 4x_6 + 1x_{13} + 3x_{14} &\leq 14 && \text{(Stage 5)} && (12.13) \\
 & x_4 + x_5 \leq 1 &&& \text{(mutually} \\
 & x_8 + x_{11} \leq 1 &&& \text{exclusives)} \\
 & x_9 + x_{14} \leq 1 \\
 & x_{11} \leq x_2 &&& \text{(dependent} \\
 & x_4 \leq x_3 &&& \text{missions)} \\
 & x_5 \leq x_3 \\
 & x_6 \leq x_3 \\
 & x_7 \leq x_3 \\
 & x_j = 0 \text{ or } 1 \quad \text{for all } j = 1, \dots, 14
 \end{aligned}$$

Figure 12.3 shows the tree of an LP-based branch and bound search, and Table 12.3 details computations.

Rounding for Incumbent Solutions

One refinement of branch and bound exhibited in these NASA application computations is **rounding** relaxation optima as in principle [12.11](#) to speed the process of finding good incumbent solutions.

Principle 12.33 | If convenient rounding schemes are available, the relaxation optimum for every partial solution that cannot be terminated in a branch and bound search is usually rounded to a feasible solution for the full model prior to branching. The feasible solution provides a new incumbent if it is better than any known.

Computations in NASA Table 12.3 round down; that is, fractional components in relaxation optima are set = 0. Every partial solution that cannot be terminated is rounded before branching, and the resulting feasible solution considered as an incumbent. For instance, at node 5, the candidate relaxation produced optimum

$$\tilde{\mathbf{x}}^{(5)} = (1, 0, 1, 0.6, 0.4, 1, 0, 0.4, 0, 0, 0, 0, 1, 1), \quad \tilde{v} = 787$$

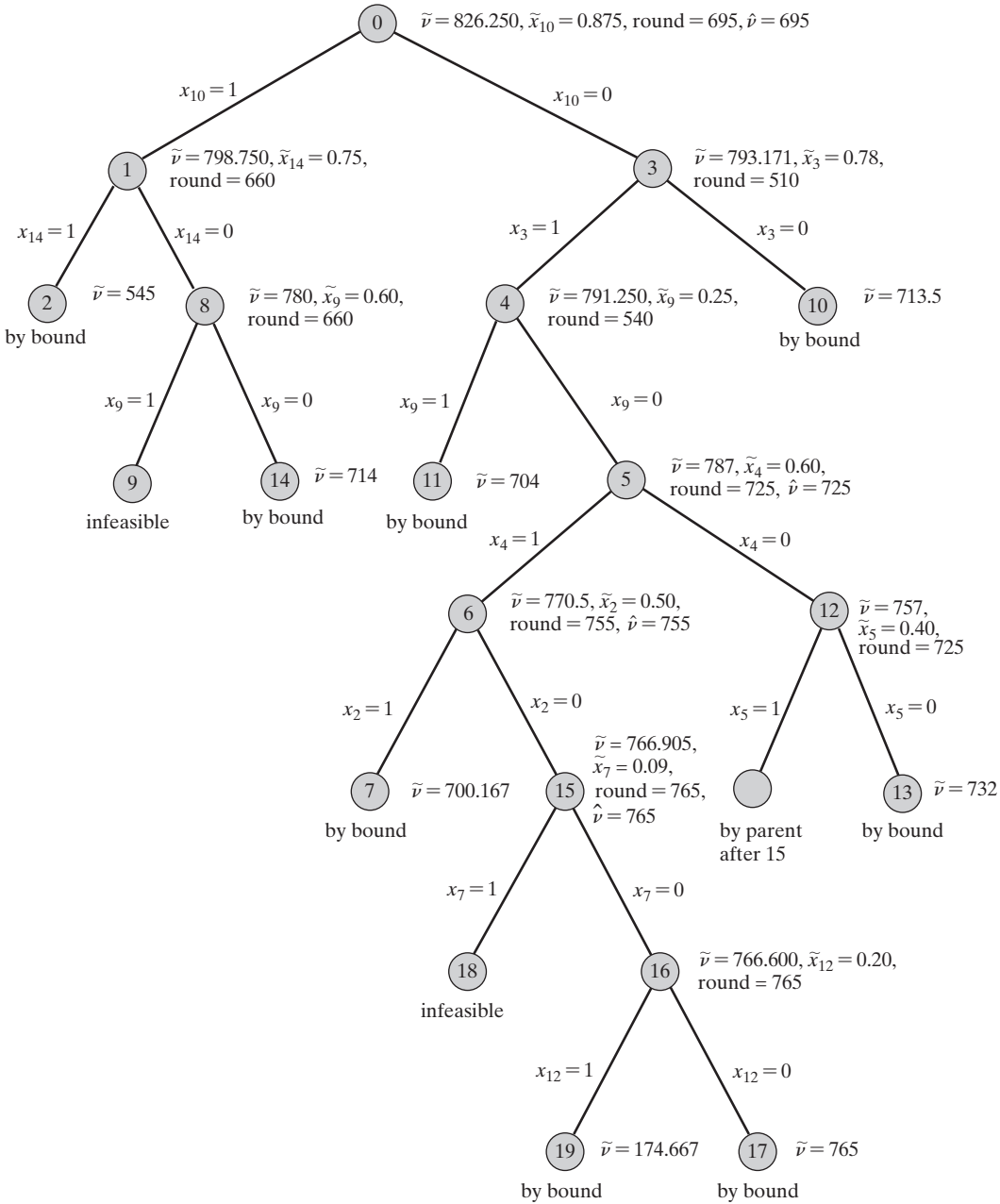


FIGURE 12.3 Branch and Bound Search of NASA Application

The solution is fractional, and bound 787 is insufficient to terminate. Instead of branching immediately, however, the relaxation is first rounded by setting each $\hat{x}_j = [\tilde{x}_j]$. The result is new incumbent solution

$$\hat{\mathbf{x}} = (1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1), \hat{\nu} = 725$$

TABLE 12.3 Branch and Bound Search of NASA Application

t	Relax Value	Relaxation Solution ^a	Round Value	Action
0	826.250	(1, 0, 0, 0, 0, 0, 0, 1, 0, 0.875, 0, 0, 1, 1)	695	First incumbent $\hat{\mathbf{x}} \leftarrow$ (1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1) branch on x_{10}
1	798.750	(1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, <u>1</u> , 0, 0, 1, 0.750)	660	Branch on x_{14}
2	545.000	(1, 0, 0, 0, 0, 0, 0, 1, 0, <u>1</u> , 0, 0, 0, <u>1</u>)	—	Terminate by bound
3	793.171	(1, 0, 0.780, 0.463, 0.537, 0.780, 0, 1, 0.415, <u>0</u> , 0, 0, 1, 0.585)	510	Branch on x_3
4	791.250	(1, 0, <u>1</u> , 0.650, 350, 1, 0, 0.550, 0.250, <u>0</u> , 0, 1, 0.750)	540	Branch on x_9
5	787.000	(1, 0, <u>1</u> , 0.600, 0.400, 1, 0, 0.400, <u>0</u> , <u>0</u> , 0, 0, 1, 1)	725	New incumbent $\hat{\mathbf{x}} \leftarrow$ (1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1) branch on x_4
6	770.500	(1, 0.500, <u>1</u> , <u>1</u> , 0, 0, 0, 0.500, <u>0</u> , <u>0</u> , 0.500, 0, 1, 1)	755	New incumbent $\hat{\mathbf{x}} \leftarrow$ (1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1) branch on x_2
7	700.167	(0.833, <u>1</u> , <u>1</u> , <u>1</u> , 0.188, 0, 0, <u>0</u> , <u>0</u> , 1, 0, 1, 0.750)	—	Terminate by bound
8	780.000	(1, 0, 0, 0, 0, 0, 0, 0, 1, 0.600, <u>1</u> , 0, 0, 1, <u>0</u>)	660	Branch on x_9
9	Infeasible	None	—	Terminate by infeasible
10	713.500	(1, 1, <u>0</u> , 0, 0, 0, 0, 0, 0.500, <u>0</u> , 1, 0, 1, 0.500)	—	Terminate by bound
11	704.000	(0.500, 0, <u>1</u> , 0.800, 0.200, 1, 0, 1, <u>1</u> , <u>0</u> , 0, 0, 1, 0)	—	Terminate by bound
12	757.000	(1, 0, <u>1</u> , <u>0</u> , 0.400, 1, 0, 0.400, <u>0</u> , <u>0</u> , 0, 0, 1, 1)	725	Branch on x_5
13	732.000	(1, 0.462, <u>1</u> , <u>0</u> , 0, 0.846, 0, 0.077, 0, <u>0</u> , <u>0</u> , 0.462, 0, 1, 1)	—	Terminate by bound
14	714.000	(1, 0, 0.600, 0.600, 0, 0.600, 0, 0, 1, <u>0</u> , <u>1</u> , 0, 0, 1, <u>0</u>)	—	Terminate by bound
15	766.909	(1, <u>0</u> , <u>1</u> , <u>1</u> , 0, 0, 0.091, 1, <u>0</u> , <u>0</u> , 0, 0.182, 1, 1)	765	New incumbent $\hat{\mathbf{x}} \leftarrow$ (1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1) branch on x_7
16	766.600	(1, <u>0</u> , <u>1</u> , <u>1</u> , 0, 0, <u>0</u> , 1, <u>0</u> , <u>0</u> , 0, 0.200, 1, 1)	765	Branch on x_{12}
17	765.000	(1, <u>0</u> , <u>1</u> , <u>1</u> , 0, 0, <u>0</u> , 1, <u>0</u> , <u>0</u> , 0, 0, 1, 1)	—	Terminate by bound
18	Infeasible	None	—	Terminate by infeasible
19	174.667	(0.333, <u>0</u> , <u>1</u> , <u>1</u> , 0, 1, 0, <u>1</u> , <u>0</u> , <u>0</u> , 0, <u>1</u> , 0, 0)	—	Terminate by bound

^aUnderlined values are fixed in the partial solution.

Rounding is not guaranteed always to produce a new incumbent. For example, at node 1 of the NASA example, rounding produced a solution of value 660, which was no better than the existing incumbent $\hat{\nu} = 695$.

It is valuable to have good incumbent solutions early in a branch and bound search for two reasons. First, time may not permit the search to be run until every node has been terminated or branched. Then the final incumbent solution provides an approximate optimum. Clearly, we would like it to be as good as possible.

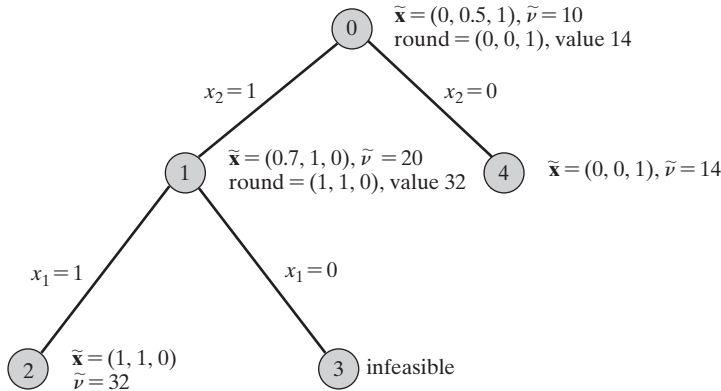
The other advantage of early incumbents comes with termination by bound (principle [12.29](#)). In the maximize case, for example, we terminate if

$$\text{relaxation bound} \triangleq \tilde{\nu} \leq \hat{\nu} \triangleq \text{incumbent value}$$

We can certainly terminate more nodes with the same bound values if a strong incumbent value is discovered quickly.

EXAMPLE 12.17: ROUNDING FOR INCUMBENT SOLUTIONS

The following tree shows Algorithm 12A computation on a minimizing ILP over $x_1, \dots, x_3 = 0$ or 1 . (Rounded solution values are shown but were not used in the computation.)



- (a) Determine the earliest moment in the search at which ultimate optimal solution value 14 was known.
- (b) Repeat the search using rounding to produce earlier incumbent solutions.
- (c) Describe the savings that resulted in part (b) from using rounded solutions.

Solution:

- (a) The final incumbent solution, which proved optimal, was encountered when the relaxation optimum at partial solution 4 solved its candidate problem.
- (b) At node 0 the relaxation optimum has value 10, and it rounds to a feasible solution with value 14. This becomes the first incumbent solution value. We then branch on x_2 as before. At node 1 relaxation bound 20 permits termination because it does not improve on the incumbent (principle 12.29). Then the relaxation at node 4 the verifies optimality of the incumbent solution.
- (c) In the search of part (b) the incumbent was uncovered at node 0, much earlier than node 4 in calculations depicted in the tree. This would be an advantage if we had to stop the search before all nodes had been terminated or branched. Also, the early incumbent avoided some candidate problem computations at nodes 2 and 3. The bound at 1 is now enough to terminate.

Branch and Bound Family Tree Terminology

To discuss issues connected with managing and controlling branch and bound trees, we need some terminology. It is standard to make analogies with family trees.

Any node created directly from another by branching is called a **child**, and the branched node is its **parent**. For example, in Figure 12.3, nodes 11 and 5 are the children of node 4. Its parent is node 3.

EXAMPLE 12.18: UNDERSTANDING TREE TERMINOLOGY

Identify each of the following in the branch and bound tree of Example 12.17.

- (a) The parent of node 3
- (b) The children of node 0

Solution:

- (a) Node 1 is the parent of node 3.
- (b) Nodes 1 and 4 are the children of node 0.

Parent Bounds

Another way of refining LP-based branch and bound Algorithm 12A is to take advantage of an easy **parent bound** readily available from prior computations.

Principle 12.34 | The relaxation optimal value for the parent of any partial solution to a minimize model provides a lower bound on the objective value of any completion of its children. The relaxation optimal value for the parent in a maximize model provides an upper bound.

To see that these bounds are valid, recall that the candidate problem associated with any partial solution is just our original model augmented by constraints for variables fixed in the partial solution (definition [12.26](#)). Extending the solution with a new fix in a child can only worsen the optimal objective function value. For example, in Figure 12.3, relaxation bound 826.250 for node 0 holds for its children nodes 1 and 3. Both have an additional constraint, so the solution value can only worsen. In fact, when relaxation bounds $\tilde{v}^{(1)} = 798.750$ and $\tilde{v}^{(3)} = 793.171$ were later computed, they proved strictly worse (for this maximize model).

EXAMPLE 12.19: UNDERSTANDING PARENT BOUNDS

Identify for node 5 in NASA branch and bound Figure 12.3 the best (upper) bound known for the optimal value of the corresponding candidate problem *before* the candidate's LP relaxation was solved.

Solution: Prior to solving the LP relaxation of the candidate problem for node 5, the best bound available on its optimal value comes from its parent. Parent bound $\tilde{v} = 791.250$ is valid for child node 5 because it is identical except for the extra $x_9 = 0$ constraint.

Terminating with Parent Bounds

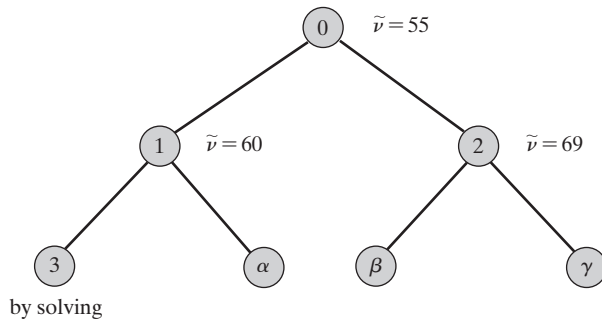
One way a sophisticated branch and bound algorithm can exploit parent bounds [12.34](#) arises whenever a new incumbent solution is discovered.

Principle 12.35 Whenever a branch and bound search discovers a new incumbent solution, any active partial solution with parent bound no better than the new incumbent solution value can immediately be terminated.

Processing of partial solution 15 in Table 12.3 illustrates principle [12.35](#). An incumbent solution was discovered there with value $\tilde{\nu} = 765$. Earlier, partial solution 12 had been branched because its bound of $\tilde{\nu} = 757$ was insufficient to terminate versus the incumbent at that time. Now we may terminate the still-active child with $x_5 = 1$. The 757 bound available from the parent is no better than the incumbent discovered at node 15.

EXAMPLE 12.20: TERMINATING WITH PARENT BOUNDS

The following tree shows a minimizing branch and bound search that has processed 4 partial solutions and just discovered its first incumbent solution at node 3. Nodes marked $\alpha, \beta,$ and γ are active but not yet explored.



Determine which if any active partial solution can be terminated if the incumbent solution discovered at node 3 has value (a) $\hat{\nu} = 80$; (b) $\hat{\nu} = 63$.

Solution: We apply principle [12.35](#), comparing with parent bounds 60 at node α and 69 at nodes β and γ .

(a) Incumbent value $\hat{\nu} = 80$ is not as good as either parent bound for this minimize model. No active solution can be terminated.

(b) Incumbent value $\hat{\nu} = 63$ is not as good as 60, but superior to 69. Thus active partial solutions β and γ should be terminated.

Stopping Early: Branch and Bound as a Heuristic

Table 12.4 provides a historical summary of the NASA application in Figure 12.3 and Table 12.3. Incumbent solutions reported in the first column show a typical pattern. Good incumbents were uncovered fairly early in the search, with the final 50 to 80% of computational effort consumed in squeezing out the last improvements and proving global optimality. For example, nearly optimal $\hat{\nu} = 755$ was discovered after only 7 of the 19 partial solutions analyzed.

TABLE 12.4 Incumbent and Best Parent Bound History in NASA Application, Branch Bound Figure 12.3

After Node	Incumbent Value	Best Parent	After Node	Incumbent Value	Best Parent
0	695	826.250	10	755	791.250
1	695	826.250	11	755	780.000
2	695	826.250	12	755	780.000
3	695	798.750	13	755	780.000
4	695	798.750	14	755	770.500
5	725	798.750	15	765	766.909
6	755	798.750	16	765	766.909
7	755	798.750	17	765	766.909
8	755	793.171	18	765	765.000
9	755	793.171			

Often, we are willing to settle for less than global optimality to avoid the long final phase. That is, we want to **stop early**, accepting the last incumbent as a heuristic optimum.

Bounds on the Error of Stopping with the Incumbent Solution

Of course, we know that such incumbent solutions are feasible for the full problem, and better than any other feasible solution that we have encountered. But we can use the best of current parent bounds [\[12.34\]](#) to conclude much more.

Principle 12.36 | The least relaxation optimal value for parents of the active partial solutions in a minimizing branch and bound search always provides a lower bound on the optimal solution value of the full model. The greatest relaxation optimal value for parents in a maximizing search provides an upper bound.

Since every full solution that might still improve on the incumbent is a completion of some active partial solution, we can bound every such solution by finding the best of the corresponding parent bounds.

The second column of Table 12.4 tracks this overall bound on remaining active partials in our NASA application. For example, at the moment after partial solution $\mathbf{x}^{(6)}$ is explored and branched, the tree then contains active children of nodes 1, 3, 4, 5, and 6. The best of their relaxation bounds

$$\max\{798.750, 793.171, 791.250, 787.000, 770.500\} = 798.750$$

is as much as any future incumbent could ever achieve. Any unexplored solution has to be a feasible completion in one of those active partial solutions.

Suppose now that we decide to terminate the search after node 6. Table 12.4 shows that the incumbent solution value was 755 at that point, and we just computed

the best parent bound of 798.750. Thus we can compute that our approximate optimum (the incumbent) is at most

$$\frac{(\text{best possible}) - (\text{best known})}{(\text{best known})} = \frac{798.750 - 755}{755} = 5.8\%$$

below optimal. That is, we can bound the error in our approximation.

Principle 12.37 | At any stage of a branch and bound search, the difference between the incumbent solution value and the best parent bound of any active partial solution shows the maximum error in accepting the incumbent as an approximate optimum.

We use the lower bound in the denominator of percent computations because the optimal solution value could be that small.

EXAMPLE 12.21: STOPPING BRANCH AND BOUND EARLY

Return to the minimizing branch and bound tree of Example 12.20 and assume the incumbent solution value discovered at node 3 was $\hat{v} = 71$. Determine the maximum error and percent error in stopping the search at that point.

Solution: After node 3, the best parent bound [12.36](#) is

$$\min\{60, 69\} = 60$$

Thus the maximum error for this minimize model is

$$(\text{best known}) - (\text{best possible}) = 71 - 60 = 11$$

As a percent the maximum error is

$$\frac{(\text{best known}) - (\text{best possible})}{(\text{best possible})} = \frac{71 - 60}{60} = 18.3\%$$

Depth First, Best First, and Depth Forward Best Back Sequences

Another important implementation question in branch and bound is how to select a partial solution to pursue among the many that may be active. We have already introduced the simple **depth first** rule [12.23](#), which selects at each iteration an active partial solution with the most components fixed (i.e., one deepest in the search tree).

Parent bounds [12.34](#) permit other alternatives.

Definition 12.38 | **Best first** search selects at each iteration an active partial solution with best parent bound.

Definition 12.39 | **Depth forward best back** search selects a deepest active partial solution after branching a node, but one with best parent bound after a termination.

All these rules require tie-breaking refinements when several partials have maximum depth or best parent bound. One alternative builds on fractional variable rule [12.31] by preferring the nearest child.

Definition 12.40 When several active partial solutions tie for deepest or best parent bound, the **nearest child** rule chooses the one with last fixed variable value nearest the corresponding component of the parent LP relaxation.

Assuming that the parent relaxation optimum tells us something about good values for the branching variable, this nearest child is more likely to lead to early discovery of good incumbent solutions.

Figure 12.4 illustrates all 3 rules [12.23], [12.38] and [12.39] on NASA model (12.13). Branch and bound trees are shown after the first 10 partial solutions. Each uses nearest child tie-breaking rule [12.40].

Part (a) applies depth first search, always selecting an active partial solution with the most variables fixed. The deepest active partials in the current tree are the children of node 9. Nearest child rule [12.40] would choose the one with $x_4 = 1$ as $\mathbf{x}^{(10)}$ because the parent LP relaxation had $\tilde{x}_4 = 0.6$, which is closer to 1 than 0.

Notice that depth first search will automatically choose a child of the last partial solution analyzed if that partial was branched. It finds a deepest solution before branching, and the children have an additional variable fixed. This means that depth first search tends to move rapidly to fix enough variables that a feasible solution to the full model is uncovered. When rounding is not easy, that may be the best way to produce early incumbent solutions.

Selecting a child of the last node investigated can also have important computational savings. For example, at node 3 of Figure 12.4(a) the associated candidate problem differs by only the $x_{14} = 0$ constraint from that of parent node 1. Often, this similarity can be exploited to solve the relaxation at a child very quickly by starting at the LP optimum of the parent.

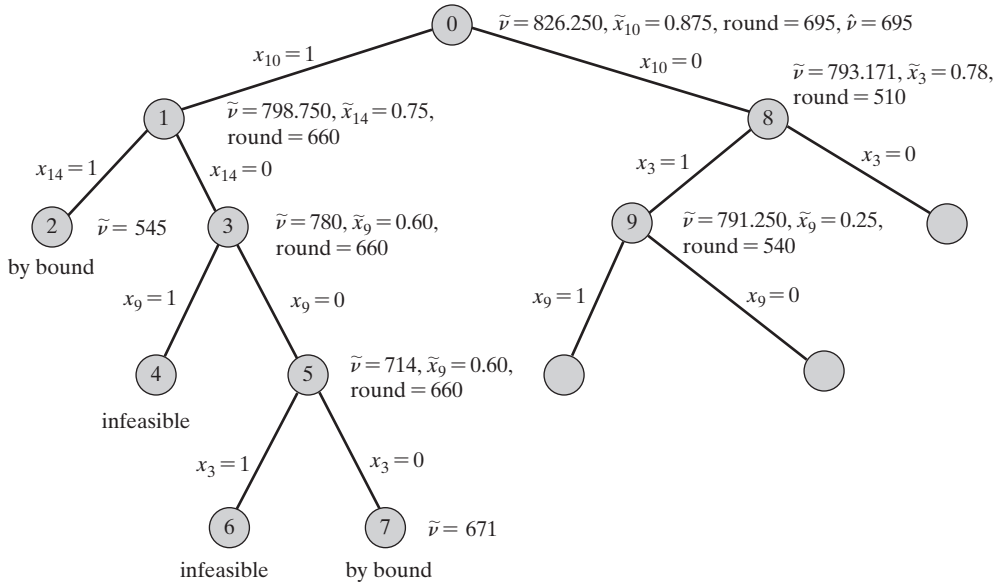
Figure 12.4(b) illustrates best first search, always advancing to an active partial solution with best parent bound. At the moment depicted there are active children of nodes 4, 7, and 9. We would next select a child of node 7 because its parent bound

$$787 = \max\{780, 787, 757\}$$

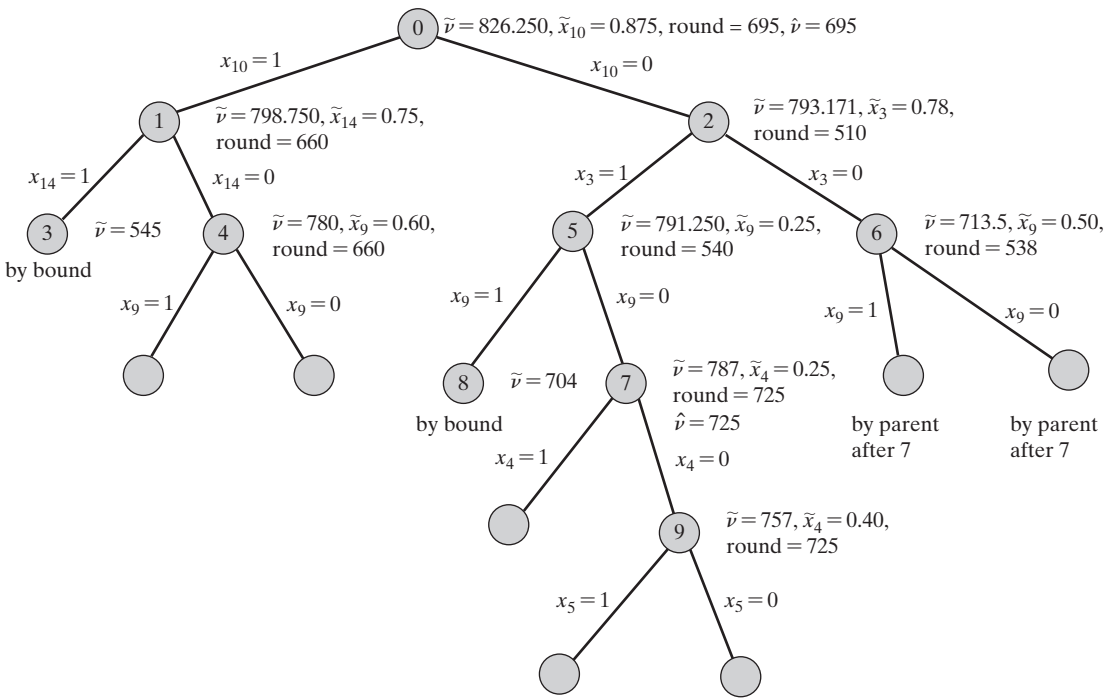
is best for this maximizing model. Partial solution $\mathbf{x}^{(10)}$ would be the child of 7 with $x_4 = 1$.

Here the idea is always to pursue a partial solution that could lead to the best possible completion. We select one with best parent bound because that is the most accurate information at hand about how good completions might be.

Notice, however, that best first search tends to skip rapidly around the branch and bound tree, with selected partial solutions often rather different than the one just before. This tendency means that depth first's efficiency when advancing to a child is often lost.

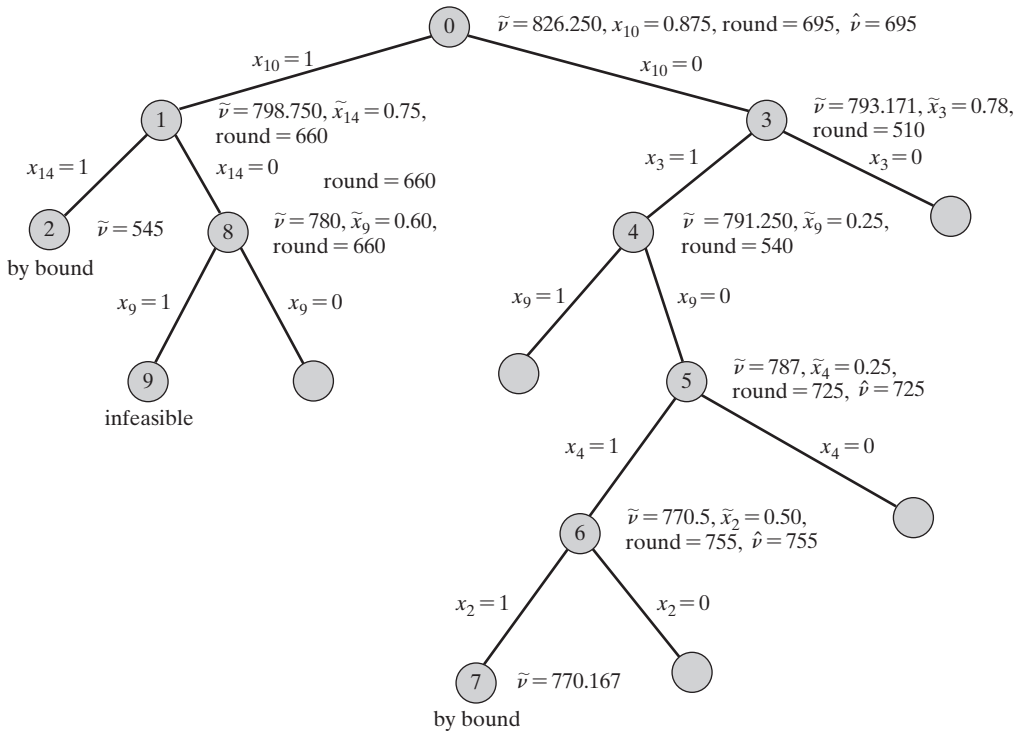


(a) Depth first



(b) Best first

FIGURE 12.4 Alternative Partial Solution Selection in NASA Application



(c) Depth forward best back

FIGURE 12.4 Alternative Partial Solution Selection in NASA Application (Continued)

The depth forward best back rule of Figure 12.4(c) (and full search tree, Figure 12.3) provides a compromise. As long as nodes are branched, so that their children are a possible next choice, this rule follows depth first in selecting one of the children. When a node is terminated, however, so that some disruption is unavoidable, rule 12.39 pursues the more ambitious best-first policy.

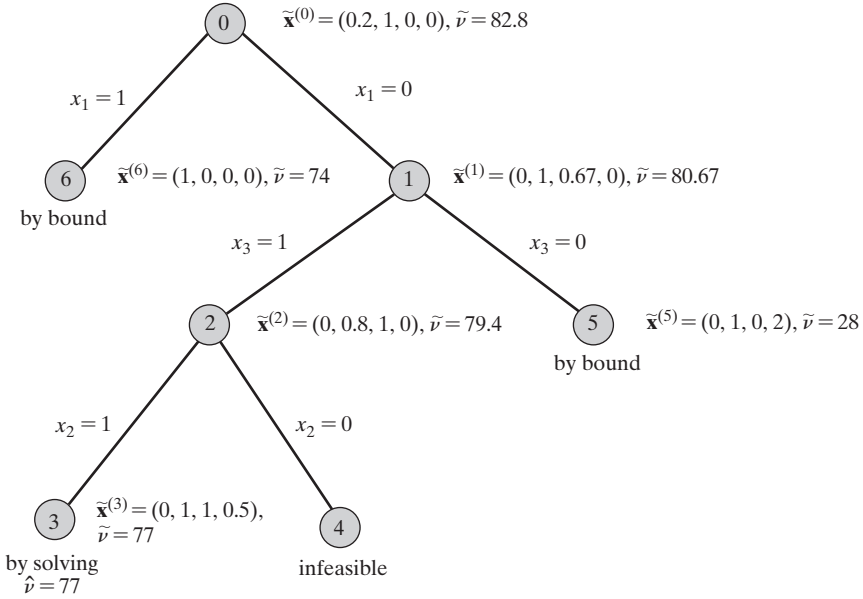
In Figure 12.4(c), partial solution $\mathbf{x}^{(9)}$ was terminated. Thus $\mathbf{x}^{(10)}$ would be an active partial with best parent bound. Here that would be the child of node 3 with $x_3 = 0$. Had node 9 been branched, $\mathbf{x}^{(10)}$ would have been one of its children.

EXAMPLE 12.22: SELECTING AMONG ACTIVE PARTIAL SOLUTIONS

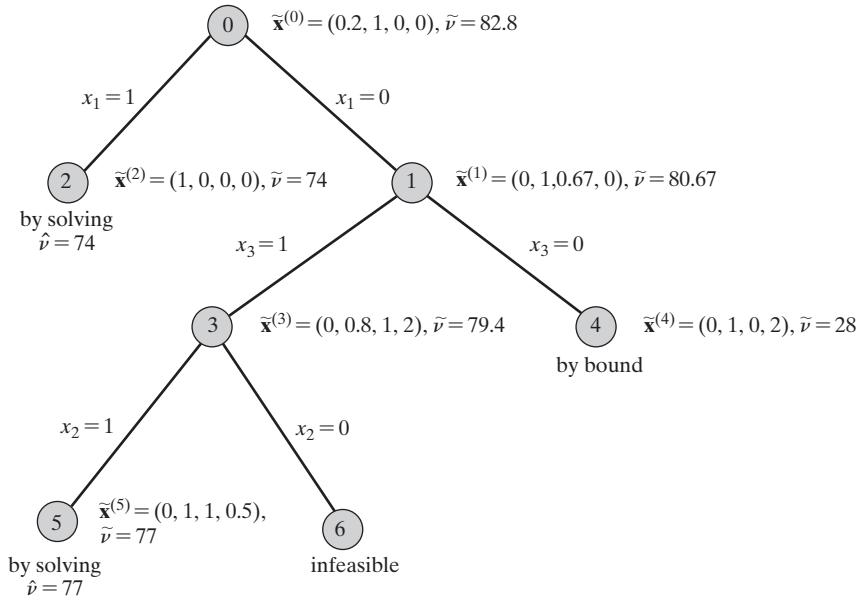
Return to the maximizing branch and bound problem of Example 12.14 and show the branch and bounds trees for searches guided by (a) depth first selection, (b) best first selection; (c) depth forward best back selection. Use nearest child tie-breaking rule 12.40 in each case.

Solution:

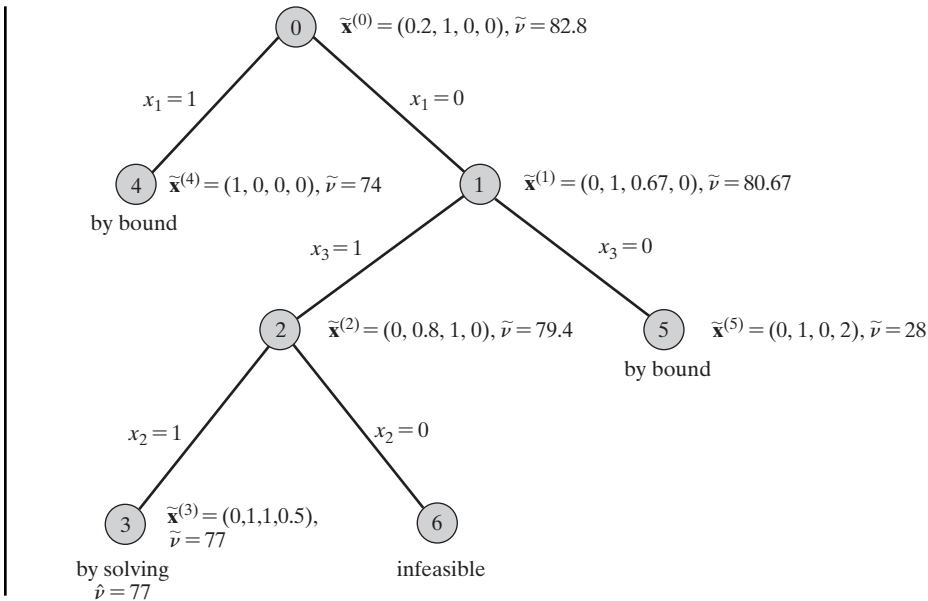
(a) The branch and bound tree for depth first rule 12.23 is



(b) The branch and bound tree for best first rule 12.38 is



(c) The branch and bound tree for depth forward best back rule 12.39 is



12.5 BRANCH AND CUT

Section 12.2 demonstrated with the Tmark facilities location model that adding or strengthening the constraints of an ILP model can sometimes dramatically improve the power of its LP relaxation. It is almost always valuable to start solution of a discrete optimization problem with a model having the strongest available relaxation. This section begins our investigation of how much more can be done as the branch and bound search proceeds by adding new constraints as needed. Subsequent sections (Sections 12.6 and 12.7) offer much more detail.

Valid Inequalities

The development begins with an understanding of what new constraints may be suitable and helpful.

Definition 12.41 A linear inequality is a **valid inequality** for a given discrete optimization model if it holds for all (integer) feasible solutions to the model.

Relaxations can often be strengthened dramatically by including valid inequalities that are not needed for a correct discrete model.

Not every valid inequality strengthens a relaxation. For example, all inequality constraints of the original formulation are trivially valid because they are satisfied by every feasible solution.

Principle 12.42 To strengthen a relaxation, a valid inequality must cut off (render infeasible) some feasible solutions to the current LP relaxation that are not feasible in the full ILP model.

This need to cut off noninteger relaxation solutions is why valid inequalities are sometimes called **cutting planes**.

EXAMPLE 12.23: RECOGNIZING USEFUL VALID INEQUALITIES

Consider the ILP

$$\begin{aligned} \max \quad & 3x_1 + 14x_2 + 18x_3 \\ \text{s.t.} \quad & 3x_1 + 5x_2 + 6x_3 \leq 10 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

with LP relaxation optimum $\tilde{\mathbf{x}} = (0, \frac{4}{5}, 1)$. Determine (by inspection) whether each of the following inequalities is valid for this model, and if so, whether adding it would strengthen the LP relaxation.

- (a) $x_2 + x_3 \leq 1$
- (b) $x_1 + x_2 + x_3 \leq 1$
- (c) $3x_1 + 5x_2 \leq 10$

Solution: We apply definition [12.41](#) and principle [12.42](#).

(a) It is obvious from the main constraint that no feasible solution can have both $x_2 = 1$ and $x_3 = 1$. Thus the constraint is valid. Also, the current LP relaxation optimum is one LP-feasible solution that violates the inequality because

$$\tilde{x}_2 + \tilde{x}_3 = \frac{4}{5} + 1 \neq 1$$

It follows that the constraint will strengthen the relaxation.

(b) This constraint is not valid. For example, $\mathbf{x} = (1, 0, 1)$ violates the constraint even though it is integer-feasible in the given model.

(c) This constraint is valid, because any integer-feasible solution satisfying main constraint $3x_1 + 5x_2 + 6x_3 \leq 10$ certainly has $3x_1 + 5x_2 \leq 10$. Still, this will also be true of all feasible solutions in the LP relaxation. Adding the inequality cannot improve the relaxation.

Branch and Cut Search

Branch and Cut algorithms integrate valid inequalities with branch and bound search dynamically as the enumeration proceeds.

Definition 12.43 | **Branch and Cut** algorithms enhance the basic branch and bound strategy of Algorithm 12A by attempting to strengthen relaxations with new valid inequalities before branching to a partial solution. Added constraints should cut off (render infeasible) the last relaxation optimum.

Branch and Cut Solution of the River Power Application

Algorithm 12B provides a formal statement of the branch and cut process. To see the idea, return to our River Power application of Section 12.3. Decisions there relate to which generators to activate, and the model is

$$\begin{aligned} \min \quad & 7x_1 + 12x_2 + 5x_3 + 14x_4 && \text{(total cost)} \\ \text{s.t.} \quad & 300x_1 + 600x_2 + 500x_3 + 1600x_4 \geq 700 && \text{(demand)} \\ & x_1, x_2, x_3, x_4 = 0 \text{ or } 1 \end{aligned} \quad (12.14)$$

Solution of the first, all-free candidate problem produces relaxation optimum

$$\tilde{\mathbf{x}}^{(0)} = (0, 0, 0, 0.438), \tilde{\nu} = 6.125$$

In normal branch and bound Figure 12.2 we immediately branched on fractional variable x_4 .

ALGORITHM 12B: BRANCH AND CUT (0-1 ILP'S)

Step 0: Initialization. Make the only active partial solution the one with all discrete variables free, and initialize solution index $t \leftarrow 0$. If any feasible solutions are known for the model, also choose the best as incumbent solution $\hat{\mathbf{x}}$ with objective value $\hat{\nu}$. Otherwise, set $\hat{\nu} \leftarrow -\infty$ if the model maximizes and $\hat{\nu} \leftarrow +\infty$ if it minimizes.

Step 1: Stopping. If active partial solutions remain, select one as $\mathbf{x}^{(t)}$, and proceed to Step 2. Otherwise, stop. If there is an incumbent solution $\hat{\mathbf{x}}$, it is optimal, and if not, the model is infeasible.

Step 2: Relaxation. Attempt to solve the linear programming relaxation of the candidate problem corresponding to $\mathbf{x}^{(t)}$.

Step 3: Termination by Infeasibility. If the LP relaxation proved infeasible, there are no feasible completions of partial solution $\mathbf{x}^{(t)}$. Terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 4: Termination by Bound. If the model maximizes and LP relaxation optimal value $\tilde{\nu}$ satisfies $\tilde{\nu} \leq \hat{\nu}$, or it minimizes and $\tilde{\nu} \geq \hat{\nu}$, the best feasible completion of partial solution $\mathbf{x}^{(t)}$ cannot improve on the incumbent. Terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 5: Termination by Solving. If the LP relaxation optimum $\tilde{\mathbf{x}}^{(t)}$ satisfies all binary constraints of the model, it provides the best feasible completion of partial solution $\mathbf{x}^{(t)}$. After saving it as new incumbent solution by $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}^{(t)}$ and $\hat{\nu} \leftarrow \tilde{\nu}$, terminate $\mathbf{x}^{(t)}$, increment $t \leftarrow t + 1$, and return to Step 1.

Step 6: Valid Inequality. Attempt to identify a valid inequality for the full ILP model that is violated by the current relaxation optimum $\tilde{\mathbf{x}}^{(t)}$. If successful, make the constraint a part of the full model increment $t \leftarrow t + 1$, and return to Step 2.

Step 7: Branching. Choose some free binary-restricted component x_p that was fractional in the last LP relaxation optimum, and branch $\mathbf{x}^{(t)}$ by creating two new actives. One is identical to $\mathbf{x}^{(t)}$ except that x_p is fixed = 0, and the other is the same except that x_p is fixed = 1. Then increment $t \leftarrow t + 1$ and return to Step 1.

Before turning the current partial solution into two in that way, branch and cut Algorithm 12B would try to improve the relaxation. The idea is to find an inequality satisfied by every binary solution to the full model but violated by $\tilde{\mathbf{x}}^{(0)}$.

Methods used to find such cutting inequalities vary enormously from one model to another. In this example we simply observe that any feasible solution in (12.14) must have a least one generator turned on. Thus

$$x_1 + x_2 + x_3 + x_4 \geq 1 \tag{12.15}$$

is valid. Also, constraint (12.15) cuts off previous relaxation optimum $\tilde{\mathbf{x}}^{(0)}$ because

$$0 + 0 + 0 + 0.438 \not\geq 1$$

Figure 12.5 shows that branch and cut advances by adding inequality (12.15) to improve the relaxation. With the same all-free partial solution, we now obtain stronger results:

$$\tilde{\mathbf{x}}^{(1)} = (0, 0, 0.818, 0.182), \quad \tilde{\nu} = 6.636$$

Suppose now that a hunt for further cuts meets with no success. The search branches as usual on fractional variable x_3 . Depth first rule [12.23] with nearest child tie-breaker [12.32] makes the next partial solution $\mathbf{x}^{(2)} = (\#, \#, 1, \#)$.

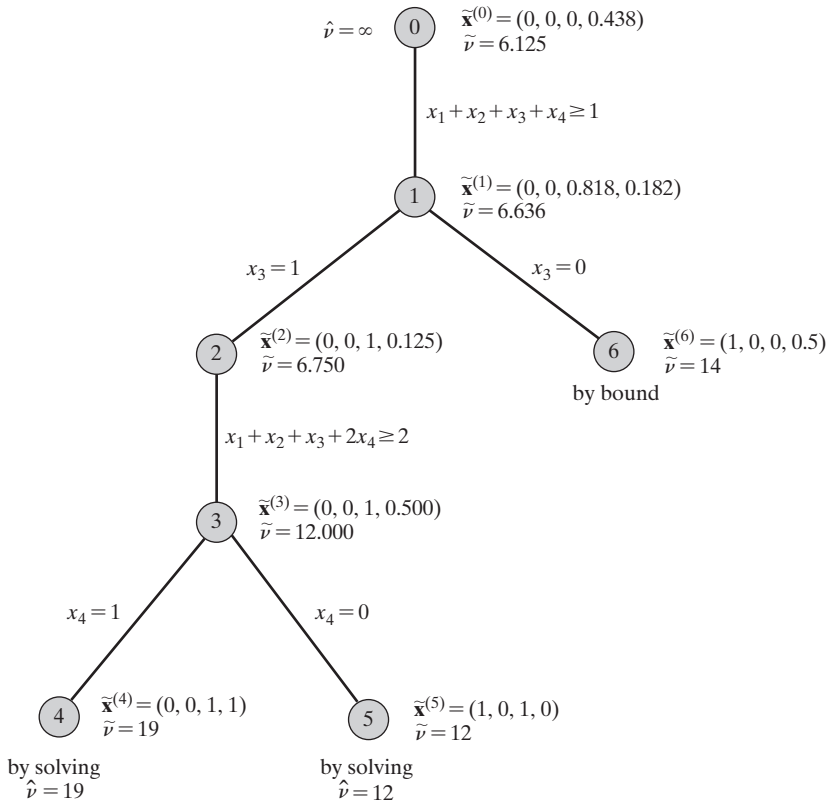


FIGURE 12.5 Branch and Cut Search of River Power Application

At node 2 the relaxation again proves inadequate for termination. This time analysis of possible cuts discovers violated inequality

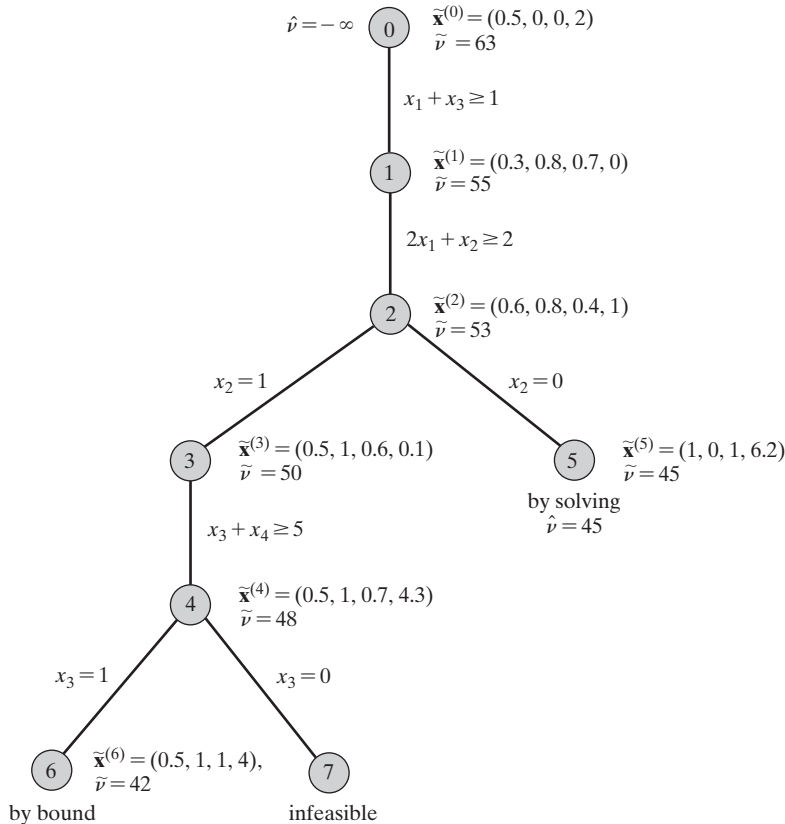
$$x_1 + x_2 + x_3 + 2x_4 \geq 2 \tag{12.16}$$

which recognizes that either generator 4 or two others are required to meet the 700-megawatt output requirement. The improved relaxation at node 3 has $\tilde{\nu} = 12$, and the search continues.

Notice that both inequalities (12.15) and (12.16) are valid for the original model (12.14). That is, they do not depend on variables fixed in partial solutions. As a consequence, they may be retained when candidate problem relaxations are solved at subsequent nodes 4 to 6. Each new cut discovered strengthens all subsequent relaxations.

EXAMPLE 12.24: UNDERSTANDING BRANCH AND CUT

The following is a branch and cut tree detailing application of Algorithm 12B to a maximizing integer linear program over $x_1, x_2, x_3 = 0$ or 1 and $x_4 \geq 0$. The best first selection rule [12.38] was employed with nearest child tie-breaker.



Assuming that all cutting inequalities shown are valid, trace the computation and justify each step.

Solution: There is no initial incumbent solution. The first relaxation at node 0 proves feasible and fractional, but cut $x_1 + x_3 \geq 1$ could be generated. We can usefully add it because

$$x_1^{(0)} + x_3^{(0)} = 0.5 + 0 = 0.5 \not\geq 1$$

The improved relaxation at node 1 still cannot be terminated, but valid inequality $2x_1 + x_2 \geq 2$ can be added because

$$2x_1^{(1)} + x_2^{(1)} = 2(0.3) + (0.8) = 1.4 \not\geq 2$$

12.6 FAMILIES OF VALID INEQUALITIES

Over the past several decades, many families of valid inequalities have been discovered that can be incorporated to speed Branch and Cut search Algorithm 12B. The section introduces some of the most familiar.

Gomory Cutting Planes (Pure Integer Case)

The systematic development of strategies for generating families of valid inequalities was pioneered by R. E. Gomory in the early days of integer programming research. Like the concepts developed in Section 12.5, the Gomory cuts are conceived as seeking a way to iteratively improve the most recent LP relaxation of the given instance by cutting off the relaxation optimum with a new valid inequality.

Gomory cuts address both pure ILPs and mixed-integer MILPs with constraints in generic standard form:

$$\begin{aligned} \sum_{j=1}^n a_{ij}x_j &= b_i, i = 1, \dots, m \\ x_j &\geq 0 \quad j = 1, \dots, n \\ x_j &\text{integer } j \in J \subseteq \{1, 2, \dots, n\} \end{aligned} \quad (12.17)$$

For simplicity, we assume all constants in these constraints are integer. When $J = \{1, 2, \dots, n\}$ the model is a pure ILP, and otherwise an MILP.

Given an optimal set of basic variables $k \in B$ for the LP relaxation of model (12.17), the corresponding **Simplex Dictionary** of its constraints (see Section 5.4 and [5.28]) can be obtained by solving main constraints for basic variables x_k in terms of the nonbasics x_j , $j \in N$:

$$\begin{aligned} x_k &= \bar{b}_k - \sum_{j \in N} \bar{a}_{kj} x_j \quad \text{for all } k \in B \\ x_j &\geq 0 \text{ for all } j \in N \quad \text{integer } j \in J \subseteq \{1, 2, \dots, n\} \end{aligned} \quad (12.18)$$

Constants \bar{b}_k and \bar{a}_{kj} are the updated forms of original data b_k and a_{kj} , respectively. As usual, the corresponding basic solution to the relaxation has $\bar{x}_k = \bar{b}_k$ for basic variables $k \in B$ and $\bar{x}_j = 0$ for nonbasics $j \in N$.

Although all original data is assumed to be integer, these updated dictionary constants may be fractional. Their **fractional parts** will be central to Gomory cut derivation.

Definition 12.44 The fractional part $\phi(q) \triangleq q - \lfloor q \rfloor$, that is, the distance down to the next lower integer value.

Some examples: $\phi(3.25) = .25$, $\phi(-3.25) = .75$, and $\phi(3) = 0$.

A first family of Gomory's cutting planes addresses the pure-integer case where all variables are required to have integer values in any feasible solution, that is, all $j \in J$.

Definition 12.45 For any row k in dictionary (12.18) of a pure-integer ILP corresponding to a \tilde{x}_k fractional, let f_{k0} denote $\phi(\bar{b}_k)$ and f_{kj} denote $\phi(\bar{a}_{kj})$. Then the classic **Gomory fractional cut** is

$$\sum_j f_{kj}x_j \geq f_{k0}$$

which can be strengthened to

$$\sum_{f_{kj} \leq f_{k0}} f_{kj}x_j + \sum_{f_{kj} > f_{k0}} \frac{f_{k0}}{1 - f_{kj}} (1 - f_{kj})x_j \geq f_{k0}$$

Before justifying these valid inequalities, consider the example of Figure 12.6(a).

Part (a) of the figure plots the feasible space of the example's LP-relaxation and demonstrates that the relaxation optimum has both variables fractional. To construct Gomory Fractional Cuts [12.45] that can progress toward an integer optimum requires that we first put the model in standard form (12.17) by adding slack variables $x_3 \geq 0$ and $x_4 \geq 0$. Note that with all model coefficients integer these slacks may also be considered integer-restricted as the difference of LHS and RHS quantities required to be integer.

It is easy to see that the optimal basis corresponding to Figure 12.6(a) is $B = \{1, 2\}$, leaving $N = \{3, 4\}$. The implied Simplex Dictionary is

$$\begin{aligned} x_1 &= 5/12 - (-5/12 x_3 + 1/12 x_4) \\ x_2 &= 11/6 - (1/6 x_3 + 1/6 x_4) \end{aligned} \quad (12.19)$$

Each of the basic variables is fractional, so two classic cuts [12.45] can be obtained:

$$\begin{aligned} \phi(-5/12)x_3 + \phi(1/12)x_4 &\geq \phi(5/12) \quad \text{and} \\ \phi(1/6)x_3 + \phi(1/6)x_4 &\geq \phi(5/6) \end{aligned}$$

Evaluating fractional parts gives

$$\begin{aligned} 7/12x_3 + 1/12x_4 &\geq 5/12 \quad \text{and} \\ 1/6x_3 + 1/6x_4 &\geq 5/6 \end{aligned}$$

The strengthened cut form of [12.45] is the same for the second of these, but the first improves to

$$\begin{aligned} (1 - \phi(-5/12))\phi((5/12)/(1 - \phi(5/12)))x_3 + \phi(1/12)x_4 &\geq \phi(5/12) \quad \text{or} \\ 25/84 x_3 + 1/12 x_4 &\geq 5/12 \end{aligned}$$

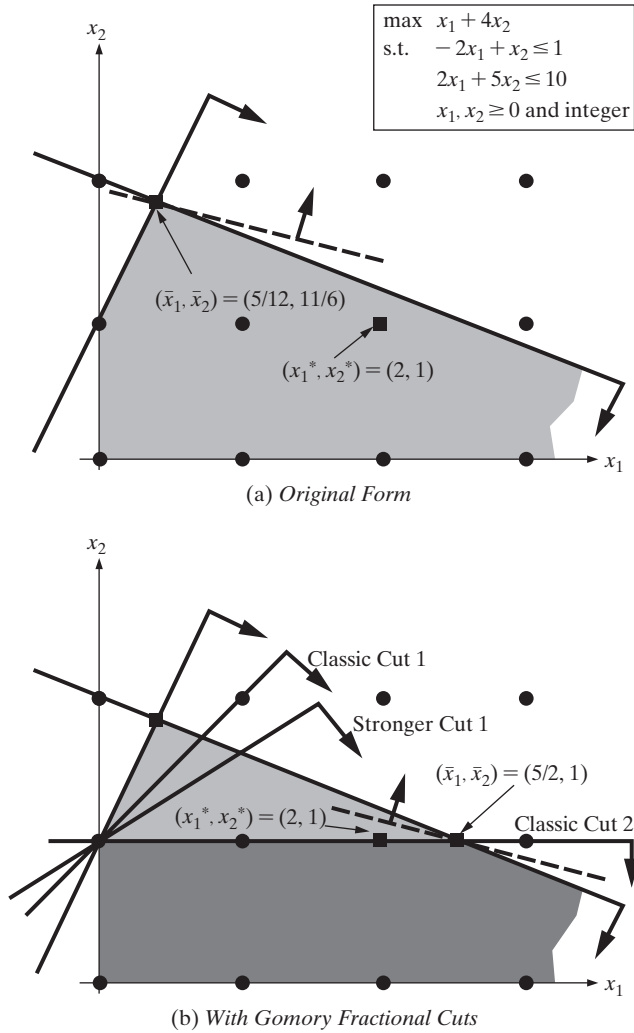


FIGURE 12.6 Numerical Example of Gomory Fractional Cuts

Then substituting for the slack variables as $x_3 = 1 + 2x_1 - x_2$ and $x_4 = 10 - 2x_1 - 5x_2$ to restate the inequalities in terms of the original variables, we obtain the three cuts depicted in Figure 12.6(b):

$$\begin{aligned} -x_1 + x_2 &\leq 1 \\ x_2 &\leq 1 \quad \text{and} \\ -3/7x_1 + 5/7x_2 &\leq 5/7 \end{aligned}$$

Notice in the figure that all 3 cuts are valid for the original ILP, and all render the original LP relaxation optimum infeasible. Still, even if all 3 cuts are added to the formulation, the updated LP feasible region (dark shading) has a fractional relaxation optimum. $(\bar{x}_1, \bar{x}_2) = (5/2, 1)$. Further computation would be required to reach an integer optimal solution.

There are two issues to justify about the cuts of [\[12.45\]](#). How can we be sure the cuts are always valid? And how do we know the strengthened form will be tighter?

Taking the second issue first, note that the only difference between the two is the coefficients of x_j with fractional part f_{kj} greater than that of the right-hand side f_{k0} . Comparison to the f_{kj} in the classic form gives (using $f_{kj} > f_0$ twice)

$$\frac{f_{k0}}{1 - f_{k0}} (1 - f_{kj}) < f_{kj} \frac{1 - f_{kj}}{1 - f_{k0}} < f_{kj}$$

We see that the changed coefficient of nonnegative x_j becomes strictly smaller in the strengthened form, meaning the cut is indeed tighter.

As to validity, note that with strengthened form tighter than the classic, both will be valid if we can establish that for the strengthened form. Any row k of the dictionary (12.18) can be rewritten

$$x_k + \sum_{f_{kj} \leq f_{k0}} (\lfloor \bar{a}_{kj} \rfloor + f_{kj}) x_j + \sum_{f_{kj} > f_{k0}} (\lceil \bar{a}_{kj} \rceil - (1 - f_{kj})) x_j = \lfloor \bar{b}_k \rfloor + f_{k0}$$

where $\lfloor q \rfloor$ is the next integer $\leq q$ and $\lceil q \rceil$ is the next integer $\geq q$. Regrouping yields

$$\begin{aligned} x_k + \sum_{f_{kj} \leq f_{k0}} \lfloor \bar{a}_{kj} \rfloor x_j + \sum_{f_{kj} > f_{k0}} \lceil \bar{a}_{kj} \rceil x_j - \lfloor \bar{b}_k \rfloor \\ = f_{k0} - \sum_{f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \end{aligned} \quad (12.19)$$

Now with all x_j restricted to be nonnegative integers, the left-hand side of (12.19) is integer, which means the right-hand side must be integer, too.

We consider two cases: either it is ≤ 0 or ≥ 1 . The first makes the right side of (12.19)

$$\sum_{f_{kj} \leq f_{k0}} f_{kj} x_j - \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \geq f_{k0} \quad (12.20)$$

and the second gives

$$- \sum_{f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \geq 1 - f_{k0}$$

which after rescaling by $f_{k0}/(1 - f_{k0})$ becomes

$$- \sum_{f_{kj} \leq f_{k0}} f_{kj} \frac{f_{k0}}{1 - f_{k0}} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) \frac{f_{k0}}{1 - f_{k0}} x_{kj} \geq f_{k0} \quad (12.21)$$

Noting all $x_j \geq 0$ and both constraints are \geq the same right-hand-side f_{k0} , we can produce a valid combination of (12.20) and (12.21) by choosing the largest of their two coefficients on each variable. When $f_{kj} \leq f_{k0}$, that will be the positive coefficient in (12.20); for j with $f_{kj} > f_{k0}$, it will come from the positive one in (12.21). The result is

$$\sum_{f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{f_{kj} > f_{k0}} \frac{f_{k0}}{1 - f_{k0}} (1 - f_{kj}) x_j \geq f_{k0}$$

which is exactly the strengthened form of definition [\[12.45\]](#), confirming its validity.

Gomory Mixed-Integer Cutting Planes

Now consider the mixed-integer (MILP) case. An extension of the stronger form in [\[12.45\]](#) can deal with that as well.

Definition 12.46 For any row k in dictionary (12.18) of a mixed-integer ILP corresponding to a $k \in J$ (integer-restricted) \bar{x}_k fractional, let f_{k0} denote $\phi(\bar{b}_k)$ and f_{kj} denote $\phi(\bar{a}_{kj})$ for $j \in J$. Then the **Gomory mixed-integer cut** is

$$\sum_{j \in J, f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{j \in J, f_{kj} > f_{k0}} \frac{f_{k0}}{1 - f_{kj}} (1 - f_{kj}) x_j + \sum_{j \notin J, \bar{a}_{kj} > 0} \bar{a}_{kj} x_j - \sum_{j \notin J, \bar{a}_{kj} < 0} \frac{f_{k0}}{1 - f_{k0}} \bar{a}_{kj} x_j \geq f_{k0}$$

The new element is retention of the \bar{a}_{kj} coefficients on continuous variables $j \notin J$.

To illustrate, consider the MILP dictionary of Table 12.5 below. All 3 basic variables are integer-restricted, but x_2 already has an integer basic value. For x_1 and x_5 , we obtain cuts [12.46](#)

$$0.3x_3 + 0.1x_4 - (0.6/0.4)2.3x_6 + 13.4x_7 \geq 0.6 \quad \text{and} \\ 0.6x_3 + 13.6x_6 - (0.4/0.6)5.9x_7 \geq 0.4$$

The argument for validity of cuts [12.46](#) closely parallels the above for the all-integer case. We start by expressing coefficients \bar{a}_{kj} with $j \in J$ in terms of their integer and fractional parts, then separating integer elements from all other to obtain the following parallel to (12.19):

$$x_k + \sum_{f_{kj} \leq f_{k0}} \lfloor \bar{a}_{kj} \rfloor x_j + \sum_{f_{kj} > f_{k0}} \lceil \bar{a}_{kj} \rceil x_j - \lfloor \bar{b}_k \rfloor \\ = f_{k0} - \sum_{f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \\ - \sum_{j \notin J, \bar{a}_{kj} > 0} \bar{a}_{kj} x_j - \sum_{j \notin J, \bar{a}_{kj} < 0} \bar{a}_{kj} x_j \tag{12.22}$$

Notice that components for continuous variables join the fractional parts on the right side of the =. For convenience at the next steps they are partitioned into ones with positive \bar{a}_{kj} vs. negative. As before, we note that the left side must be integer, so the right must be also. We consider the cases of it being ≤ 0 or ≥ 1 . The former leads to

$$\sum_{f_{kj} \leq f_{k0}} f_{kj} x_j - \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \\ + \sum_{j \notin J, \bar{a}_{kj} > 0} \bar{a}_{kj} x_j + \sum_{j \notin J, \bar{a}_{kj} < 0} \bar{a}_{kj} x_j \geq f_{k0} \tag{12.23}$$

and the latter implies

$$- \sum_{f_{kj} \leq f_{k0}} f_{kj} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) x_j \\ - \sum_{j \notin J, \bar{a}_{kj} > 0} \bar{a}_{kj} x_j - \sum_{j \notin J, \bar{a}_{kj} < 0} \bar{a}_{kj} x_j \geq 1 - f_{k0}$$

TABLE 12.5 Mixed-Integer Gomory Cut Application

x_1	=	1.6	-	(-2.7 x_3	+	1.1 x_4	+	-2.3 x_6	+	13.4 x_7)
x_2	=	3.0	-	(3.9 x_3	-	4.7 x_4	+	2.8 x_6	+	2.2 x_7
x_5	=	2.4	-	(0.6 x_3			+	13.6 x_6	-	5.9 x_7)
all x_j	$\geq 0, x_j$	integer	for	$j \in J \triangleq \{1, 2, 3, 4, 5\}$						

After rescaling by $f_{k0}/(1 - f_{k0})$, the second becomes

$$\begin{aligned} & - \sum_{f_{kj} \leq f_{k0}} f_{kj} \frac{f_{k0}}{1 - f_{k0}} x_j + \sum_{f_{kj} > f_{k0}} (1 - f_{kj}) \frac{f_{k0}}{1 - f_{k0}} x_j \\ & - \sum_{j \notin J, \bar{a}_{kj} > 0} \bar{a}_{kj} \frac{f_{k0}}{1 - f_{k0}} x_j - \sum_{j \notin J, \bar{a}_{kj} < 0} \bar{a}_{kj} \frac{f_{k0}}{1 - f_{k0}} x_j \geq f_{k0} \end{aligned} \quad (12.24)$$

Then, as before, choosing the highest (or positive) of the two coefficients on each variable in (12.23) and (12.24) produces exactly the mixed-integer cut of [12.46](#).

Families of Valid Inequalities from Specialized Models

Although complete ILP formulations may be complex, with many families of constraints, certain characteristic constraint structures recur frequently. Examples are budget constraints on availability of some resource (Section 11.3, definition [11.6](#)), and set covering/packing constraint requiring ≥ 1 or ≤ 1 option to be chosen from a set (Section 11.3, definitions [11.9](#) and [11.10](#)).

Many of the successes produced by the valid inequality theory above have involved studying the polyhedral structure of such families of constraints, then applying strong forms in more general settings.

Principle 12.47 | Strong valid inequalities for selected sets of constraints from an ILP model remain valid for the full model.

Dropping all full-model constraints not part of the selected set certainly produces a constraint relaxation (definition [12.4](#)). Since every feasible solution to the full model must be feasible in the relaxation, valid cuts for that row relaxation remain valid for all solutions to the full model.

To illustrate return to the **Binary Knapsack Problem (BKP)** form of Examples Applications 9.7 and 11.1. Main and variable-type constraints have the form

$$\begin{aligned} \sum_{j=1}^n a_j x_j &\leq b \\ x_j &\text{ binary } j = 1, \dots, n \end{aligned} \quad (12.25)$$

Coefficients a_j denote the consumption associate with each choice j , and b is the available capacity. We may assume all $a_j \leq b$ because otherwise x_j will always = 0.

Definition 12.48 | **Minimal cover** inequalities of the form

$$\sum_{j \in C} x_j \leq |C| - 1$$

are valid for Binary Knapsack Problem form (12.25) with subsets $C \subset \{1, \dots, n\}$ satisfying

$$\sum_{j \in C} a_j > b$$

$$\sum_{j \in C \setminus k} a_j \leq b \quad \text{for all } k \in C$$

That is, the inequalities are defined by members of minimally infeasible index subsets C that (i) cannot all be part of a feasible solution, but (ii) become feasible when any member of the collection is dropped. Validity of these inequalities for any instance of BKP follows directly from those two defining properties.

For a specific example, consider the following data for a BKP.

$$\begin{array}{cccccc|c}
 a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & b \\
 \hline
 2 & 3 & 3 & 5 & 7 & 11 & 15 \\
 \hline
 \end{array} \tag{12.26}$$

One minimal valid inequality [12.48] uses $C = \{1, 2, 4, 5\}$ to obtain

$$x_1 + x_2 + x_4 + x_5 \leq 3 \tag{12.27}$$

As required, the sum of the associated coefficients is $2 + 3 + 5 + 7 = 17 > 15$, and dropping even the smallest = 2 produces a total within b -limit 15.

EXAMPLE 12.25: RECOGNIZING MINIMAL COVER INEQUALITIES

Return to the BKP instance of (12.26).

Determine whether each of the following is a proper minimum cover inequality for that instance.

$$\begin{aligned}
 x_2 + x_4 + x_5 &\leq 2 \\
 x_3 + x_4 + x_6 &\leq 2 \\
 x_2 + x_3 + x_4 + x_5 &\leq 3
 \end{aligned}$$

Solution: For the first inequality, the sum of associated coefficients is $3 + 5 + 7 = 15$ which does not exceed $b = 15$. Thus it fails definition [12.48]. The second inequality has coefficient sum $3 + 5 + 11 = 19 > 15$ as required. However, dropping the smallest = 3 leaves total $5 + 11 = 16 > 15$. It too fails definition [12.48]. The coefficient sum for the third case is $3 + 3 + 5 + 7 = 18 > 15$ as required, and dropping the smallest leaves feasible total $3 + 5 + 7 = 15$. The cut is a proper minimal cover inequality.

12.7 CUTTING PLANE THEORY

To understand which inequalities (or cutting planes) like those developed in Section 12.6 are likely to be most powerful requires a theory classifying proposed cuts relative to the best possible. This section develops that **cutting plane theory**, also known as **polyhedral combinatorics** for its focus on the best possible polyhedral representation of the set of feasible (integer) solutions to a given ILP.

The Convex Hull of Integer Feasible Solutions

Definition 12.49 The **convex hull** of feasible solutions to a given integer or mixed-integer linear program is the intersection of all convex sets containing every such solution, informally the smallest such convex set.

The feasible set of any LP is always a convex set (principle [3.32](#)). Thus the convex hull is the tightest relaxation containing every integer-feasible solution to an ILP.

To see the idea, consider the following ILP.

$$\begin{aligned}
 \max \quad & -3x_1 + x_2 \\
 \text{s.t.} \quad & -x_1 + x_2 \leq 1 \\
 & 2x_1 - 2x_2 \leq 3 \\
 & 4x_1 + x_2 \geq 2 \\
 & 0 \leq x_1, x_2 \leq 2 \text{ and integer}
 \end{aligned} \tag{12.28}$$

The feasible space for this pure-integer model is depicted in Figure 12.7(a), with integer points shown as heavy dots, the LP relaxation feasible space in light shading, and the convex hull in the darker. An LP-relaxation optimum occurs at $\bar{x} = (1/5, 6/5)$, which is integer-infeasible. The unique integer optimal solution is $x^* = (1, 2)$.

To compare for a mixed-integer case, consider the MILP obtained from (12.28) by dropping integrality on x_1 , that is, requiring only x_2 to be an integer. Figure 12.7(b) graphs that case. Integer-feasible points lie along the heavy lines for $x_2 = 0, 1, \text{ or } 2$, with $x^* = (1/4, 1)$. The convex hull has now grown just enough to encompass all those solutions.

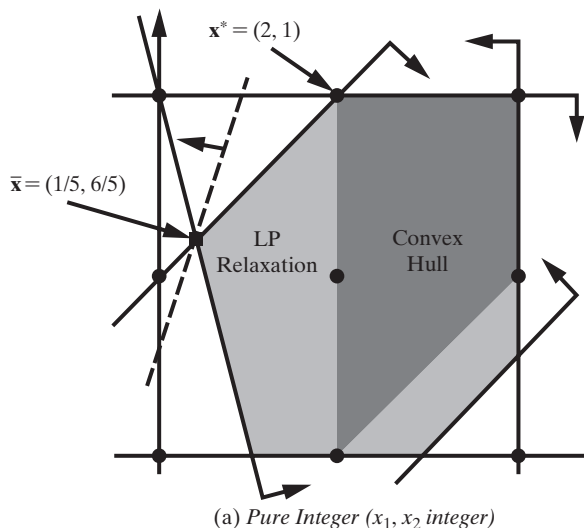


FIGURE 12.7 Convex Hull Examples

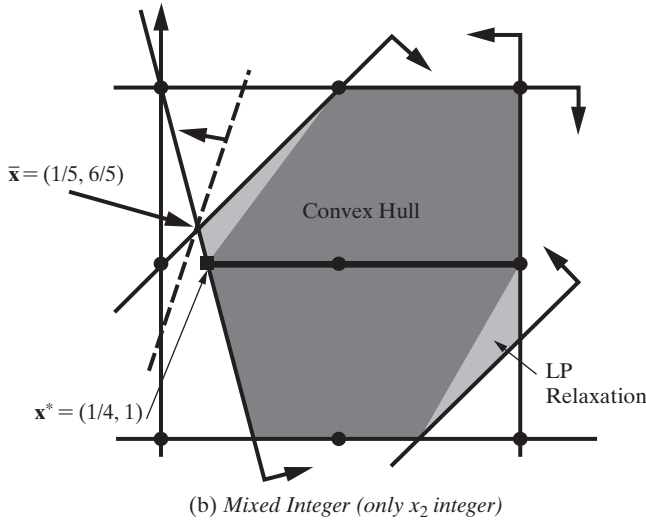


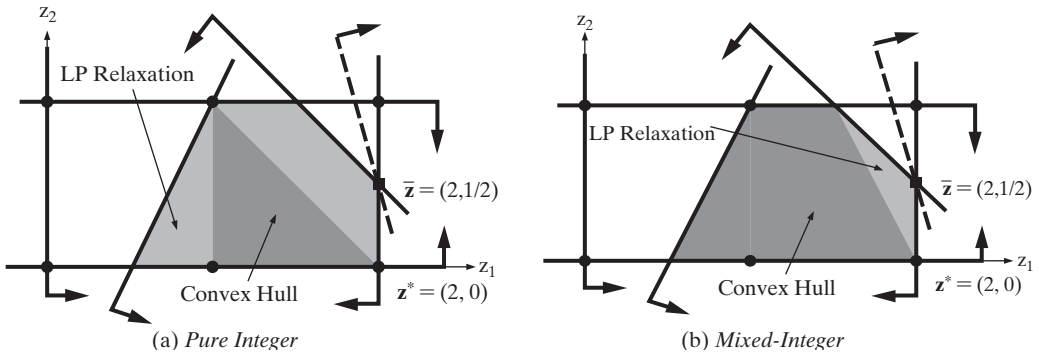
FIGURE 12.7 Convex Hull Examples (*Continued*)

EXAMPLE 12.26: IDENTIFYING CONVEX HULLS

Consider the ILP

$$\begin{aligned}
 \max \quad & 2z_1 + z_2 \\
 \text{s.t.} \quad & 2z_1 - z_2 \geq 1 \\
 & 2z_1 + 2z_2 \leq 5 \\
 & 0 \leq z_1 \leq 2 \\
 & 0 \leq z_2 \leq 1 \\
 & z_1, z_2 \text{ integer}
 \end{aligned}$$

- (a) Plot the feasible space of the model's LP-relaxation, and the convex hull of integer-feasible solutions.
- (b) Using your plot, identify optimal solutions to both the LP-relaxation and the full ILP.
- (c) Now consider a corresponding MILP where only z_2 is required to be integer, and plot the feasible space of the corresponding LP-relaxation and the convex hull of integer-feasible solutions.
- (d) Did either the LP-relaxation optimum or the integer optimum change in this revised model?

**Solution:**

- (a) Part (a) of the plots above shows by the LP-relaxation and the convex hull feasible sets.
- (b) Relaxation and integer optima are $\bar{z} = (2, 1/2)$ and $z^* = (2, 0)$, respectively.
- (c) Part (b) of the plots shows corresponding results for the mixed-integer case.
- (d) Neither optimum changes.

Linear Programs over Convex Hulls

It is rare that we can completely detail the convex hull of solutions for any integer program of practical size and complexity. Still, the convex hull serves as an ideal and a reference point for characterizing the strongest valid inequalities.

Both instances in Figure 12.7 illustrate one fundamental attraction of convex hulls.

Principle 12.50 Assuming rational coefficient data, convex hulls of solutions to integer and mixed-integer linear programs are polyhedral sets, that is, sets fully defined by linear constraints.

Proof of this important proposition is lengthy and technical, so it will be omitted here. The proposition's assumption of rational-number coefficients in the original model is required for some purely mathematical issues, but it represents no limitation for IPs of applied interest because digital computer inputs are inherently rational.

For cases of practical interest, we may conclude that convex hulls can, in principle, be fully defined as feasible sets of linear programs obtained by supplementing the equalities and inequalities of the original ILP or MILP with a sufficient collection of valid inequalities. For example, in the case of Figure 12.7(a), the representation of its convex hull can be obtained from the LP relaxation by adding valid inequalities $x_1 \geq 1$ and $x_1 - x_2 \leq 1$. In mixed-integer part (b) the needed additional inequalities are $2x_1 - x_2 \leq 3$ and $-4x_1 + 3x_2 \leq 2$.

Property 5.5 established that if any LP has a finite optimum, it has one at an extreme-point of its feasible set. The corresponding property for convex hulls of integer programs is directly analogous.

Principle 12.51 If an ILP or MILP has a finite optimum, it has one at an extreme-point of the convex hull of its integer-feasible solutions.

That is, if we knew completely a polyhedral description of the convex hull for a given integer program, we could compute an optimal integer solution by solving a linear program over the convex hull constraints. Optimal solutions \mathbf{x}^* in both parts of Figure 12.7 confirm this fact.

To see why this must be true in general, observe first that every extreme-point of a convex hull must be a fully feasible solution to the underlying ILP or MILP. Otherwise, an integer-infeasible extreme-point could be cut off by adding a new valid inequality. This would produce a strictly smaller polyhedral (and thus convex) set containing all the solutions which are feasible – a violation of definition 12.49. Furthermore, every feasible solution to the given integer model lies somewhere within its convex hull. If the model’s linear objective function produces a finite optimum over those solutions, it must produce at least one at an extreme-point of the polyhedral convex hull (principle 5.5), all of which are feasible in the given integer model.

Faces, Facets, and Categories of Valid Inequalities

Valid inequalities for a given ILP or MILP cannot cut off parts of its convex hull because it is already the tightest possible polyhedral set that contains all integer-feasible solutions. In the more typical situation where the current LP-relaxation can still be improved, we can categorize valid inequalities by how close they come to the convex hull. Figure 12.8 will provide a helpful point of reference. It depicts the convex hull of an all-integer program over $x_1, x_2, x_3 \geq 0$ and integer. The current LP-relaxation must contain this convex hull, but it presumably also includes many integer-infeasible solutions.

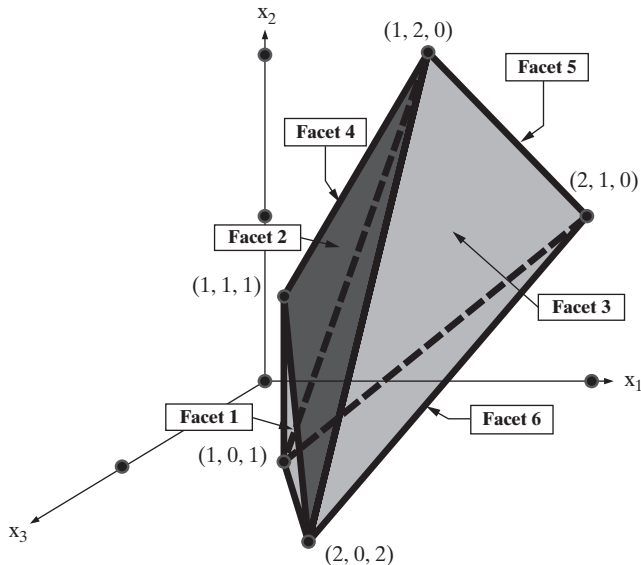


FIGURE 12.8 Faces and Facets Examples

Definition 12.52 A **face** of the convex hull is any subset satisfying some valid inequality as equality at all subset points.

The higher the dimension of a face, the more powerful the corresponding valid inequality becomes.

Definition 12.53 A **facet** is a face of maximum dimension, that is, of dimension one less than the dimension of the convex hull itself.

The polytope in Figure 12.8 is the full dimensional for its $n = 3$ decision variables. This assures no 3-dimensional subset could satisfy a valid cutting plane as equality at all its points. The highest possible dimension for this case is the $n - 1 = 2$ dimensional facets enumerated in the figure. Table 12.6 details the **facet inducing** valid inequalities that characterize each.

All of the facets in Table 12.6 are faces of the convex hull because there are valid inequalities they strictly satisfy. Indeed, they are the most preferred cutting planes because they intersect the convex hull in a maximum dimension face. Still, lower-dimensional subsets of the convex hull boundary are also faces. For example, each of the extreme-points shown in Figure 12.8 are faces of dimension = 0. Any **supporting valid inequality** at those points would satisfy definition [12.52]. Similarly, all the edges connecting pairs of those extreme-points are faces of dimension = 1. Again, a supporting valid inequality active along the whole edge would fulfill definition [12.52]. Although not as strong as the facet-inducing ones, supporting valid inequalities do at least touch the convex hull.

TABLE 12.6 Facet-Inducing Valid Inequalities of Figure 12.8

Facet Number	Inducing Valid Inequality
1	$x_1 - x_3 \geq 0$
2	$x_2 + x_3 \leq 2$
3	$2x_1 + 2x_2 + x_3 \leq 6$
4	$x_1 \geq 1$
5	$x_1 + x_2 + 2x_3 \geq 3$
6	$x_1 - 2x_2 - x_3 \leq 0$

EXAMPLE 12.27: IDENTIFYING FACE-AND FACET-INDUCING INEQUALITIES

Return to the ILPs of Example 12.26.

- Identify all the faces of the pure-integer convex hull plot (a), establish their dimensions, and determine which are facets.
- For each face in part (a) determine a face-inducing valid inequalities intersecting the convex hull in that face.

- (c) Verify that the optimal \mathbf{z}^* is an extreme-point of the convex hull, and identify the facet-defining inequalities that delineate it.
- (d) Repeat part (a) for the mixed-integer convex hull plot (b).
- (e) Repeat part (b) for the mixed-integer convex hull plot (b).
- (f) Repeat part (c) for the mixed-integer convex hull plot (b).

Solution:

(a) The three facets are the dimension 1 lines joining (1, 0) to (1, 1), (1, 0) to (2, 0), and (1, 1) to (2, 0). The three corner points (1, 0), (1, 1), and (2, 0) are dimension 0 faces, but not facets.

(b) The three facets are induced by $z_2 \geq 0$, $z_1 \geq 1$, and $z_1 + z_2 \leq 2$. Among the many supporting inequalities inducing the dimension 0 extreme-point faces are $z_1 + z_2 \geq 1$, $z_1 \leq 2$, and $z_2 \leq 1$.

(c) Integer optimal solution $\mathbf{z}^* = (2, 0)$ is indeed an extreme-point at the intersection of $z_2 \geq 0$ and $z_1 + z_2 \leq 2$.

(d) The four facets for this case are the dimension 1 lines joining (1/2, 0) to (2, 0), (1/2, 0) to (1, 1), (1, 1) to (3/2, 1), and (3/2, 1) to (2, 0). All four corner points (1/2, 0), (1, 1), (3/2, 1), and (2, 0) are dimension 0 faces, but not facets.

(e) The four facets are induced by $z_2 \geq 0$, $z_2 \leq 1$, $2z_1 - z_2 \geq 1$, and $2z_1 + z_2 \leq 4$. Among the many supporting inequalities inducing the dimension 0 extreme-point faces are $z_1 \geq 1/2$, $z_1 \leq 2$, $z_1 - z_2 \geq 0$, and $2z_1 + 2z_2 \leq 5$.

(f) Integer optimal solution $\mathbf{z}^* = (2, 0)$ is indeed an extreme-point at the intersection of $z_2 > 0$ and $2z_1 + z_2 < 5$.

Affinely Independent Characterization of Facet-Inducing Valid Inequalities

Like so many other parts of this book, graphic examples like Figure 12.8 are helpful for building insight, but more rigorous mathematical characterizations are required for instances of realistic size. The core of what is needed are independent solutions numbering 1 more than the dimension of the face being considered, that is, 1 for a point, 2 for a line, 3 for a plane, etc.

Still, care is required in defining what is meant by points being independent. First of all, they cannot all fall within a lower-dimensional set such as 3 points along a line within a plane. On the other hand, it is too much to require the defining points to be linearly independent. For example, the 3 corner points of Facet 2 in Figure 12.8 fully define the corresponding facet inducing inequality, but they are not linearly independent. $\mathbf{x}^{(2)} = (2, 0, 2)$ is a multiple of $\mathbf{x}^{(1)} = (1, 0, 1)$.

What is needed is a more refined idea of independence.

Definition 12.54 A collection of n -vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \dots, \mathbf{x}^{(k)}$ are **affinely independent** if differences $(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}), (\mathbf{x}^{(3)} - \mathbf{x}^{(1)}), \dots, (\mathbf{x}^{(k)} - \mathbf{x}^{(1)})$ from one of them are linearly independent.

Linearly independent collections of vectors are certainly affinely independent, but some linearly dependent collections like the corners of Facet 1 can also qualify after translating the origin of vectors to one of them. That is, we require only that differences $(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}) = (2, 0, 2) - (1, 0, 1) = (1, 0, 1)$ and $(\mathbf{x}^{(3)} - \mathbf{x}^{(1)}) = (1, 1, 1) - (1, 0, 1) = (0, 1, 0)$ be linearly independent, which they clearly are.

This brings us the characterization required.

Principle 12.55 | A valid inequality for a given ILP or MILP induces a face of dimension k in the corresponding convex hull if and only if there exist $k + 1$ affinely independent, integer-feasible solutions to the model satisfying the inequality as equality. In particular, a valid inequality is facet-inducing if and only if there exist n affinely independent solutions satisfying it as equality, where n is the dimension of the convex hull.

EXAMPLE 12.28: IDENTIFYING FACE-AND FACET-INDUCING INEQUALITIES

Return to the ILPs of Example 12.26.

- (a) Exhibit the required number of affinely independent feasible points satisfying each facet as equality and demonstrate their affine independence for the pure-integer convex hull plot (a).
- (b) Repeat part (a) for the mixed-integer case of plot (b).

Solution:

(a) For each of the (dimension 1) facets we need two affinely independent points. For $z_2 \geq 0$, $(1, 0)$ and $(2, 0)$ will serve, but they are not linearly independent. Still the difference $(2, 0) - (1, 0) = (1, 0)$ is linearly independent as required by principle [12.55](#). For $z_1 \geq 1$, linearly independent $(1, 0)$ and $(1, 2)$ suffice. For $z_1 + z_2 \leq 2$, linearly independent $(2, 0)$ and $(1, 1)$ fulfill the requirements.

(b) Again we require 2 affinely independent points for each of the (dimension 1) facets. For $z_2 \geq 0$, $(1, 0)$ and $(2, 0)$ suffice even though they are not linearly independent. Many other pairs would serve as well. For $z_2 \leq 1$, linearly independent $(1, 1)$ and $(3/2, 1)$ meet the requirement. For $2z_1 - z_2 \geq 1$, linearly independent $(1/2, 0)$ and $(1, 1)$ suffice. For $2z_1 + z_2 \leq 4$, linearly independent $(3/2, 1)$ and $(2, 0)$ satisfy the need.

Partial Dimensional Convex Hulls and Valid Equalities

For simplicity so far we have used only full-dimensional examples of convex hulls, but many cases do not meet this standard. To explore such possibilities, more rigorous definitions are needed.

Definition 12.56 | The **dimension** of any polyhedral set of n -vectors is 1 less than the maximum number of affinely independent points belonging to the set. The set is **full dimensional** if its dimension = n , and **partial dimensional** otherwise ($< n$).

We can use any of the facets in Figure 12.8 plus one separate point to establish that it is formally full dimensional. From Facet 1 using

$$\begin{aligned} \mathbf{x}^{(1)} &= (1, 0, 1) \\ \mathbf{x}^{(2)} &= (1, 1, 1) \quad \mathbf{x}^{(2)} - \mathbf{x}^{(1)} = (0, 1, 0) \\ \mathbf{x}^{(3)} &= (2, 2, 1) \quad \mathbf{x}^{(3)} - \mathbf{x}^{(1)} = (1, 2, 0) \\ \mathbf{x}^{(4)} &= (2, 0, 2) \quad \mathbf{x}^{(4)} - \mathbf{x}^{(1)} = (1, 0, 1) \end{aligned}$$

Differences for $\mathbf{x}^{(1)}$ are linearly independent, proving the 4 points are affinely independent (definition 12.54), and the dimension of the set = 4 - 1 or 3. This is full dimension for 3-vectors.

To see a partial dimensional case, consider the convex hull in Figure 12.9. Even though it is contained in 3-space, its 3 extreme-points form a maximum set of affinely independent solutions, so that dimension = 3 - 1 = 2 (definition 12.56). Thus (principle 12.55) the convex hull's three 1-dimensional edges connecting pairs of extreme-points are its facets. Any supporting valid inequality intersecting one of those edges is facet-inducing, such as $x_3 \leq 1$ for the edge from (0, 1, 1) to (1, 0, 1).

Still, those facets, or corresponding facet-inducing inequalities, do not fully describe the polyhedral set in Figure 12.9. We clearly need the **valid equality** $x_1 + x_2 + x_3 = 2$ containing the whole set.

Definition 12.57 A valid equality of a given polyhedral set is a linear equality satisfied by every member of the set.

Existence of such valid equalities is what makes a polyhedral set partial-dimensional. Often they are apparent in the original LP-relaxation of an ILP or MILP, so their importance comes less from being discovered to describe the convex hull than in determining its dimension, and thus the dimension of facet-inducing valid inequalities we do need to identify.

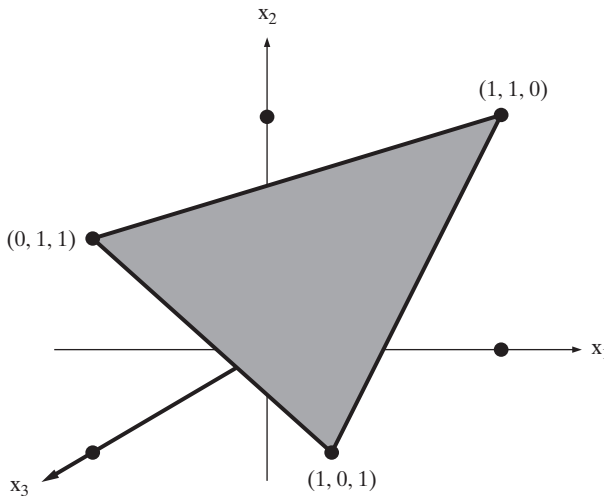


FIGURE 12.9 Example of a Partial Dimensional Convex Hull

EXAMPLE 12.29: RECOGNIZING PARTIAL DIMENSION AND VALID EQUALITIES

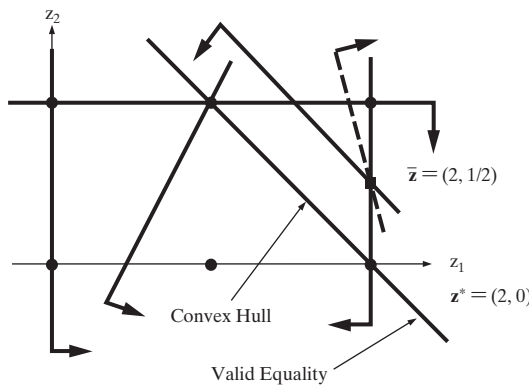
Return to the two IPs of Examples 12.26.

- (a) Establish that both convex hulls are full dimensional.
- (b) Suppose now that the original LP-relaxation constraints had included the valid equality $z_1 + z_2 = 2$. Demonstrate that the revised LP-relaxation in both cases, and also the new convex hull, would be just the line segment from $\mathbf{z} = (1, 1)$ to $(2, 0)$.
- (c) Establish that this revised convex hull is partial dimensional.

Solution:

(a) With two decision variables in each case, full dimension would be $n = 2$. In both parts (a) and (b) solutions $\mathbf{z} = (1, 0)$, $(2, 0)$, and $(1,1)$ provide the 3 affinely independent solutions required to establish dimension 2.

(b) Adding the valid equality reduces both cases to the plot below.



The new convex hull is simply the line segment from $\mathbf{z} = (1, 1)$ to $(2, 0)$.

(c) The same two solutions from part (b) provide the affinely independent points needed to establish that the dimension of the new convex hull is $2 - 1 = 1$. With two decision variables, this demonstrates the new convex hull is partial dimension $1 < 2$.

EXERCISES

12-1 Solve each of the following discrete optimization models by total enumeration.

✓ (a) $\min \quad 2x_1 + x_2 + 4x_3 + 10x_4$
 s.t. $x_1 + x_2 + x_3 \leq 2$
 $3x_1 + 7x_2 + 19x_3 + x_4 \geq 20$
 $x_1, x_2, x_3 = 0 \text{ or } 1$
 $x_4 \geq 0$

(b) $\max \quad 30x_1 + 12x_2 + 24x_3 + 55x_4$
 s.t. $30x_1 + 20x_2 + 40x_3 + 35x_4 \leq 60$
 $x_2 + 2x_3 + x_4 \geq 2$
 $x_1 \geq 0$
 $x_2, x_3, x_4 = 0 \text{ or } 1$

12-2 Suppose that you have been asked to solve a mixed-integer ILP with 10,000 continuous and n

binary decision variables. Determine the largest n 's for which the problem could be totally enumerated in one 24-hour day and in one 24-hour, 30-day month by each of the following computer environments.

- ✓ (a) An engineering workstation that can enumerate one choice of binary variables each second, including solving the resulting LP.
 (b) A parallel processing computer that can evaluate 8192 choices simultaneously every second, including solving the resulting LPs.

12-3 Consider the ILP

$$\begin{aligned} \max \quad & 14x_1 + 2x_2 - 11x_3 + 17x_4 \\ \text{s.t.} \quad & 2x_1 + x_2 + 4x_3 + 5x_4 \leq 12 \\ & x_1 - 3x_2 - 3x_3 - 3x_4 \leq 0 \\ & x_1 \geq 0 \\ & x_2, x_3, x_4 = 0 \text{ or } 1 \end{aligned}$$

Determine whether each of the following is a constraint relaxation.

- ✓ (a) $\max \quad 14x_1 + 2x_2 - 11x_3 + 17x_4$
 $\text{s.t.} \quad 2x_1 + x_2 + 4x_3 + 5x_4 \leq 12$
 $x_1 - 3x_2 - 3x_3 - 3x_4 \leq 0$
 $x_j \geq 0, \quad j = 1, \dots, 4$
- (b) $\max \quad 14x_1 + 2x_2 - 11x_3 + 17x_4$
 $\text{s.t.} \quad 2x_1 + x_2 + 4x_3 + 5x_4 \leq 12$
 $x_1 - 3x_2 - 3x_3 - 3x_4 \leq 0$
 $x_1, x_2, x_3, x_4 = 0 \text{ or } 1$
- ✓ (c) $\max \quad 14x_1 + 2x_2 - 11x_3 + 17x_4$
 $\text{s.t.} \quad 2x_1 + x_2 + 4x_3 + 5x_4 \leq 5$
 $x_1 - 3x_2 - 3x_3 - 3x_4 \leq 0$
 $x_1 \geq 0$
 $x_2, x_3, x_4 = 0 \text{ or } 1$
- (d) $\max \quad 14x_1 + 2x_2 - 11x_3 + 17x_4$
 $\text{s.t.} \quad x_1 - 3x_2 - 3x_3 - 3x_4 \leq 10$
 $x_1 \geq 0$
 $x_2, x_3, x_4 = 0 \text{ or } 1$

12-4 Form the linear programming relaxation of each of the following ILPs.

- ✓ (a) $\min \quad 12x_1 + 45x_2 + 67x_3 + 1x_4$
 $\text{s.t.} \quad 4x_1 + 2x_2 - x_4 \leq 10$
 $6x_1 + 19x_3 \geq 5$
 $x_2, x_3, x_4 \geq 0$
 $x_1 = 0 \text{ or } 1$
 x_3 integer

- (b) $\max \quad 3x_1 + 8x_2 + 9x_3 + 4x_4$
 $\text{s.t.} \quad 2x_1 + 2x_2 + 2x_3 + 3x_4 \leq 20$
 $29x_1 + 14x_2 + 78x_3 + 20x_4 \leq 100$
 $x_1, x_2, x_3 = 0 \text{ or } 1$
 $x_4 \geq 0$

12-5 Each of the following ILPs has no feasible solutions. Solve the corresponding LP relaxation graphically and indicate whether your relaxation results are sufficient to show that the ILP is infeasible.

- ✓ (a) $\min \quad 10x_1 + 15x_2$
 $\text{s.t.} \quad x_1 + x_2 \geq 2$
 $-2x_1 + 2x_2 \geq 1$
 $x_1, x_2 = 0 \text{ or } 1$
- (b) $\max \quad 40x_1 + 17x_2$
 $\text{s.t.} \quad 2x_1 + x_2 \geq 2$
 $2x_1 - x_2 \leq 0$
 $x_1, x_2 = 0 \text{ or } 1$
- ✓ (c) $\min \quad 2x_1 + x_2$
 $\text{s.t.} \quad x_1 + 4x_2 \leq 2$
 $-4x_1 + 4x_2 \geq 1$
 $x_1 \geq 0, x_2 = 0 \text{ or } 1$
- (d) $\max \quad 57x_1 + 20x_2$
 $\text{s.t.} \quad x_1 + x_2 \geq 4$
 $x_1 = 0 \text{ or } 1$
 $0 \leq x_2 \leq 2$

12-6 Determine the best bound on the optimal solution value of an ILP with each of the following objective functions that is available from the specified LP relaxation optima $\tilde{\mathbf{x}}$.

- ✓ (a) $\max \quad 24x_1 + 13x_2 + 3x_3$
 $\tilde{\mathbf{x}} = \left(2, \frac{1}{2}, 0\right)$
- (b) $\min \quad x_1 - 6x_2 + 49x_3$
 $\tilde{\mathbf{x}} = \left(1, 0, \frac{2}{7}\right)$
- ✓ (c) $\min \quad 60x_1 - 16x_2 + 10x_3$
 $\tilde{\mathbf{x}} = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$
- (d) $\max \quad 90x_1 + 11x_2 + 30x_3$
 $\tilde{\mathbf{x}} = \left(0, \frac{1}{2}, 3\right)$

12-7 Determine whether each of the following LP relaxation optima $\tilde{\mathbf{x}}$ is optimal in the corresponding ILP over the specified variable type constraints.

- ✓ (a) $x_j = 0$ or $1, j = 1, \dots, 4$

$$\tilde{\mathbf{x}} = \left(1, 0, \frac{1}{3}, \frac{2}{3}\right)$$

- (b) $x_1, x_2 = 0$ or $1, x_3, x_4 \geq 0$

$$\tilde{\mathbf{x}} = \left(0, 1, \frac{3}{2}, \frac{1}{2}\right)$$

- ✓ (c) $x_1, x_2, x_3 = 0$ or $1, x_4 \geq 0$

$$\tilde{\mathbf{x}} = \left(1, 0, 1, \frac{23}{7}\right)$$

- (d) $x_j \geq 0$ and integer, $j = 1, \dots, 4$

$$\tilde{\mathbf{x}} = \left(0, 3, \frac{3}{2}, 1\right)$$

12-8 The ILP

$$\begin{aligned} \max \quad & 3x_1 + 6x_2 + 4x_3 + 10x_4 + 3x_5 \\ \text{s.t.} \quad & 2x_1 + 4x_2 + x_3 + 3x_4 + 7x_5 \leq 10 \\ & x_1 + x_3 + x_4 \leq 2 \\ & 4x_2 + 4x_4 + 4x_5 \leq 7 \\ & x_1, \dots, x_5 = 0 \text{ or } 1 \end{aligned}$$

has LP relaxation optimal solution $\tilde{\mathbf{x}} = (0, 0.75, 1, 1, 0)$.

- ✓ (a) Determine the best bound on the ILP optimal solution value available from relaxation results.
- ✓ (b) Determine whether the relaxation optimum solves the full ILP. If not, round to an ILP-feasible solution either by moving all binary variables at fractional values in the relaxation up to 1 or by moving all down to 0.
- ✓ (c) Combine parts (a) and (b) to determine the best upper and lower bounds on the ILP optimal solution value available from the combination of relaxation and rounding.
- ✓ (d) Verify your bounds of part (c) by solving the full ILP with class optimization software.

12-9 Do Exercise 12-8 for the ILP

$$\begin{aligned} \min \quad & 12x_1 + 5x_2 + 4x_3 + 6x_4 + 7x_5 \\ \text{s.t.} \quad & 6x_1 + 8x_2 + 21x_3 + 6x_4 + 5x_5 \geq 11 \\ & x_1 + x_2 + 2x_3 + x_4 \geq 1 \\ & 2x_2 + 5x_3 + x_5 \geq 2 \\ & x_1, \dots, x_5 = 0 \text{ or } 1 \end{aligned}$$

and LP relaxation optimum $\tilde{\mathbf{x}} = (0, 0, 0.524, 0, 0)$.

12-10 Do Exercise 12-8 for the ILP

$$\begin{aligned} \min \quad & 17x_1 + 12x_2 + 24x_3 + 2x_4 + 8x_5 \\ \text{s.t.} \quad & 3x_1 + 5x_3 + 7x_4 + 9x_5 \geq 13 \\ & 7x_2 + 4x_4 + 11x_5 \geq 5 \\ & 2x_1 + 3x_2 + 2x_3 + 3x_4 \geq 7 \\ & x_2, x_3, x_4 = 0 \text{ or } 1 \\ & x_1, x_5 \geq 0 \end{aligned}$$

and LP relaxation optimum $\tilde{\mathbf{x}} = (0.5, 1, 0, 1, 0.5)$.

12-11 Do Exercise 12-8 for the ILP

$$\begin{aligned} \min \quad & 50x_1 + 25x_2 + 100x_3 + 300x_4 \\ & \quad \quad \quad + 200x_5 + 500x_6 \\ \text{s.t.} \quad & 10x_1 + 6x_2 + 2x_3 = 45 \\ & 2x_1 + 3x_2 + x_3 \geq 12 \\ & 0 \leq x_1 \leq 5x_4 \\ & 0 \leq x_2 \leq 5x_5 \\ & 0 \leq x_3 \leq 5x_6 \\ & x_4, x_5, x_6 = 0 \text{ or } 1 \end{aligned}$$

and LP relaxation optimum $\tilde{\mathbf{x}} = (1.5, 5, 0, 0.3, 1, 0)$.

12-12 Consider the ILP

$$\begin{aligned} \min \quad & 10x_1 + 20x_2 + 40x_3 + 80x_4 - 144y \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \geq 4y \\ & x_1, \dots, x_4, y = 0 \text{ or } 1 \end{aligned}$$

- ✓ (a) Solve the full ILP model by inspection.
- (b) Verify by inspection that its LP relaxation has optimal solution $\tilde{\mathbf{x}} = (1, 1, 0, 0)$, $\tilde{y} = \frac{1}{2}$.
- ✓ (c) Show that an equivalent ILP would result if the main constraint were replaced by $x_j \geq y \quad j = 1, \dots, 4$
- ✓ (d) Verify that the revised formulation of part (c) has a stronger LP relaxation than the original of part (b).

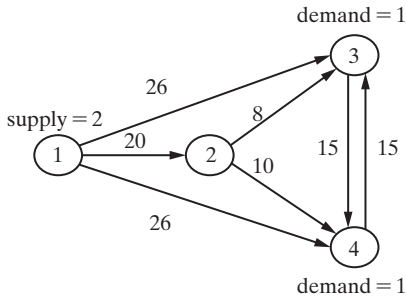
12-13 Do Exercise 12-12 for ILP

$$\begin{aligned} \min \quad & 16x_1 + 14x_2 + 15x_3 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \\ & x_2 + x_3 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_1, \dots, x_3 = 0 \text{ or } 1 \end{aligned}$$

LP relaxation optimum $\tilde{\mathbf{x}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ and revised main constraint

$$x_1 + x_2 + x_3 \geq 2$$

12-14 Consider the Fixed Charge Network Flow instance displayed below (see Section 11.6).



Numbers on arcs are fixed charges f_{ij} . All variable costs $c_{ij} = 0$.

- (a) Formulate the instance as in 11.31 using decision variables $x_{ij} \triangleq$ flow on arc (i, j) , and $y_{ij} = 1$ if $x_{i,j} > 0$ and $= 0$ otherwise.
- (b) Identify by inspection an optimal solution to the full ILP model of (a).
- (c) Identify by inspection an optimal solution to the LP relaxation of the model in (a).
- (d) Now Develop an extended formulation of the instance which artificially divides flows into two separate flow networks. The first in decision variables $x_{ij}^{(1)}$ should define constraints for flows from source 1 to first demand 3. The other in $x_{ij}^{(2)}$ should define constraints on flows from source 1 to second demand 4. A common set of y -variables for the fixed costs should be subject to pairs of switching constraints for every arc (i, j) as $x_{ij}^{(1)} \leq (\text{demand at 3})y_{ij}$, and $x_{ij}^{(2)} \leq (\text{demand at 4})y_{ij}$
- (e) Explain why the new formulation of (d) is a correct representation of the given fixed-charge instance.
- (f) Use class optimization software to solve the LP relaxation of the new version in (d). Then compare with results in (b) and (c), and comment.

12-15 The fixed-charge ILP

$$\begin{aligned} \min \quad & 60x_1 + 78x_2 + 200y_1 + 400y_2 \\ \text{s.t.} \quad & 12x_1 + 20x_2 \geq 64 \\ & 15x_1 + 10x_2 \leq 60 \\ & x_1 + x_2 \leq 10 \end{aligned}$$

$$\begin{aligned} 0 \leq x_1 &\leq 100y_1 \\ 0 \leq x_2 &\leq 100y_2 \\ y_1, y_2 &= 0 \text{ or } 1 \end{aligned}$$

has LP relaxation optimum $\tilde{\mathbf{x}} = (0, 3.2)$, $\tilde{\mathbf{y}} = (0, 0.032)$.

- ✓ (a) Compute the smallest replacements for big- M values of 100 in this formulation that can be inferred simply by examining constraints of the model.
- ✓ (b) Show that the LP relaxation optimum will change if the lower big- M 's of part (a) are employed.
- ✓ (c) Verify part (b) by solving the model having smaller big- M 's with class optimization software.

12-16 Do Exercise 12-15 for the average-completion-time, single-machine scheduling ILP

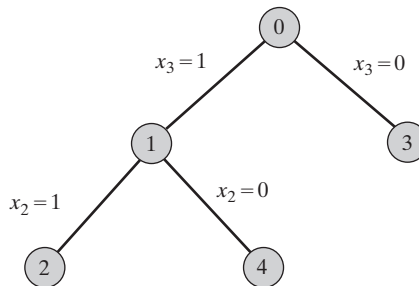
$$\begin{aligned} \min \quad & 0.5(x_1 + 12 + x_2 + 8) \\ \text{s.t.} \quad & x_1 + 12 \leq x_2 + 75(1 - y) \\ & x_2 + 8 \leq x_1 + 75y \\ & x_1, x_2 \geq 0 \\ & y = 0 \text{ or } 1 \end{aligned}$$

with LP relaxation optimum $\tilde{\mathbf{x}} = (0, 0)$, $\tilde{y} = 0.08$. [*Hint:* Consider the sum of the process times in part (a).]

12-17 Suppose that an integer linear program has decision variables $x_1, x_2, x_3 = 0$ or 1. List all completions of the following partial solutions.

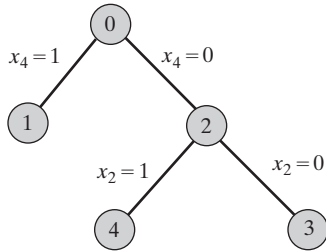
- ✓ (a) $(\#, 0, \#)$
- (b) $(1, 0, \#)$

12-18 The following is the complete branch and bound tree for an ILP over decision variables $x_1, \dots, x_4 = 0$ or 1.



- ✔ (a) List the partial solutions associated with each node of the tree.
- ✔ (b) Which nodes were branched and which terminated?
- ✔ (c) Identify the nodes of the tree that have $\mathbf{x} = (0, 1, 0, 1)$ as a feasible completion.

12-19 Do Exercise 12-18 for the branch and bound tree



12-20 Suppose that the ILP of Exercise 12-8 is being solved by branch and bound. State the candidate problem associated with each of the following partial solutions.

- ✔ (a) $(\#, 1, \#, \#, 0)$
- (b) $(\#, 1, 0, 1, \#)$

12-21 Suppose that a minimizing ILP is being solved by LP-based branch and bound Algorithm 12A over decision variables $x_1, x_2, x_3 = 0$ or 1 , $x_4 \geq 0$. Show how the search should process the node with $x_2 = 1$ and other variables free if the corresponding LP relaxation has each of the following outcomes. Assume that the incumbent solution value is 100.

- ✔ (a) $\tilde{\mathbf{x}} = (0.9, 1, 0, 6)$, value $\tilde{\nu} = 97$
- (b) $\tilde{\mathbf{x}} = (0.2, 1, 0.77, 4.5)$, value $\tilde{\nu} = 102$
- ✔ (c) $\tilde{\mathbf{x}} = (1, 1, 0, 4.2)$, value $\tilde{\nu} = 75$
- (d) LP relaxation infeasible
- ✔ (e) $\tilde{\mathbf{x}} = (1, 1, 0.6, 0)$, value $\tilde{\nu} = 100$
- (f) $\tilde{\mathbf{x}} = (0.4, 1, 0.1, 5.9)$, value $\tilde{\nu} = 86$

✔ **12-22** The following table shows the LP relaxation outcomes for all possible combinations of fixed and free variables in branch and bound solution of a minimizing integer linear program

over decision variables $x_1, x_2, x_3 = 0$ or 1 , $x_4 \geq 0$. Solve the problem by LP-based Algorithm 12A and record your results in a branch and bound tree. Apply the depth first rule for selecting among active nodes and pick whichever of $= 0$ and $= 1$ is closest to the preceding relaxation value when nodes have equal depth. Branch on the integer-restricted variable with fractional relaxation value nearest to integer.

x_1	x_2	x_3	$\tilde{\mathbf{x}}$	$\tilde{\nu}$
#	#	#	(0, 0.60, 0.14, 0)	60.9
#	#	0	(0.20, 0.60, 0, 0)	61.0
#	#	1	(0.60, 0, 1, 0)	69.0
#	0	#	(0.60, 0, 1, 0)	69.0
#	0	0	Infeasible	—
#	0	1	(0.60, 0, 1, 0)	69.0
#	1	#	(0, 1, 0, 400)	4090.0
#	1	0	(0, 1, 0, 400)	4090.0
#	1	1	Infeasible	—
0	#	#	(0, 0.60, 0.14, 0)	60.9
0	#	0	(0, 0.60, 0, 1.9)	73.6
0	#	1	(0, 0, 1, 6)	108.0
0	0	#	(0, 0, 1, 6)	108.0
0	0	0	Infeasible	—
0	0	1	(0, 0, 1, 6)	108.0
0	1	#	(0, 1, 0, 400)	4090.0
0	1	0	(0, 1, 0, 400)	4090.0
0	1	1	Infeasible	—
1	#	#	(1, 0.33, 0, 0)	65.0
1	#	0	(1, 0.33, 0, 0)	65.0
1	#	1	(1, 0, 1, 0)	83.0
1	0	#	(1, 0, 0.71, 0)	69.3
1	0	0	Infeasible	—
1	0	1	(1, 0, 1, 0)	83.0
1	1	#	(1, 1, 0, 400)	4125.0
1	1	0	(1, 1, 0, 400)	4125.0
1	1	1	Infeasible	—

12-23 Do Exercise 12-22 for a minimizing mixed-integer linear program over binary variables w_1, w_2, w_3 , and $w_4 \geq 0$.

w_1	w_2	w_3	LP Optimum	LP Value
#	#	#	(0.2, 0.0, 0.0, 0.0)	24.6
#	#	0	(0.2, 0.0, 0.0, 0.0)	24.6
#	#	1	(0.0, 0.0, 1.0, 0.0)	83.0
#	0	#	(0.2, 0.0, 0.0, 0.0)	24.6
#	0	0	(0.2, 0.0, 0.0, 0.0)	24.6
#	0	1	(0.0, 0.0, 1.0, 0.0)	83.0
#	1	#	(0.06, 1.0, 0.0, 0.0)	58.4
#	1	0	(0.06, 1.0, 0.0, 0.0)	58.4
#	1	1	(0.0, 1.0, 1.0, 0.0)	134.0
0	#	#	(0.0, 0.0, 0.833, 0.0)	69.2
0	#	0	(0.0, 0.368, 0.0, 0.147)	72.3
0	#	1	(0.0, 0.0, 1.0, 0.0)	83.0
0	0	#	(0.0, 0.0, 0.833, 0.0)	69.2
0	0	0	infeasible	—
0	0	1	(0.0, 0.0, 1.0, 0.0)	83.0
0	1	#	(0.0, 1.0, 0.25, 0.0)	71.8
0	1	0	(0.0, 1.0, 0.0, 0.059)	72.6
0	1	1	(0.0, 1.0, 1.0, 0.0)	134.0
1	#	#	(1.0, 0.0, 0.0, 0.0)	123.0
1	#	0	(1.0, 0.0, 0.0, 0.0)	123.0
1	#	1	(1.0, 0.0, 1.0, 0.0)	206.0
1	0	#	(1.0, 0.0, 0.0, 0.0)	123.0
1	0	0	(1.0, 0.0, 0.0, 0.0)	123.0
1	0	1	(1.0, 0.0, 1.0, 0.0)	206.0
1	1	#	(1.0, 1.0, 0.0, 0.0)	174.0
1	1	0	(1.0, 1.0, 0.0, 0.0)	174.0
1	1	1	(1.0, 1.0, 1.0, 0.0)	257.0

12-24 Consider a maximizing MILP over $x_1 \geq 0$, and $x_2, x_3, x_4 = 0$ or 1.

x_2	x_3	x_4	LP Optimum	LP Value
#	#	#	(0, 1, .17, 0)	232.67
#	#	0	(0, 1, .17, 0)	232.67
#	#	1	(0, .67, 0, 1)	220.00
#	0	#	(0, 1, 0, .25)	230.00
#	0	0	(1.25, 1, 0, 0)	211.25
#	0	1	(0, .67, 0, 1)	220.00
#	1	#	(0, .44, 1, 0)	229.33
#	1	0	(0, .44, 1, 0)	229.33
#	1	1	(0, 0, 1, 1)	216.00
0	#	#	(0, 0, 1, 1)	216.00
0	#	0	(5, 0, 1, 0)	141.00
0	#	1	(0, 0, 1, 1)	216.00
0	0	#	(7.5, 0, 0, 1)	87.50
0	0	0	(12.5, 0, 0, 0)	12.50
0	0	1	(7.5, 0, 0, 1)	87.50
0	1	#	(0, 0, 1, 1)	216.00
0	1	0	(5, 0, 1, 0)	141.00
0	1	1	(0, 0, 1, 1)	216.00
1	#	#	(0, 1, .17, 0)	232.67
1	#	0	(0, 1, .17, 0)	232.67
1	#	1	infeasible	—
1	0	#	(0, 1, 0, .25)	230.00
1	0	0	(1.25, 1, 0, 0)	211.25
1	0	1	infeasible	—
1	1	#	infeasible	—
1	1	0	infeasible	—
1	1	1	infeasible	—

- (a) Solve the problem by LP-based Branch and Bound Algorithm 12A, using the given table of candidate problem solutions, and record your computations in a branch and bound tree. When more than one node is active, select the deepest in the tree, breaking ties in favor of the child with newly fixed variable value most like that of its parent’s relaxation optimum. When needed, branch on the fractional variable of the most recent LP relaxation that is closest to integer in value. Start with no incumbent solution, and do not round to create early incumbents.
- (b) Briefly explain why the logic of Branch and Bound assures your final solution is optimal.
- (c) The table used for part (a) is a convenience, but really solving the given ILP by branch and bound would have required actually solving a series of candidate problem LP relaxations. How many would have been needed to do the computations of part (a)?

12-25 Consider a minimizing MILP over x_1, x_2, x_3 binary and $x_4 \geq 0$.

x_1	x_2	x_3	LP Optimum	LP Value
#	#	#	(1.0, 0.143, 0.0, 0.0)	2.429
#	#	0	(1.0, 0.143, 0.0, 0.0)	2.429
#	#	1	(0.7, 0.0, 1.0, 0.0)	5.400
#	0	#	(1.0, 0.0, 0.25, 0.0)	3.000
#	0	0	(1.0, 0.0, 0.0, 0.067)	5.333
#	0	1	(0.7, 0.0, 1.0, 0.0)	5.400
#	1	#	(0.4, 1.0, 0.0, 0.0)	3.800
#	1	0	(0.4, 1.0, 0.0, 0.0)	3.800
#	1	1	(0.3, 1.0, 1.0, 0.0)	7.600
0	#	#	Infeasible	—
0	#	0	Infeasible	—
0	#	1	Infeasible	—
0	0	#	Infeasible	—
0	0	0	Infeasible	—
0	0	1	Infeasible	—
0	1	#	Infeasible	—
0	1	0	Infeasible	—
0	1	1	Infeasible	—
1	#	#	(1.0, 0.143, 0.0, 0.0)	2.429
1	#	0	(1.0, 0.143, 0.0, 0.0)	2.429
1	#	1	(1.0, 0.0, 1.0, 0.0)	6.000
1	0	#	(1.0, 0.0, 0.25, 0.0)	3.000
1	0	0	(1.0, 0.0, 0.0, 0.067)	5.333
1	0	1	(1.0, 0.0, 1.0, 0.0)	6.000
1	1	#	(1.0, 1.0, 0.0, 0.0)	5.000
1	1	0	(1.0, 1.0, 0.0, 0.0)	5.000
1	1	1	(1.0, 1.0, 1.0, 0.0)	9.000

- (a) Solve the problem by LP-based Branch and Bound Algorithm 12A, using the given table of candidate problem solutions, and record your computations in a branch and bound tree. When more than one node is active, apply the Depth-First rule, breaking ties in favor of the child with newly fixed variable value most like that of its parent’s relaxation optimum. When needed, branch on the fractional variable of the most recent LP relaxation that is closest to integer in value. Start with no incumbent solution, and do not round to create early incumbents.
- (b) Briefly explain why the logic of branch and bound assures your final solution is optimal.
- (c) Point out the first time in your enumeration of (a) where a comparable search based on depth-forward-best-back enumeration would have taken up a different candidate than was done in your depth-first sequence. Explain.

12-26 Consider a maximizing mixed-integer liner program with $x_1 \geq 0, x_2, x_3, x_4 = 0$ or 1.

x_2	x_3	x_4	LP Relaxation	Value
#	#	#	(0.0, 0.58, 1.00, 0.00)	135.0
#	#	0	(0.0, 0.58, 1.00, 0.00)	135.0
#	#	1	(0.0, 0.00, 0.75, 1.00)	125.0
#	0	#	(0.0, 1.00, 0.00, 1.00)	110.0
#	0	0	(0.9, 1.00, 0.00, 0.00)	87.5
#	0	1	(0.0, 1.00, 0.00, 1.00)	110.0
#	1	#	(0.0, 0.58, 1.00, 0.00)	135.0
#	1	0	(0.0, 0.58, 1.00, 0.00)	135.0
#	1	1	infeasible	—
0	#	#	(0.0, 0.00, 1.00, 0.64)	131.8
0	#	0	(0.6, 0.00, 1.00, 0.00)	117.5
0	#	1	(0.0, 0.00, 0.75, 1.00)	125.0
0	0	#	(1.0, 0.00, 0.00, 1.00)	80.0
0	0	0	(1.9, 0.00, 0.00, 0.00)	57.5
0	0	1	(1.0, 0.00, 0.00, 1.00)	80.0
0	1	#	(0.0, 0.00, 1.00, 0.64)	131.8
0	1	0	(0.6, 0.00, 1.00, 0.00)	117.5
0	1	1	infeasible	—
1	#	#	(0.0, 1.00, 0.69, 0.00)	128.8
1	#	0	(0.0, 1.00, 0.69, 0.00)	128.8
1	#	1	(0.0, 1.00, 0.00, 1.00)	110.0
1	0	#	(0.0, 1.00, 0.00, 1.00)	110.0
1	0	0	(0.9, 1.00, 0.00, 0.00)	87.5
1	0	1	(0.0, 1.00, 0.00, 1.00)	110.0
1	1	#	infeasible	—
1	1	0	infeasible	—
1	1	1	infeasible	—

- (a) Solve the problem by LP-based Branch and Bound Algorithm 12A, using the given table of candidate problem solutions, and record your computations in a branch and bound tree. When more than one node is active, apply the Depth-First rule, breaking ties in favor of the child with newly fixed variable value most like that of its parent’s relaxation optimum. When needed, branch on the fractional variable of the most recent LP relaxation that is closest to integer in value. Start with no incumbent solution, and do not round to create early incumbents.
- (b) Briefly explain why the logic of branch and bound assures your final solution is optimal.
- (c) Point out the first time in your enumeration of (a) where a comparable search based on best-first enumeration would have taken up a different candidate than was done in your depth-first sequence. Explain.

12-27 Students often mistakenly believe ILPs are more tractable than LPs because the straightforward rules of Algorithm 12A seem less complex than the simplex and interior point methods of Chapters 5–7.

- (a) Explain why solution of any ILP by LP-based branch and bound always takes at least as much work as a linear program of comparable size and coefficients.
- (b) Justify the number of LP relaxations that might have to be solved in branch-and-bound enumeration of an ILP with n binary variables is $2^{(n+1)} - 1$.
- (c) Use part (b) to compute the number of linear programs that could have to be solved in branch and bound search of ILP models with 100, 300, and 500 binary variables respectively, and determine how long each such search could take at the rate of one LP per second.
- (d) How practical is it be to solve LPs of 100, 300, and 500 variables in reasonable amounts of time?

(e) Comment on the implications of your analysis in parts (a) to (d) for tractability of LPs versus ILPs of comparable size.

12-28 In most applications, LP based Branch and Bound Algorithm 12A actually investigates only a tiny fraction of the possible partial solutions. Still, this is not always the case. Consider the family of ILPs of the form

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & 2 \sum_{j=1}^n x_j + y = n \\ & x_j = 0 \text{ or } 1 \quad j = 1, \dots, n \\ & y = 0 \text{ or } 1 \end{aligned}$$

where n is odd.

- (a) Enter and solve versions for $n = 7, n = 11,$ and $n = 15$ with class branch and bound software, and record the number of branch and bound nodes explored. (*Warning:* Program limits must be set big enough to allow up to 20,000 branch and bound nodes.)
- (b) Express your results in part (a) as fractions as the total number of nodes that might have to be investigated. [*Hint:* Use the formula in Exercise 12-27(b).]
- (c) Comment on the implications for tractability of ILP's via branch and bound if fractions like those of part (b) were typical.

✓ **12-29** The branch and bound tree Figure 12.10 records solution of the knapsack model

$$\begin{aligned} \min \quad & 90x_1 + 50x_2 + 54x_3 \\ \text{s.t.} \quad & 60x_1 + 110x_2 + 150x_3 \geq 50 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

by LP-based Algorithm 12A under rules of Exercise 12-22.

Briefly describe the processing, including how and why nodes were branched or terminated, when incumbent solutions were discovered, and what solution proved optimal. Assume that there was no initial incumbent solution.

12-30 Do Exercise 12-29 for

$$\begin{aligned} \max \quad & 51x_1 + 72x_2 + 41x_3 \\ \text{s.t.} \quad & 17x_1 + 10x_2 + 14x_3 \leq 19 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

and tree in Figure 12.11

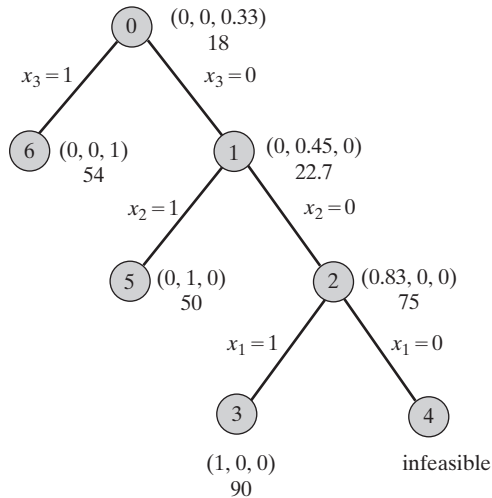


FIGURE 12.10 Branch and Bound Tree for Exercise 12-29

12-31 Return to the knapsack problem of Exercise 12-29.

- (a) Explain why LP relaxation optimal solutions can be rounded to integer-feasible solutions by setting $\hat{x}_j \leftarrow \lceil \tilde{x}_j \rceil$.
- ✓ (b) Repeat the branch and bound computations of Exercise 12-29, this time rounding up each relaxation solution in this way to produce earlier incumbent solutions.
- (c) Comment on the computational savings with rounding.

12-32 Do Exercise 12-31 on the knapsack model of Exercise 12-30, this time rounding solutions $\hat{x}_j \leftarrow \lfloor \tilde{x}_j \rfloor$.

12-33 For each of the following branch and bound trees determine the best lower and upper bounds on the value of an optimal solution known from parent bounds and incumbent solutions after processing of each node. Also show the maximum percent error that would have resulted if processing had terminated after the node, accepting the current incumbent solution as an approximate optimum.

- ✓ (a) The tree of Exercise 12-29.
- (b) The tree of Exercises 12-30.

12-34 The branch and bound tree that follows shows the incomplete solution of a maximizing ILP

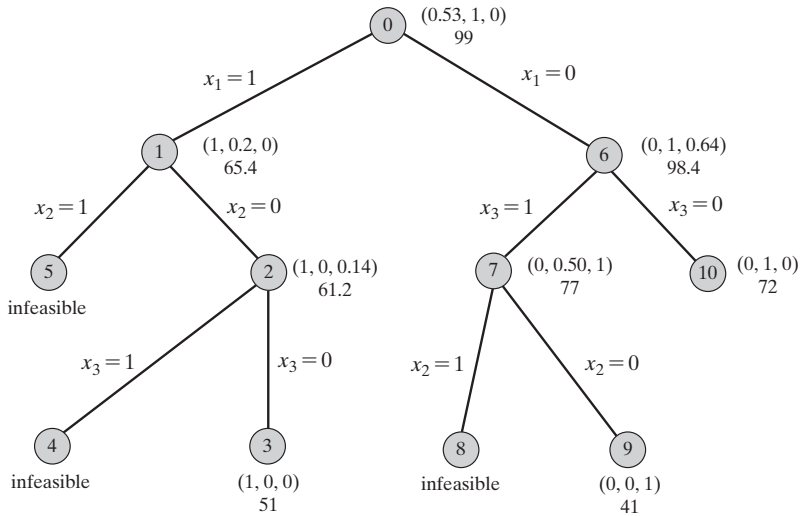
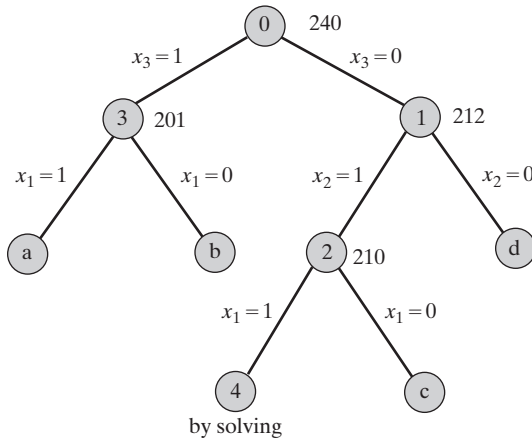


FIGURE 12.11 Branch and Bound Tree for Exercise 12-30

by LP-based Algorithm 12A, with numbers next to nodes indicating LP relaxation solution values.



Node 4 has just produced the first incumbent solution, and nodes *a* to *d* remain unexplored.

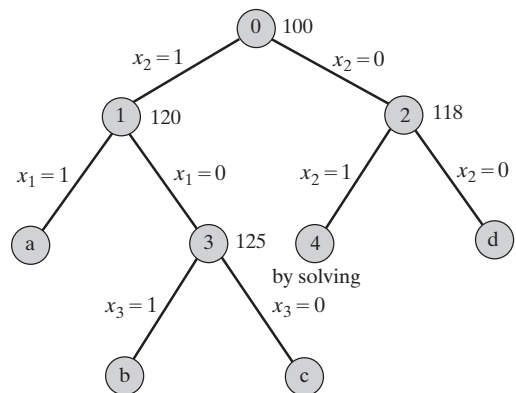
- ✓ (a) Show which unexplored nodes could be immediately terminated by parent bound if the incumbent at node 4 had objective function value 205. How about 210?
- ✓ (b) Determine the best upper bound on the ultimate ILP optimal value that is available after processing of node 4.
- ✓ (c) Assuming the incumbent at node 4 has objective value 195, compute the

maximum absolute and percent objective value error in accepting the incumbent as an approximate optimum.

12-35 The branch and bound tree that follows shows the incomplete solution of a minimizing ILP by LP-based Algorithm 12A, with numbers next to nodes indicating LP relaxation solution values.

Node 4 has just produced the first incumbent solution, and nodes *a* to *d* remain unexplored.

- (a) Show which unexplored nodes could be immediately terminated by the parent bound if the incumbent at node 4 had objective function value 120. How about 118?



- (b) Determine the best lower bound on the ultimate ILP optimal value that is available after processing of node 4.
- (c) Assuming that the incumbent at node 4 has objective value 125, compute the maximum absolute and percent objective value error in accepting the incumbent as an approximate optimum.

12-36 Repeat Exercise 12-22, following the same rules except:

- ✓ (a) Use the best-first enumeration sequence and allow termination by parent bounds.
- ✓ (b) Use the depth-forward best-back enumeration sequence and allow termination by parent bounds.

12-37 Do Exercise 12-36 for the branch and bound of Exercise 12-26.

12-38 Three company trucks must be assigned to pickup 7 miscellaneous loads on the way back from their regular deliveries. Truck capacities and load sizes (cubic yards) are shown in the following table, together with the extra distance (in miles) that each truck would have to travel if it is to deviate to pick up any of the loads.

Load	Distance for Truck:			Load Size
	1	2	3	
1	23	45	50	4
2	25	72	23	8
3	29	13	41	13
4	12	23	40	31
5	49	7	42	11
6	37	39	59	9
7	2	9	20	21
Capacity	30	40	50	

- (a) Formulate this problem as a generalized assignment ILP using the decision variables ($i = 1, \dots, 7; j = 1, \dots, 3$)

$$x_{i,j} \triangleq \begin{cases} 1 & \text{if load } i \text{ goes to truck } j \\ 0 & \text{otherwise} \end{cases}$$

- ✓ (b) Enter and use class optimization software to compute an optimal solution.
- ☐ (c) Use class optimization software to solve the corresponding LP relaxation and verify that the relaxation optimal value provides a lower bound.

- ☐ (d) Using class optimization software to solve relaxations, verify your ILP optimal solution by executing LP-based branch and bound Algorithm 12A including parent bounds. Apply the depth-first rule for selecting among active nodes and pick whichever of $= 0$ and $= 1$ is closest to the preceding relaxation value when nodes have equal depth. Branch on the integer-restricted variable with fractional relaxation value nearest to integer picking the one with least subscript if there are ties. Record incumbent solutions only when an LP relaxation comes out integer (i.e., do not round).
- (e) Determine when your search of part (d) could have been stopped if we were willing to accept an incumbent solution no worse than 25% above optimal.
- ☐ (f) Do the same branch and bound computation as part (d) except with the best first enumeration rule using LP relaxation values as parent bounds.
- ☐ (g) Do the same branch and bound computation as part (d) except with the depth forward best back enumeration rule using LP relaxation values as parent bounds.
- (h) Compare your results in parts (d), (f) and (g).

12-39 Return to the ILP of Exercise 12-12 with LP relaxation optimum $\tilde{\mathbf{x}} = (1, 1, 0, 0)$, $\tilde{y} = \frac{1}{2}$. Determine whether each of the following is a valid inequality for the ILP, and if so, whether it would strengthen the original LP relaxation to add the inequality as a constraint.

- ✓ (a) $x_2 + x_3 + x_4 \geq 3y$
- ✓ (b) $x_1 + x_2 + x_3 + x_4 \geq 4y$
- ✓ (c) $x_1 + x_2 \geq 1$
- ✓ (d) $x_3 \geq y$

12-40 Return to the ILP of Exercise 12-13 with LP relaxation optimum $\tilde{\mathbf{x}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Determine whether each of the inequality for the ILP, and if so, whether it would strengthen the original LP relaxation to add the inequality as a constraint.

- (a) $10x_1 + 10x_2 + 10x_3 \geq 25$
- (b) $x_1 + x_2 + x_3 \geq 1$
- (c) $x_1 + x_2 + x_3 \geq 2$
- (d) $14x_1 + 20x_2 + 16x_3 \geq 30$

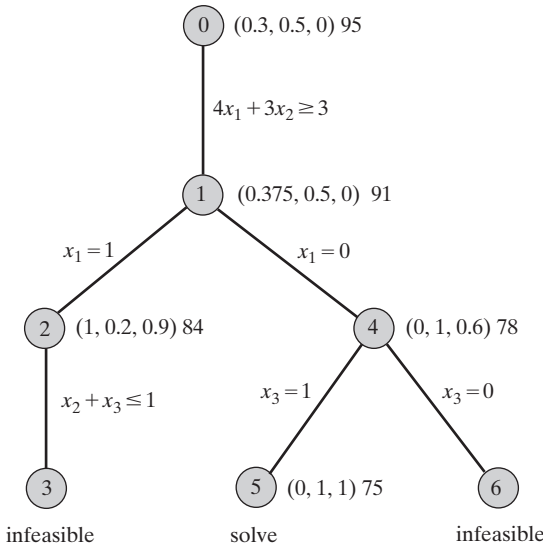
12-41 The ILP

$$\begin{aligned} \max \quad & 40x_1 + 5x_2 + 60x_3 + 8x_4 \\ \text{s.t.} \quad & 18x_1 + 3x_2 + 20x_3 + 5x_4 \leq 25 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

has LP relaxation optimum $\tilde{\mathbf{x}} = (\frac{5}{18}, 0, 1, 0)$. Determine whether each of the following is a valid inequality for the ILP, and if so, whether it would strengthen the LP relaxation to add the inequality as a constraint.

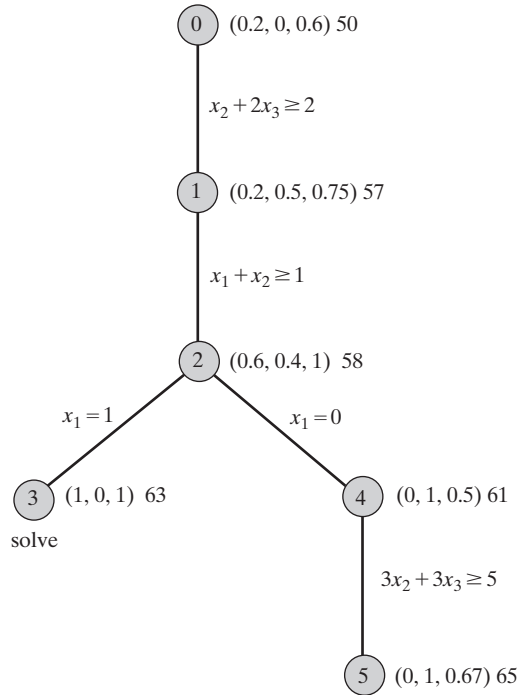
- ✓ (a) $x_1 + x_3 \leq 1$
- (b) $x_1 + x_2 + x_3 + x_4 \leq 3$
- ✓ (c) $x_2 + x_4 \geq 1$
- (d) $18x_1 + 20x_3 \leq 25$

12-42 ✓ The following tree records solution of a maximizing ILP over $x_1, x_2, x_3 = 0$ or 1 by branch and cut Algorithm 12B. LP relaxations solutions show next to each node.

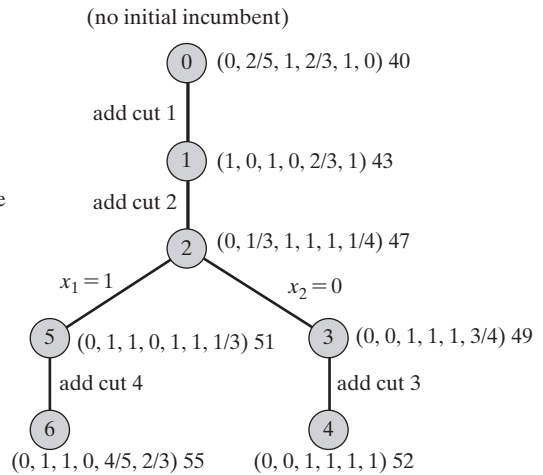


Briefly describe the processing, including how and why nodes were branched tightened or terminated, when incumbent solutions were discovered, and what solution proved optimal. Assume that all added inequalities are valid for the original ILP.

12-43 Do Exercise 12-42 for the following tree of a minimizing ILP over $x_1, x_2, x_3 = 0$ or 1.



12-44 The tree below shows the evolution of a hypothetical Branch and Cut Algorithm 12B solution of a given minimizing, all-binary ILP. Relaxation solutions are shown next to nodes, and both fixed variables and new cuts are indicated on the branches.



Take nodes in sequence, one by one, and briefly describe what apparently happened and why. Also show the ultimate optimal solution. Assume that generated cuts are all valid for the original ILP, and that whenever branching or termination occurs, the cut-generation subroutine has concluded that no valid inequality of the family being used can separate the last LP relaxation.

12-45 The LP relaxation of a standard form ILP over constraints $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, \mathbf{x} integer, with 3 rows and 7 variables, has been solved for basic variables x_1 , x_2 , and x_5 to obtain the dictionary form (see Section 5.4).

	RHS		x_3	x_4	x_6	x_7
$x_1 =$	1.6		-2.7	1.1	-2.3	13.4
$x_2 =$	3.0	=	3.9	-4.7	2.8	2.2
$x_3 =$	2.4		0.6	0.0	13.6	-5.9

- (a) Generate the Gomory fractional cutting plane for x_1 .
- (b) Generate all other available Gomory fractional cutting planes.

12-46 Consider the binary knapsack polytope

$$18x_1 + 3x_2 + 20x_3 + 17x_4 + 6x_5 + 10x_6 + 5x_7 + 12x_8 \leq 25$$

$$x_1, \dots, x_8 = 0 \text{ or } 1$$

- (a) Identify 5 minimal cover inequalities valid for this knapsack polytope, including the one involvin the last 3 variables, and briefly justify why each meets the requirements of definition 12.48.
- (b) Explain how the valid inequality forms of (a) remain valid, and could contribute to sharpening the LP relaxation, if the knapsack constraints above were only part a larger model with many constraints.

12-47 Air National Guard planners are loading a cargo plane with crates of supplies and equipment needed to help victims of the recent wave of floods in the Northeast. The table below shows the criticality of the need for each crate, its weight (in tons), and its volume (in hundreds of cubic feet). The plane’s capacity limits are 38 tons and 26 hundred cubic feet of cargo.

Crate $j =$	1	2	3	4	5	6	7	8
Criticality	9	3	22	19	21	14	16	23
Weight	5	11	15	12	20	7	10	18
Volume	7	4	13	9	4	18	9	6

- (a) Formulate the problem of choosing the maximum total criticality combination of crates to put on the plane as a pure 0-1 ILP over two main constraints and decision variables $x_j = 1$ if crate j is selected and $= 0$ otherwise. Be sure to annotate the objective and each constraint to indicate their meaning.
- (b) Derive one minimum cover inequality from each of the two main constraints of your model in (a). Demonstrate that each meets the requirements of definition 12.48 for such constraints, and explain why both must be valid for the full model of (a).
- (c) Suppose now that we are solving your model of (a) by Branch and Cut Algorithm 12B, and the first LP-relaxation solution is $\mathbf{x} = (0.5, 0, 0, 1, 0, 0, 1, 0.75)$. Which, if either of your cuts in (b) could the algorithm add productively as the next step? Explain.

12-48 Consider the binary ILP

$$\max \quad 21x_1 + 21x_2 + 48x_3 + 33x_4 + 18x_5 + 17x_6 + 39x_7$$

$$\text{s.t.} \quad 5x_1 + 14x_2 + 12x_3 + 21x_4 + 1x_5 + 6x_6 + 13x_7 \leq 30 \quad \text{[i]}$$

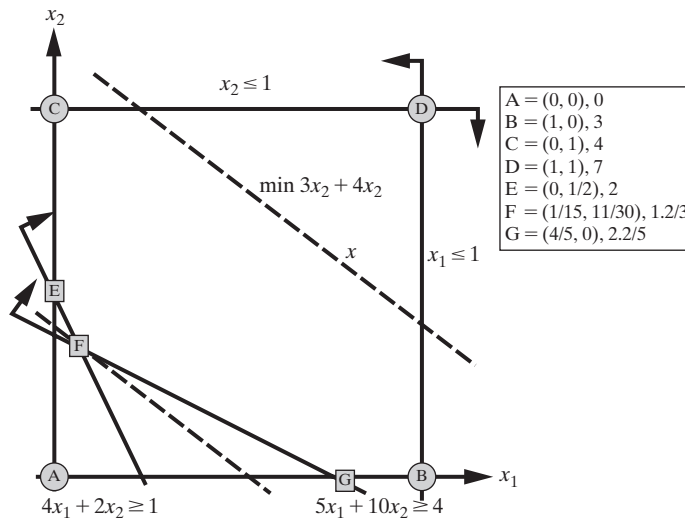
$$x_1 + x_2 + x_3 \leq 1 \quad \text{[ii]}$$

$$x_j \text{ binary for all } j \quad \text{[iii]}$$

- (a) Use class optimization software to solve both the full ILP and its LP relaxation.
- (b) Explain why minimum cover constraints 12.48 derived from constraints [i] and [iii] are valid for the full model.
- (c) Now consider solving the given ILP via Branch and Cut Algorithm 12B, starting from your LP relaxation of (a), and invoking class optimization software as each node is investigated. Use minimal cover inequalities from (b) when adding appropriate cuts, and branch on the fractional variable with highest subscript. Branch after at most 3 cuts have been generated for any partial solution, and continue until a total of 5 nodes have been explored.

12-49 The plot below depicts the solution space of pure binary ILP

$$\begin{aligned} \min \quad & 3x_1 + 4x_2 \\ \text{s.t.} \quad & 4x_1 + 2x_2 \geq 1 \quad 5x_1 + 10x_2 \geq 4 \quad x_1, x_2 = 0 \text{ or } 1 \end{aligned}$$



Coordinates and solution values of important points are pre-computed in the key.

- ✓ (a) Sketch the convex hull of integer solutions on the plot and briefly justify your choice.
- ✓ (b) Determine the dimension of the convex hull, and justify your answer with suitable affinely independent points.
- ✓ (c) Determine graphically whether each of the three inequalities below is valid. Then, determine whether each intersects the convex hull in a facet, or in a face of lower dimension, or not at all, including calling out affinely independent points that satisfy the cuts as equalities in the facet and face cases.

$$\begin{aligned} 4x_1 + 2x_2 &\geq 1 \\ 5x_1 + 2x_2 &\geq 2 \\ x_1 + x_2 &\geq 1 \end{aligned}$$

12-50 Consider a pure-integer program over constraints

$$\begin{aligned} -1x_1 + 2x_2 &\leq 4 \\ 5x_1 + 1x_2 &\leq 20 \\ -2x_1 - 2x_2 &\leq -7 \\ x_1, x_2 &\geq 0 \text{ and integer} \end{aligned}$$

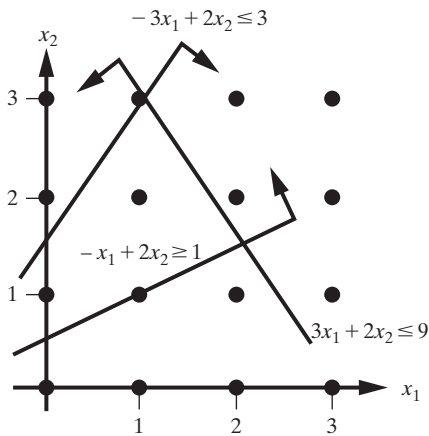
- (a) Draw a 2-dimensional plot of the LP-relaxation feasible set.

- (b) Identify all integer-feasible solutions and their convex hull.
- (c) Determine the dimension of this convex hull.
- (d) Identify all facet-defining inequalities of the convex hull and the required affinely independent solutions proving each.
- (e) Identify another non-facet-defining inequality that at least intersects the convex hull.
- (f) Identify another valid inequality that cuts off part of the LP-feasible set without intersecting the convex hull at all.

12-51 Return to the ILP of Exercises 12-13 and 12-40.

- (a) Draw a 3-dimensional sketch of the convex hull of feasible solutions to this model
- (b) Determine the dimension of the convex hull.
- (c) For each of (a)–(d) in Exercise 12-40 yielding a valid inequality, determine (by showing the required number of affinely independent solutions) whether it is facet-inducing; or intersects the convex hull in a face of lower dimension; or neither.

12-52 The following plot shows the feasible space for an ILP over nonnegative integer variables x_1 and x_2 .



- Identify the convex hull of integer-feasible solutions.
- Determine the dimension of the convex hull and confirm your answer with a suitable number of affinely independent points.
- Derive the equations of all facet-defining inequalities of your convex hull in (a).
- Develop the equation of another facet-defining inequality that is not facet-defining.
- Develop the equation of another valid inequality that cuts off part of the LP-feasible space but does not induce a face of any dimension.

REFERENCES

Chen, Der-San, Robert G. Batson, and Yu Dang (2010), *Applied Integer Programming - Modeling and Solution*, Wiley, Hoboken, New Jersey.

Nemhauser, George L. and Laurence Wolsey (1988), *Integer and Combinatorial Optimization*, John Wiley, New York, New York.

Parker, R. Gary and Ronald L. Rardin (1988), *Discrete Optimization*, Academic Press, San Diego, California.

Schrijver, Alexander (1998), *Theory of Linear and Integer Programming*, John Wiley, Chichester, England.

Wolsey, Laurence (1998), *Integer Programming*, John Wiley, New York, New York.

Large-Scale Optimization Methods

Almost all of the optimization methods considered in this book directly address a fully formulated model of an application of interest. Starting with a feasible solution, they search systematically for better ones until a satisfactory result is at hand. Some temporary modifications may relax requirements like integrality and/or fix some variable values as part of a partial enumeration search, but the full formulation is always in sight.

This chapter introduces the **large-scale** (or **decomposition**) alternatives that adopt a more indirect approach to deal with problem forms having instances that quickly become too big or too complex to be addressed frontally. The methods “outsource” to one or more **subproblem** pieces of the models that are tractable enough to be solved repeatedly if considered alone. An associated **master problem** produces an overall exact or near-exact optimum by integrating results from frequent subproblem invocations, changing their parameters with guidance from the master.

13.1 DELAYED COLUMN GENERATION AND BRANCH AND PRICE

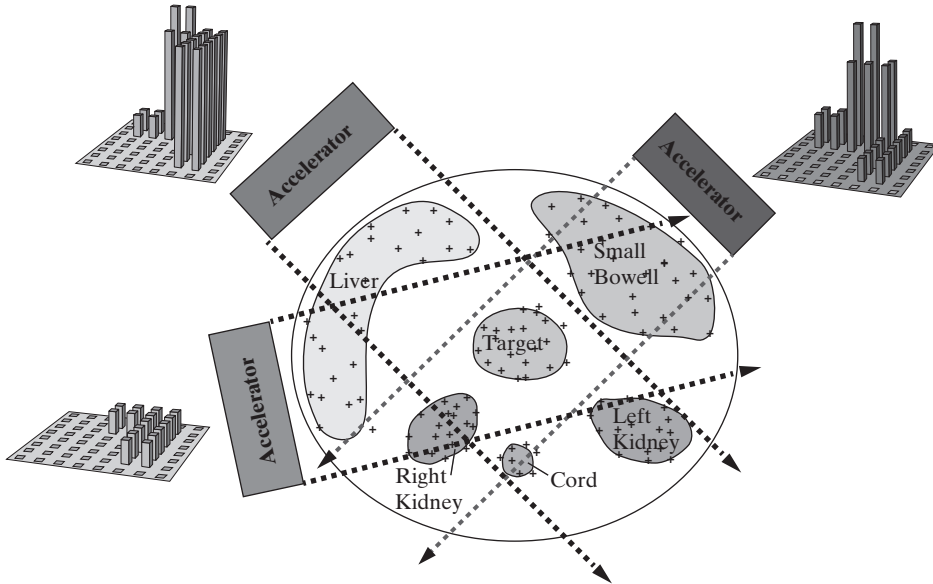
The **Delayed Column Generation** large-scale strategy becomes necessary when the optimization needs to address combinatorially many decision options that can be represented by columns of a full formulation. Instead of explicitly addressing all the options at once, a **partial master problem** optimizes over just the subset explicitly generated so far. The rest are delayed until identified by a **column-generation** subproblem as ones that can improve the current master problem solution. The overwhelming number of possible columns never need to be considered explicitly because the column-generation subproblem shows implicitly that they cannot improve the current solution.

As usual, it will help to begin with an example application.¹

¹Based on F. Preciado-Wlaters, M. Langer, R. Rardin, and V. Thai (2006), “Column Generation for IMRT Cancer Therapy Optimization with Implementable Segments,” *Annals of Operations Research*, 148, 55–63.

APPLICATION 13.1: IMRT PLANNING FOR RADIATION THERAPY OPTIMIZATION

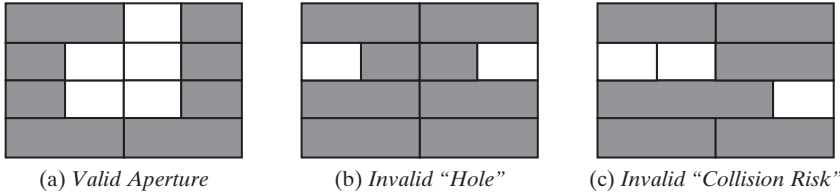
Radiation therapy planning for cancer treatment begins with images of body cross-sections like the figure below. A tumorous target (here the prostate) is identified along with surrounding healthy tissues. The goal of the treatment is to provide maximum radiation to the tumor while avoiding damage to surrounding tissues by limiting radiation to these tissues.



Radiation is provided from a large **accelerator** that can shoot beams from multiple angles around the patient's body (here 3 beams are shown) so as to spread the danger to healthy tissues while focusing on the tumor. The accelerator beam is relatively large, often approximately 10 cm square. That is why **Intensity Modulated Radiation Therapy (IMRT)** adds precision to plans by treating each beam as made up of many **beamlets** (often a few hundred) with different intensities (roughly exposure times).

The beamlets are virtual, not physical, with their impacts induced by focusing the beam through a **multileaf collimator** with moving leaves (or fingers) that limit intensity to an open **aperture** as shown in the following figure. Effects of separate apertures add to produce a combined beamlet-by-beamlet impact for each angle like those shown as bar graphs in the following figure.

Not every combination of beamlets forms a valid aperture. Among many limitations, it must be possible to create the chosen opening just by moving leaves in from the two boundaries; closed "holes" or middle areas such as row 2 of example (b) are impossible. Another rule preclude leaves overlapping as in rows 2 and 3 of example (c) because overlaps constitute a "collision risk."



Each tissue is modeled as a discrete set of points, with the objective of maximizing total (or equivalently average) dose delivered to tumor points, and constraints limiting the total dose at each point of the various healthy tissues to the maximum safe level. Input parameters estimate the dose per unit intensity of beamlets to points:

$t_{i,q,k} \triangleq$ the dose per unit intensity to tumor point i by beamlet q of angle k
 $d_{i,h,q,k} \triangleq$ the dose per unit intensity to tumor point i in healthy tissue h
 by beamlet q of angle k

Table 13.1 shows fictitious values for a tiny 2-angle, 18-beamlet, 18-point instance. Dose limits for the 3 healthy tissues are $b_1 = 50$, $b_2 = 65$, and $b_3 = 70$, respectively.

Once beamlet sets $Q_{m,k}$ for apertures m of angles k have been chosen, aggregations of detail data such as Table 13.1 become

$t_{m,k} \triangleq$ the total dose per unit intensity to tumor points
 by aperture m of angle k (i.e., $\sum_{q \in Q_{m,k}} \sum_i t_{i,q,k}$)
 $d_h^{m,k} \triangleq$ vector of dose per unit intensity $d_{i,h}^{m,k}$ to points i of healthy tissue h
 by aperture m of angle k (i.e., the vector of $\sum_{q \in Q_{m,k}} d_{i,h,q,k}$)

Then the corresponding formulation of the IMRT therapy planning problem as a linear program takes the form

$$\begin{aligned}
 \max \quad & \sum_{m,k} t_{m,k} x_{m,k} \\
 (\text{IMRT}) \quad \text{s.t.} \quad & \sum_{m,k} d_h^{m,k} x_{m,k} \leq b \quad \text{for all } h \\
 & x_{m,k} \geq 0 \quad \text{for all } m, k
 \end{aligned}$$

where decision variables $x_{m,k} \triangleq$ the intensity applied to aperture m of angle k , and b_h is a vector of the total dose allowed to points in health tissue h .

Models Attractive for Delayed Column Generation

IMRT radiation therapy planning as in Application 13.1 illustrates settings where delayed column generation can work well because the optimization is relatively simple to express in terms of variables/columns for each possible aperture/column (formulation (IMRT)). If instead, we tried to formulate the problem with beamlet variables and constraints, summing them to form apertures, all the limits on feasible beamlet combinations (like the holes and collision risks illustrated above) would have to be explicitly modeled without destroying the tractability of the model; doing that is often difficult if not impossible.

TABLE 13.1 Dose Data for Small IMRT Instance

	Pt	Angle 1 Beamlets								
		1	2	3	4	5	6	7	8	9
Target	1	0.100	0.250	0.100	0.080	0.150	0.080	0.050	0.100	0.050
	2	0.090	0.070	0.050	0.050	0.070	0.090	0.150	0.120	0.100
	3	0.090	0.070	0.050	0.050	0.070	0.090	0.090	0.080	0.070
	4	0.100	0.120	0.150	0.090	0.120	0.150	0.090	0.070	0.050
	5	0.090	0.080	0.110	0.080	0.070	0.130	0.090	0.070	0.050
	6	0.050	0.100	0.050	0.070	0.120	0.090	0.100	0.250	0.100
Healthy No. 1	1	0.150	0.375	0.150	0.120	0.225	0.120	0.075	0.150	0.075
	2	0.135	0.105	0.075	0.075	0.105	0.135	0.225	0.180	0.150
	3	0.135	0.105	0.075	0.075	0.105	0.135	0.135	0.120	0.105
	4	0.150	0.180	0.225	0.135	0.180	0.225	0.135	0.105	0.075
	5	0.135	0.120	0.165	0.120	0.105	0.195	0.135	0.105	0.075
	6	0.075	0.150	0.075	0.105	0.180	0.135	0.150	0.375	0.150
Healthy No. 2	1	0.070	0.100	0.075	0.042	0.060	0.042	0.060	0.040	0.028
	2	0.020	0.028	0.068	0.020	0.028	0.020	0.038	0.048	0.034
	3	0.020	0.028	0.068	0.020	0.028	0.020	0.038	0.032	0.022
	4	0.034	0.048	0.075	0.034	0.048	0.034	0.068	0.028	0.020
	5	0.022	0.032	0.068	0.020	0.028	0.020	0.060	0.028	0.020
	6	0.028	0.040	0.038	0.034	0.048	0.034	0.053	0.100	0.070
Healthy No. 3	1	0.090	0.225	0.090	0.072	0.135	0.072	0.045	0.090	0.045
	2	0.081	0.063	0.045	0.045	0.063	0.081	0.135	0.108	0.090
	3	0.081	0.063	0.045	0.045	0.063	0.081	0.081	0.072	0.063
	4	0.090	0.108	0.135	0.081	0.108	0.135	0.081	0.063	0.045
	5	0.081	0.072	0.099	0.072	0.063	0.117	0.081	0.063	0.045
	6	0.045	0.090	0.045	0.063	0.108	0.081	0.090	0.225	0.090
	Pt	Angle 2 Beamlets								
		1	2	3	4	5	6	7	8	9
Target	1	0.100	0.250	0.100	0.080	0.150	0.080	0.050	0.100	0.050
	2	0.100	0.120	0.150	0.090	0.120	0.150	0.090	0.070	0.050
	3	0.090	0.080	0.110	0.080	0.070	0.130	0.090	0.070	0.050
	4	0.090	0.070	0.050	0.050	0.070	0.090	0.150	0.120	0.100
	5	0.090	0.070	0.050	0.050	0.070	0.090	0.090	0.080	0.070
	6	0.050	0.100	0.050	0.070	0.120	0.090	0.100	0.250	0.100
Healthy No. 1	1	0.075	0.100	0.070	0.060	0.060	0.042	0.038	0.040	0.028
	2	0.068	0.028	0.020	0.038	0.028	0.020	0.113	0.048	0.034
	3	0.068	0.028	0.020	0.038	0.028	0.020	0.068	0.032	0.022
	4	0.075	0.048	0.034	0.068	0.048	0.034	0.068	0.028	0.020
	5	0.068	0.032	0.022	0.060	0.028	0.020	0.068	0.028	0.020
	6	0.038	0.040	0.028	0.053	0.048	0.034	0.075	0.100	0.070
Healthy No. 2	1	0.150	0.375	0.150	0.120	0.225	0.120	0.075	0.150	0.075
	2	0.150	0.180	0.225	0.135	0.180	0.225	0.135	0.105	0.075
	3	0.135	0.120	0.165	0.120	0.105	0.195	0.135	0.105	0.075
	4	0.135	0.105	0.075	0.075	0.105	0.135	0.225	0.180	0.150
	5	0.135	0.105	0.075	0.075	0.105	0.135	0.135	0.120	0.105
	6	0.075	0.150	0.075	0.105	0.180	0.135	0.150	0.375	0.150
Healthy No. 3	1	0.090	0.225	0.090	0.072	0.135	0.072	0.045	0.090	0.045
	2	0.090	0.108	0.135	0.081	0.108	0.135	0.081	0.063	0.045
	3	0.081	0.072	0.099	0.072	0.063	0.117	0.081	0.063	0.045
	4	0.081	0.063	0.045	0.045	0.063	0.081	0.135	0.108	0.090
	5	0.081	0.063	0.045	0.045	0.063	0.081	0.081	0.072	0.063
	6	0.045	0.090	0.045	0.063	0.108	0.081	0.090	0.225	0.090

Delayed column generation separates the construction of columns satisfying all the applicable side constraints from the larger optimization. Still, in a real-world instance with thousands of tissue points and beamlets, the number of possible apertures for any instance quickly becomes combinatorially large. A mechanism must be devised to generate only the columns likely to be a part of an optimal or near-optimal solution.

Principle 13.1 Delayed column generation decompositions are attractive when the model of an application is most naturally expressed in terms of an extended list of options/columns that can be generated as needed by a separate subproblem applying (often ad hoc) constructions to produce attractive and feasible new columns that conform to even difficult-to-model side constraints.

Partial Master Problems

Overall coordination of a delayed column generation decomposition is accomplished by iteratively solving a partial master problem over columns already known explicitly.

Definition 13.2 The partial master problem of a delayed column generation decomposition solves the restricted version, the full model allowing only decision variable for columns explicitly known to take on nonzero values. If the underlying model is an ILP, the LP relaxation of its partial form is solved.

Generic Delayed Column Generation Algorithm

Algorithm 13A presents a generic form of column generation for the following extended linear program:

$$\begin{aligned}
 \max \text{ (or min)} \quad & \sum_{j \in J} c_j x_j \\
 \text{ELP}(J) \quad \text{s.t.} \quad & \sum_{j \in J} \mathbf{a}^{(j)} x_j \leq \mathbf{b} \\
 & x_j \geq 0 \quad \text{for all } j \in J
 \end{aligned} \tag{13.1}$$

Here J_{all} is the set of all columns under consideration, and $J \subseteq J_{all}$ is the subset of explicit columns in the current partial master problem. Symbols x_j , c_j , and $\mathbf{a}^{(j)}$ denote corresponding decision variables, objective coefficients, and constraint coefficient vectors, respectively, with \mathbf{b} the vector of constraint right-hand sides.

Application of Algorithm 13A to Application 13.1

Table 13.2 tracks 3 iterations of Algorithm 13A on the IMRT application model. Two aperture columns are generated as each step, and the associated partial master problem is solved to derive the primal and dual solutions shown.

ALGORITHM 13A: DELAYED COLUMN GENERATION

Step 0: Initialization. Set iteration index $\ell \leftarrow 0$, and choose a $J_0 \subseteq J_{all}$ such that the corresponding partial master problem $ELP(J_0)$ is feasible.

Step 1: Partial Master Solution. Solve partial master problem $ELP(J_\ell)$ for primal optimum $\mathbf{x}^{(\ell)}$ and corresponding dual optimum $\mathbf{v}^{(\ell)}$.

Step 2: Column Generation Subproblem. Attempt to construct a column $\mathbf{a}^{(g)}$ satisfying all column-specific side constraints for $g \in J_{all}$, and having reduced objective $\bar{c}_g \triangleq c_g - \mathbf{a}^{(g)}\mathbf{v}^{(\ell)} < 0$ for a maximize (or > 0 for minimize), which qualifies it to enter the solution of the most recent master problem $ELP(J_\ell)$.

Step 3: Stopping. If no suitable column g was discovered in Step 2, stop. Solution $\mathbf{x}^{(\ell)}$ is an optimal or near optimal solution in full model $ELP(J_{all})$. Otherwise update $J_{\ell+1} \leftarrow J_\ell \cup g$, advance $\ell \leftarrow \ell + 1$, and return to Step 1.

TABLE 13.2 IMRT Application Progress

Aperture Beamlets		$Q_{1,1}$ all	$Q_{2,1}$ all	Dual $\mathbf{v}^{(0)}$	$Q_{1,2}$ 7, 8	$Q_{2,2}$ 1, 4, 7, 8	Dual $\mathbf{v}^{(1)}$	$Q_{1,3}$ 2, 3	$Q_{2,3}$ 8
Total Target		5.050	5.050	—	3.615	4.987	—	3.546	1.932
Healthy No. 1	1	1.440	0.513	0.000	0.225	0.200	0.000	0.525	0.040
	2	1.185	0.394	0.000	0.405	0.245	0.000	0.180	0.048
	3	0.990	0.322	0.000	0.255	0.115	0.000	0.180	0.028
	4	1.410	0.421	0.000	0.240	0.115	0.000	0.180	0.028
	5	1.155	0.345	0.000	0.240	0.157	0.000	0.285	0.028
	6	1.395	0.485	0.000	0.525	0.122	4.840	0.225	0.100
Healthy No. 2	1	0.517	1.440	0.000	0.100	0.106	0.000	0.175	0.150
	2	0.301	1.410	0.000	0.086	0.675	0.000	0.096	0.105
	3	0.274	1.155	0.000	0.070	0.555	0.000	0.096	0.105
	4	0.387	1.185	0.000	0.096	0.420	0.000	0.123	0.180
	5	0.297	0.990	0.000	0.088	0.315	0.000	0.100	0.120
	6	0.443	1.395	0.000	0.153	0.315	0.000	0.078	0.375
Healthy No. 3	1	0.864	0.864	0.000	0.135	0.300	7.957	0.315	0.090
	2	0.711	0.846	5.845	0.243	0.405	0.000	0.108	0.063
	3	0.594	0.693	0.000	0.153	0.333	0.000	0.108	0.063
	4	0.846	0.711	0.000	0.144	0.252	0.000	0.243	0.108
	5	0.693	0.594	0.000	0.144	0.189	0.000	0.171	0.072
	6	0.837	0.837	0.000	0.315	0.189	0.000	0.135	0.225
$\nu = 292.20, \mathbf{x}^{(0)} =$		43.49	14.38	—	—	—	—	—	—
$\nu = 736.63, \mathbf{x}^{(1)} =$		0.000	0.000	—	112.3	49.41	—	—	—
$\nu = 737.20, \mathbf{x}^{(2)} =$		0.000	0.000	—	85.54	13.15	—	82.09	36.91

The algorithm must start with a feasible solution. Here this is accomplished by opening all beamlets in both beam angles. The following illustrates how this is accomplished from data in Table 13.1 for aperture $A_{1,1}$:

$$\begin{aligned} t_{1,1} &= 5.050 \leftarrow \sum_{i,q} t_{i,q,k} \\ &= 0.100 + 0.250 + \dots + 0.250 + 0.100 \quad \text{and} \\ d_{6,2}^{1,1} &= 0.443 \leftarrow \sum_q d_{6,2,q,k} = 0.028 + 0.040 + \dots + 0.070 \end{aligned}$$

Later apertures open only the beamlets listed. For aperture $A_{2,2}$, which uses beamlets 1,4,7, and 8 of angle 2, this yields coefficients

$$\begin{aligned} t_{2,2} &= 4.987 \leftarrow \sum_i (t_{i,q,2} + t_{i,4,2} + t_{i,7,2} + t_{i,8,2}) \\ &= 0.100 + 0.080 + 0.050 + 0.050 + \dots \\ &\quad + 0.050 + 0.070 + 0.100 + 0.100 \quad \text{and} \\ d_{6,2}^2 &= 0.315 \leftarrow d_{6,2,1,2} + d_{6,2,4,2} + d_{6,2,7,2} + d_{6,2,8,2} \\ &= 0.075 + 0.105 + 0.150 + 0.150 \end{aligned}$$

Generating Eligible Columns to Enter

Typically, the most challenging task in any implementation of Algorithm 13A is to conceive a method of finding attractive new columns p using the last dual solution $\mathbf{v}^{(\ell)}$ and corresponding reduced cost $\bar{c}_p \triangleq c_p - \mathbf{a}^{(p)}\mathbf{v}^{(\ell)}$. Note that any such column with $\bar{c}_p > 0$ (< 0 for a minimize) will automatically be new to the partial master problem; optimality in the current one $ELP J_t$ means $\bar{c}_j \leq 0$ for all $j \in J_t$ (≥ 0 for minimize).

Occasionally, such a method actually optimizes over possible p to minimize (or maximize) the resulting \bar{c}_p . Example 13.1 below illustrates. Where possible, such optimization is attractive because it results in a clear stopping rule for Algorithm 13A Step 3. We can terminate when even the best column p has reduced cost failing the required sign.

In most applications, side constraints and other complications on column generation leave Step 2 heuristic. Feasible columns are produced that meet the reduced-cost sign rules and have some reason to suggest they may be attractive.

Aperture generation in model (*IMRT*) falls into the latter category. We proceed from the insight that the reduced objective for aperture m of angle k is the sum of “mini-reduced objectives” for component beamlets of the aperture, that is, $\bar{c}_{m,k} = \sum_{q \in A_{m,l}} \bar{c}_{q,m,k}$. Thus the beamlets for new apertures can be selected by examining those mini-values. Table 13.3 illustrates the aperture columns of Table 13.2. Each block of the table shows the beamlet mini-reduced objectives for one iteration and angle.

Values in bold are the ones for the selected aperture set. For $\ell = 0$ all beamlets are chosen. Later cases for $\ell = 1$ and $\ell = 2$ select feasible combinations of beamlets with the highest mini-reduced objective values. Even for this tiny instance, there are obviously many alternatives that might produce better results, but the process does and can be expected without extensive enumerations or other expensive computations. Here we terminate because the target dose objective improvement with the last pair of apertures is minimal.

TABLE 13.3 IMRT Aperture Selection

Angle 1 $\ell = 0$			Angle 1 $\ell = 1$			Angle 1 $\ell = 2$		
1.441	1.932	1.614	0.915	0.617	1.088	0.374	0.976	0.975
1.176	1.680	1.680	0.755	0.891	1.259	0.496	0.682	0.530
1.683	1.932	1.201	1.420	1.406	0.938	-0.130	0.130	-0.076
Angle 2 $\ell = 0$			Angle 2 $\ell = 1$			Angle 2 $\ell = 2$		
1.638	1.932	1.417	1.112	0.617	0.891	0.515	0.592	-0.228
1.323	1.680	1.680	0.902	0.891	1.259	0.287	0.329	0.027
1.796	1.932	1.201	1.532	1.406	0.938	0.728	1.045	0.569

EXAMPLE 13.1: DELAYED COLUMN GENERATION FOR STOCK CUTTING

Consider the task of cutting $b \triangleq 11$ feet stock boards to meet requirements for various smaller lengths in furniture manufacture as shown in the following table:

i	Length h_i	Needed d_i
1	2	22
2	3	17
3	6	13
4	7	9

The manufacturer wishes to meet all requirements with the minimum number of stock pieces cut.

(a) Define a cutting pattern k as a combination of size quantities $a_{i,k}$ for different lengths p_i totaling to at most b feet. Then, using $K \triangleq$ the set of all such cutting patterns, and decision variables $x_k \triangleq$ the number of times pattern k is to be used, show that this stock cutting problem can be formulated as the following ILP:

$$\begin{aligned}
 \min \quad & \sum_{k \in K} x_k \\
 \text{s.t.} \quad & \sum_{k \in K} a_{i,k} x_k \geq d_i \quad \text{for all } i \\
 & x_k \geq 0 \text{ and integer} \quad \text{for all } k \in K
 \end{aligned}$$

(b) Suppose solving the LP relaxation of a current partial master version ℓ of the model in part (a) has produced dual optima $\{\bar{v}_i\}$. Show that the next column p to enter should solve (over decision variables $a_{i,g}$) a knapsack problem

$$\begin{aligned}
 \min \quad & 1 - \sum_i \bar{v}_i a_{i,g} \\
 \text{s.t.} \quad & \sum_i a_{i,g} \leq b \\
 & a_{i,g} \geq 0 \text{ and integer for all } i
 \end{aligned}$$

(c) Comment on the advantages of a delayed column generation decomposition such as this one where a best new column, or a proof that none exists, can be found by a tractable model.

(d) Suppose Algorithm 13A is being applied starting with a partial master of just the 4 single-length cutting patterns, each producing only one of the lengths needed. Explain why this is an appropriate initial collection of columns.

(e) Solving the initial LP of part (d) yields primal optimal solution $\bar{x} = (4.4, 5.67, 6.5, 9)$ and dual $\bar{v} = (0.2, 0.33, 0.5, 1)$. Use these results and the method of part (b) to solve by inspection for the next column to enter.

Solution:

(a) The objective minimizes the total number of stock boards used, and constraints assure that at least the required number of each piece length will be obtained from the optimal combination of cutting patterns.

(b) The best column will be the one g of most negative reduced objective \bar{c}_g . This knapsack objective function computes the \bar{c}_g of the pattern selected, and the main constraint assures its mix of various length fits within the stock length b .

(c) Having a tractible model for constructing new columns assures rapid progress toward an ultimate LP relaxation optimum in the master problem. Also, failure to produce a column with $\bar{c}_g < 0$ proves conclusively that all needed columns are already present in the last partial master problem.

(d) The requirement for a starting column set is that it can produce a feasible solution to the master problem. Using the single-length columns easily satisfies this need.

(e) The knapsack to find a column will be

$$\begin{aligned} \min \quad & 1 - 0.2*a_1 - 0.33*a_2 - 0.5*a_3 - 1*a_4 \\ \text{s.t.} \quad & 2*a_1 + 3*a_2 + 6*a_3 + 7*a_4 \leq 11 \\ & a_i \geq 0 \text{ and integer for all } i \end{aligned}$$

An optimal solution is $a = (0, 1, 0, 1)$ with objective value = reduced cost $- 0.33$.

Branch and Price Search

Many applications of delayed column generation, including the ones above, keep integer and combinatorial elements in their column-generating subproblems. Then, the master problem across the generated columns can be effectively treated as a Linear Program. For instance, in radiation planning Application 13.1, the decision variables associated with aperture columns are nonnegative and continuous, resulting in an LP master problem. In stock cutting Example 13.1, decision variables are nominally integer—the number of times a cutting pattern is to be used—but easy, near-optimal solutions can be obtained by simply rounding up LP relaxation optimal values to the next higher integer.

As in many other settings, optimization of the master problem becomes more complex when the associated decision variables are binary. Rounding is unlikely to assure near-optimal solutions, and the partial enumeration ideas of Chapter 12 must be engaged.

Branch and Price search methods do exactly that. Much like the Branch and Bound methods of Section 12.5, which combine cut generation with Branch and Bound search over LP relaxations, Branch and Price enhances LP Branch and Bound with periodic generation of new columns with attractive reduced costs. Algorithm 13B provides details on this.

ALGORITHM 13B: BRANCH AND PRICE SEARCH (0–1 ILPS)

Step 0: Initialization. Make the active partial solution the one with all discrete variables free, initialize, solution index $\ell \leftarrow 0$, and take as the root candidate problem one defined over a starting working collection of columns. If (integer) feasible solutions are known for the model, choose the best as incumbent solution $\hat{\mathbf{x}}$ with objective value \hat{v} . Otherwise, define $\hat{v} \leftarrow \pm \infty$ depending on whether the problem minimizes or maximizes.

Step 1: Stopping. If active partial solutions remain, select one as $\mathbf{x}^{(\ell)}$, and proceed to Step 2. Otherwise, stop. If there is an incumbent solution $\hat{\mathbf{x}}$, it is optimal, and if not, the model is infeasible.

Step 2: Relaxation. Attempt to solve the linear programming relaxation of the candidate problem corresponding to $\mathbf{x}^{(\ell)}$.

Step 3: Incumbent Update. If LP relaxation optimum $\tilde{\mathbf{x}}^{(\ell)}$ satisfies all binary constraints of the model and its objective value \tilde{v} is better than incumbent $\hat{\mathbf{x}}$, save $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}^{(\ell)}$ and $\hat{v} \leftarrow \tilde{v}$ as the new incumbent.

Step 4: Column Generation. Invoke a subproblem using optimal dual relaxation solution $\tilde{\mathbf{v}}^{(\ell)}$ to attempt to generate a new column j eligible to enter the current LP relaxation with reduced cost \bar{c}_j ; positive for a maximize problem or negative for a minimize. If any such column j is produced, add it to the current candidate problem, increment $\ell \leftarrow \ell + 1$ and return to Step 2. Otherwise, the current LP relaxation is optimal for the candidate problem.

Step 5: Termination by Infeasibility. If the (now optimal) LP relaxation is integer infeasible, there are no feasible completions of partial solution $\mathbf{x}^{(\ell)}$. Terminate it, increment $\ell \leftarrow \ell + 1$, and return to Step 1.

Step 6: Termination by Bound. If the model maximizes and LP optimal value \hat{v} is at most \tilde{v} , or it minimizes and \hat{v} is at least \tilde{v} , the best feasible completion of partial solution $\mathbf{x}^{(\ell)}$ cannot improve on the incumbent. Terminate it, increment $\ell \leftarrow \ell + 1$, and return to Step 1.

Step 7: Termination by Solving. If the (now optimal) LP relaxation optimum $\tilde{\mathbf{x}}^{(\ell)}$ satisfies all binary constraints of the model, it provides an optimal completion of partial solution $\mathbf{x}^{(\ell)}$. Terminate $\mathbf{x}^{(\ell)}$, increment $\ell \leftarrow \ell + 1$, and return to Step 1.

Step 8: Branching. Choose some free binary-restricted component x_p that was fractional in the last LP relaxation optimum and branch $\mathbf{x}^{(\ell)}$ by creating two new active candidate problems. One is identical to $\mathbf{x}^{(\ell)}$ except that x_p is fixed = 0 and the other is identical to $\mathbf{x}^{(\ell)}$ except that x_p is fixed = 1. Then, increment $\ell \leftarrow \ell + 1$ and return to Step 1.

Figure 13.1 illustrates a synthetic minimizing binary ILP. Computation begins at root node 0 with a working collection of 4 of presumably many possible columns. The corresponding LP relaxation solution $\tilde{\mathbf{x}}^{(0)} = (0, 1, 1, 0)$ happens to be binary-feasible with value = 136, so it provides a first incumbent at Algorithm 13B Step 3.

Notice however, that in contrast to usual Branch and Bound, the search cannot be terminated immediately. Even though this solution is integer feasible, generating additional columns may allow it to be improved. This is exactly what happens at node 1, where adding a new column 5 reduces the relaxation bound to = 129.7. It happens again a node 2 after new column 6 is included to obtain even better value = 124.56.

We assume at this point that no additional column can produce improvement. Thus solution $\tilde{\mathbf{x}}^{(2)}$ solves the LP relaxation of the full model. Still, it cannot be terminated because it is feasible and has bound less than the current incumbent. Thus we proceed to branch on fractional component $\tilde{x}_2 = 0.3$.

Trying the $x_2 = 0$ side first, the relaxation is infeasible. Also, no new columns are able to enter in order to escape that result. We can terminate by infeasibility.

Proceeding to node 4 with $x_2 = 1$, we obtain a still fractional relaxation solution with bound = 130.55. This time, however, a new column 7 qualifies to enter. The resulting node 5 reduces the relaxation solution value to = 125.2. Still, it too is fractional and no additional columns are available to improve results.

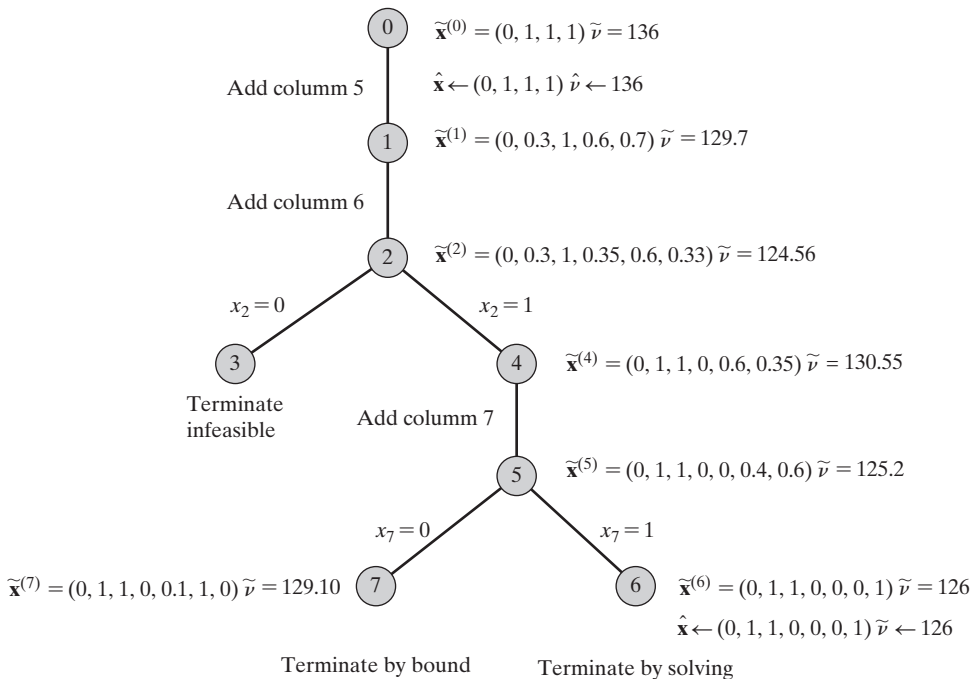


FIGURE 13.1 Branch and Price Tree of a Synthetic Example

We may conclude that $\tilde{\mathbf{x}}^{(5)}$ is optimal in the LP relaxation with x_2 fixed = 1, but we must branch on a fractional variable, this time $x_7 = 0.6$. Proceeding first to the nearest child with x_7 fixed = 1 yields integer-feasible solution $\tilde{\mathbf{x}}^{(6)} = (0, 1, 1, 0, 0, 0, 1)$ with value = 126. The solution improves on the current incumbent, so takes its place. Furthermore, since no additional columns are able to improve upon it, the node can be terminated by solving.

Finally, computation proceeds to node 7 with $x_7 = 0$ and a fractional relaxation solution with value = 129.10. Again, no additional columns are able to improve the relaxation. Thus, with $129.10 > \hat{v} = 126$, we may terminate this last active candidate problem by bound. The most recent incumbent is optimal.

13.2 LAGRANGIAN RELAXATION

Rather than directly seeking an optimal solution to a given ILP model, the **Lagrangian Relaxation** large-scale strategy uses decomposition to compute strong bounds on the value of an optimal solution to the full ILP. Sometimes the goal is to compute the linear-programming relaxation (definition 12.6) of a very large discrete model. Other times the decomposition can yield an even stronger bound than LP. Either way, the result can speed Branch and Bound/Cut Algorithms 12A and 12B, or provide a source for rounding to an approximate optimum in the full model (see Section 12.4).

Lagrangian Relaxations

Unlike LP relaxations that relax integrality requirements, Lagrangian Relaxations are formed by relaxing some of the main linear constraints of the given ILP, while keeping integrality requirements and the other main constraints. However, the relaxed constraints are not dropped entirely. Instead, they are **dualized** or weighted in the objective function with suitable **Lagrange multipliers** to discourage their violation.

Definition 13.3 | **Lagrangian relaxations** partially relax some of the main linear constraints of a given ILP by dualizing them in the objective function as terms

$$\dots + v_i \left(b_i - \sum_j a_{i,j} x_j \right) + \dots$$

Here v_i is a Lagrange multiplier on constraint i .

- If the relaxed constraint has form $\sum_j a_{i,j} x_j \geq b_i$, multiplier $v_i \leq 0$ for a maximize model and $v_i \geq 0$ for a minimize.
- If the relaxed constraint has form $\sum_j a_{i,j} x_j \leq b_i$ multiplier $v_i \geq 0$ for a maximize model and $v_i \leq 0$ for a minimize.
- Equality constraints $\sum_j a_{i,j} x_j = b_i$ have unrestricted multipliers v_i .

APPLICATION 13.2: LAGRANGIAN RELAXATION OF CDOT GENERALIZED ASSIGNMENT

We can illustrate with the CDOT generalized assignment model of Section 11.4:

$$\begin{aligned}
 \min \quad & 130x_{1,1} + 460x_{1,2} + 40x_{1,3} \\
 & + 30x_{2,1} + 150x_{2,2} + 370x_{2,3} \\
 & 510x_{3,1} + 20x_{3,2} + 120x_{3,3} \\
 & + 30x_{4,1} + 40x_{4,2} + 390x_{4,3} \\
 & 340x_{5,1} + 30x_{5,2} + 40x_{5,3} \\
 & + 20x_{6,1} + 450x_{6,2} + 30x_{6,3} \\
 \text{s.t.} \quad & x_{1,1} + x_{1,2} + x_{1,3} = 1 && \text{(district 1)} \\
 & x_{2,1} + x_{2,2} + x_{2,3} = 1 && \text{(district 2)} \\
 & x_{3,1} + x_{3,2} + x_{3,3} = 1 && \text{(district 3)} \\
 & x_{4,1} + x_{4,2} + x_{4,3} = 1 && \text{(district 4)} \\
 \text{(CDOT)} \quad & x_{5,1} + x_{5,2} + x_{5,3} = 1 && \text{(district 5)} \\
 & x_{6,1} + x_{6,2} + x_{6,3} = 1 && \text{(district 6)} \\
 & 30x_{1,1} + 50x_{2,1} + 10x_{3,1} && \text{(Estevan)} \\
 & + 11x_{4,1} + 13x_{5,1} + 9x_{6,1} \leq 50 \\
 & 10x_{1,2} + 20x_{2,2} + 60x_{3,2} && \text{(Mackenzie)} \\
 & + 10x_{4,2} + 10x_{5,2} + 17x_{6,2} \leq 50 \\
 & 70x_{1,3} + 10x_{2,3} + 10x_{3,3} && \text{(Skidgate)} \\
 & + 15x_{4,3} + 8x_{5,3} + 12x_{6,3} \leq 50 \\
 & x_{i,j} = 0 \text{ or } 1 \quad i = 1, 6; \quad j = 1, 3
 \end{aligned}$$

$$\text{where } x_{i,j} \begin{cases} 1 & \text{if district } i \text{ is assigned to ship } j \\ 0 & \text{otherwise} \end{cases}$$

An optimal solution assigns districts 1,4, and 6 to the Estevan, districts 2 and 4 to the Mackenzie, and district 3 to the Skidgate for a total cost of 480.

One strong Lagrangian relaxation keeps integrality requirements and the last 3 main constraints, while dualizing the first 6 with weights v_i to obtain

$$\begin{aligned}
 \min \quad & 130x_{1,1} + 460x_{1,2} + 40x_{1,3} + 30x_{2,1} + 150x_{2,2} + 370x_{2,3} \\
 & 510x_{3,1} + 20x_{3,2} + 120x_{3,3} + 30x_{4,1} + 40x_{4,2} + 390x_{4,3} \\
 & 340x_{5,1} + 30x_{5,2} + 40x_{5,3} + 20x_{6,1} + 450x_{6,2} + 30x_{6,3} \\
 \text{(CDOT}_v) \quad & + v_1(1 - x_{1,1} - x_{1,2} - x_{1,3}) + v_2(1 - x_{2,1} - x_{2,2} - x_{2,3}) \\
 & + v_3(1 - x_{3,1} - x_{3,2} - x_{3,3}) + v_4(1 - x_{4,1} - x_{4,2} - x_{4,3}) \\
 & + v_5(1 - x_{5,1} - x_{5,2} - x_{5,3}) + v_6(1 - x_{6,1} + x_{6,2} + x_{6,3}) \\
 \text{s.t.} \quad & 30x_{1,1} + 50x_{2,1} + 10x_{3,1} + 11x_{4,1} + 13x_{5,1} + 9x_{6,1} \leq 50 \\
 & 10x_{1,2} + 20x_{2,2} + 60x_{3,2} + 10x_{4,2} + 10x_{5,2} + 17x_{6,2} \leq 50 \\
 & 70x_{1,3} + 10x_{2,3} + 10x_{3,3} + 15x_{4,3} + 8x_{5,3} + 12x_{6,3} \leq 50 \\
 & x_{i,j} = 0 \text{ or } 1 \quad i = 1, 6; \quad j = 1, 3
 \end{aligned}$$

Notice that the 6 constraints of the full model have not been completely dropped. Instead, they have been rolled into the objective function as in definition [13.3](#). Being equalities, the corresponding v_i are *URS*. Feasible solutions to the relaxed subproblem ($CDOT_v$) may very well have say

$$x_{3,1} + x_{3,2} + x_{3,3} \neq 1 \text{ or equivalently, } (1 - x_{3,1} - x_{3,2} - x_{3,3}) \neq 0$$

Still, if chosen multiplier $v_3 \neq 0$, violations will at least affect the relaxation objective function.

EXAMPLE 13.2: FORMING LAGRANGIAN RELAXATIONS

Consider the ILP

$$\begin{aligned} \max \quad & 20x_1 + 30x_2 - 550y_1 - 720y_2 \\ \text{s.t.} \quad & 1.5x_1 + 4x_2 \leq 300 \\ & x_1 - 200y_1 \leq 0 \\ & x_2 - 75y_2 \leq 0 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 = 0 \text{ or } 1 \end{aligned}$$

- (a) Use multipliers v_1 and v_2 to form a Lagrangian relaxation dualizing the last two main constraints.
 (b) Indicate any required sign restrictions on choices of multipliers v_1 and v_2 .

Solution: We apply definition [13.3](#).

- (a) The required Lagrangian relaxation weights the two relaxed constraints in the objective function as

$$\begin{aligned} \max \quad & 20x_1 + 30x_2 - 550y_1 - 720y_2 + v_1(0 - x_1 + 200y_1) + v_2(0 - x_2 + 75y_2) \\ \text{s.t.} \quad & 1.5x_1 + 4x_2 \leq 300 \\ & x_1, x_2 \geq 0 \\ & y_1, y_2 = 0 \text{ or } 1 \end{aligned}$$

- (b) For these \leq constraints in a maximize model, multipliers should satisfy $v_1, v_2 \geq 0$.

Tractable Lagrangian Relaxations

Lagrangian relaxation ($CDOT_v$) keeps variables $x_{i,j}$ binary; integrality requirements of the full model ($CDOT$) have not been dropped. Instead, the improved tractability required of any useful relaxation is achieved by dualizing enough of the model's linear constraints to make the subproblems that remain significantly easier to solve.

Principle 13.4 Constraints chosen for dualization in Lagrangian relaxations should leave remaining subproblems with enough special structure to be relatively tractable even if they remain integer programs.

To see how the above relaxation $CDOT_v$ meets criterion [13.4](#), arbitrarily assign Lagrange multipliers $\hat{v}_1, \hat{v}_2, \hat{v}_3 \leftarrow -150$ and $\hat{v}_4, \hat{v}_5, \hat{v}_6 \leftarrow -90$. Then we may collect objective function terms involving each variable to obtain a single value. For example, the coefficient of $x_{1,1}$ becomes $130 - \hat{v}_1 = 130 - 150 = -20$. Repeating for all variables produces the relaxation

$$\begin{aligned}
 \min \quad & -20x_{1,1} - 120x_{2,1} + 360x_{3,1} + 120x_{4,1} + 430x_{5,1} + 110x_{6,1} \\
 & + 310x_{1,2} + 0x_{2,2} - 130x_{3,2} + 130x_{4,2} + 120x_{5,2} + 540x_{6,2} \\
 & - 110x_{1,3} + 220x_{2,3} - 30x_{3,3} + 480x_{4,3} + 130x_{5,3} + 120x_{6,3} \\
 (CDOT_{\hat{v}}) \quad & + 180 \\
 \text{s.t.} \quad & 30x_{1,1} + 50x_{2,1} + 10x_{3,1} + 11x_{4,1} + 13x_{5,1} + 9x_{6,1} \leq 50 \\
 & 10x_{1,2} + 20x_{2,2} + 60x_{3,2} + 10x_{4,2} + 10x_{5,2} + 17x_{6,2} \leq 50 \\
 & 70x_{1,3} + 10x_{2,3} + 10x_{3,3} + 15x_{4,3} + 8x_{5,3} + 12x_{6,3} \leq 50 \\
 & x_{i,j} = 0 \text{ or } 1 \quad i = 1, 6; \quad j = 1, 3
 \end{aligned}$$

The key insight is that each variable now appears only once in the objective and in only one constraint. Thus (except for the constant $\sum_i \hat{v}_i = 180$) relaxation $(CDOT_{\hat{v}})$ decomposes into 3 separate problems, each with a single main constraint.

Variables are still required to take on binary values, but these one-constraint ILPs take one of the simplest forms—Binary Knapsack Problems (definition [11.4](#)). It is then easy to see by inspection that the only nonzero variable values in the relaxation optimum are the ones with negative collected objective coefficients. In the above, they are $\hat{x}_{2,1} = 1$ and $\hat{x}_{3,3} = 1$, yielding objective value of $-120 - 30 + 180 = 30$. Notice also that these $\hat{x}_{i,j}$ satisfy the relaxed constraints for $i = 1, 3$ but leave all the others infeasible with no assignments.

More effort is required to solve the Knapsack Problems of the relaxation than would be needed if variables with negative coefficients competed for space in the retained knapsack constraints. Still, knapsack optima can be obtained for quite large instances (see, for example, Section 9.9). Repeated solution of relaxation subproblems is practical as part of a multiplier-driven Lagrangian decomposition.

Lagrangian Relaxation Bounds and Optima

Lagrangian relaxations fit within Section 12.2's discussion of ILP relaxations that modify the objective as well as the feasible set. Specifically principle [12.8](#) requires a valid relaxation to (i) admit as feasible every feasible solution to

the full model, and (ii) make the objective value in the relaxation of every such globally feasible solution equal to or better than its objective value in the full model.

Lagrangian definition [13.3] fulfills both requirements. Dropping explicit constraints in the relaxation cannot exclude feasible solutions of the full model. In the objective function, globally feasible solutions will satisfy all the relaxed constraints. Then careful examination of the sign rules on Lagrange multipliers shows that the added terms from dualization can only improve the objective value. For example, constraint $\sum_j a_{i,j}x_j \geq b_i$ of a minimize would have multiplier $v_i \geq 0$. Thus the corresponding Lagrangian objective function term for feasible $\hat{\mathbf{x}}$ would be the product

$$v_i \left(b_i - \sum_j a_{i,j} \hat{x}_j \right) \text{ which must be } \leq 0$$

for all feasible x_j . Dualized terms can only improve the minimize objective, which implies valid Lagrangian bounds.

Principle 13.5 The optimal solution value of any valid Lagrangian relaxation conforming to [13.3] is a lower bound on the optimal value of the underlying ILP if it minimizes, and an upper bound on the overall optimal value if it maximizes.

The minimizing ($CDOT_{\hat{v}}$) relaxation of Application 13.2 illustrates. Its optimal value of 30 is indeed a lower bound of the 480 optimal value for the full ($CDOT$) model.

Section 12.2 also explores cases where the relaxation optimum happens to prove optimal for the full model. Principles [12.11] and [12.12] showed that a relaxation optimum will be optimal in the full model if (i) it is feasible in the full model, and (ii) it has the same objective function value in both. Specialization to the Lagrangian case is as follows.

Principle 13.6 An optimal solution to a Lagrangian relaxation that is feasible for the full model and satisfies complementary slackness (see Section 6.7) for each dualized inequality constraint i so that either the associated multiplier $v_i = 0$ or the corresponding constraint is satisfied as equality is optimal in the full model.

Complementary slackness assures each dualized term = 0 so that original and relaxation objective values match—exactly as required to establish overall optimality of the relaxation optimum.

Notice the distinction from simpler relaxations that do not modify the objective. A Lagrangian relaxation optimum that is feasible in the full model may produce a useful incumbent, but principle [13.6] will establish its optimality in the full model only if needed complementarity on dualization terms is also met.

EXAMPLE 13.3: BOUNDS AND OPTIMA FROM LAGRANGIAN RELAXATIONS

Consider the ILP

$$\begin{array}{ll} \max & 5x_1 + 1x_2 + 4x_3 \\ \text{s.t.} & x_1 + 2x_2 \leq 2 \\ & \quad + 1x_2 + 1x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

- (a) Compute an optimal solution to the model by inspection.
- (b) Formulate a Lagrangian relaxation dualizing just the last main constraint with multiplier $\hat{v} \geq 0$.
- (c) Solve by inspection the relaxation of part (b) with $\hat{v} = 10$, and show that the result bounds the optimal solution value of the full model.
- (d) Solve by inspection the relaxation of part (b) with $\hat{v} = 3$, and use [13.6](#) to show that the relaxation optimum also solves the full model.

Solution:

- (a) By enumeration, an optimal solution is $\hat{\mathbf{x}} = (1, 0, 1)$ with objective value = 9.

(b)

$$\begin{array}{ll} \max & 5x_1 + (1 - \hat{v})x_2 + (4 - \hat{v})x_3 + \hat{v} \\ \text{s.t.} & x_1 + 2x_2 \leq 2 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{array}$$

- (c) With $\hat{v} = 10$, a relaxation optimum is $\hat{\mathbf{x}} = (1, 0, 0)$ and relaxation objective value = 15, which is a valid bound on the full optimum of 9 for this maximize instance.

- (d) With $\hat{v} = 3$, a relaxation optimum is $\hat{\mathbf{x}} = (1, 0, 1)$ with relaxation objective value = 9. To verify its overall optimality by [13.6](#), we first note that $\hat{\mathbf{x}}$ is feasible for the relaxed constraint. Furthermore, it satisfies the dualized constraint with equality, so that a $\hat{v} = 3 > 0$ conforms to the complementary slackness condition for optimality.

Lagrangian Duals

So far we have developed the idea of Lagrangian relaxations and explored some of their properties. Little has been said about how useful multipliers are chosen to empower the relaxations, and the limits of what can be achieved.

Principle 13.7 | The Lagrangian decomposition strategy for any dualization choice seeks to compute the best possible bound on the value of an optimal solution to a given ILP model by searching iteratively over possible choices of multipliers, solving the corresponding Lagrangian relaxations at each step, and learning from results how to improve multiplier choice or conclude significant further progress is unlikely.

Rather than treat all the possible permutations of min or max, and linear constraint forms, it will be useful to think most of the time about a single compact standard model to which all other cases can be reduced:

$$\begin{aligned}
 & \min \quad \mathbf{c}\mathbf{x} \\
 (P) \quad & \text{s.t.} \quad \mathbf{R}\mathbf{x} \geq \mathbf{r} \\
 & \quad \quad x \in T \triangleq \{ \mathbf{x} \geq \mathbf{0} : \mathbf{H}\mathbf{x} \geq \mathbf{h}, x_j \text{ integer for } j \in J \}
 \end{aligned}$$

Here \mathbf{x} is a vector of nonnegative decision variables, \mathbf{c} is the vector of cost coefficients, and J is the subset of decision variables required to take integer values. Systems $\mathbf{R}\mathbf{x} \geq \mathbf{r}$ and $\mathbf{H}\mathbf{x} \geq \mathbf{h}$ partition the main (linear) constraints. The first part is dualized and the second retained to complete the definition of the relatively tractable relaxation feasible set T .

The corresponding Lagrangian relaxation for multipliers $\mathbf{v} \geq 0$ is

$$\begin{aligned}
 & \min \quad \mathbf{c}\mathbf{x} + \mathbf{v}(\mathbf{r} - \mathbf{R}\mathbf{x}) \\
 (P_v) \quad & \text{s.t.} \quad \mathbf{x} \in \mathbf{T}
 \end{aligned}$$

Then the **Lagrangian Dual** computes multipliers producing the strongest possible bound from this relaxation in accord with goal [13.7](#).

Definition 13.8 For the primal (\mathbf{P}) and Lagrangian relaxation form (P_v) above, the corresponding Lagrangian Dual (D_L) seeks multipliers \mathbf{v} solving

$$\begin{aligned}
 & \max \quad \nu(P_v) \\
 (D_L) \quad & \text{s.t.} \quad \mathbf{v} \geq 0
 \end{aligned}$$

where $\nu(\cdot)$ denotes the value of an optimal solution to problem (\cdot) .

Some properties of these standard-form Lagrangian models follow immediately from principles [13.5](#) and [13.6](#) above.

Principle 13.9 For (P) , $(P_{\hat{\mathbf{v}}})$, and (D_L) above, $\nu(P) \geq \nu(D_L)$. Furthermore, if $\hat{\mathbf{x}}$ solves $(P_{\hat{\mathbf{v}}})$ for some $\hat{\mathbf{v}} \geq 0$, $\mathbf{R}\hat{\mathbf{x}} \geq \mathbf{r}$ and $\hat{\mathbf{v}}(\mathbf{r} - \mathbf{R}\hat{\mathbf{x}}) = 0$, then $\hat{\mathbf{x}}$ solves (\mathbf{P}) , and $\nu(P) = \nu(D_L)$.

With every $\mathbf{v} \geq 0$ providing a lower bound on $\nu(P)$, the strongest, which comes from $\nu(D_L)$ must too. Likewise, a relaxation optimum $\hat{\mathbf{x}}$ that satisfies dualized constraints and meets complementary slackness conditions between the constraints and their multipliers must be a (P) optimal, proving $\nu(P) = \nu(D_L)$.

APPLICATION 13.3: SMALL LAGRANGIAN NUMERICAL EXAMPLE

To better understand issues around Lagrangian duals and multiplier search, it will be helpful to work with an example taken from Parker and Rardin (1988), which is small enough to be addressed graphically. Consider the ILP

$$\begin{aligned}
 & \min && 3x_1 + 2x_2 \\
 & \text{s.t.} && 5x_1 + 2x_2 \geq 3 \\
 & && 2x_1 + 5x_2 \geq 3 \\
 (SMALL) & && 8x_1 + 8x_2 \geq 1 \\
 & && 0 \leq x_1 \leq 1 \\
 & && 0 \leq x_2 \leq 2 \\
 & && x_1, x_2 \text{ integer}
 \end{aligned}$$

Dualizing the first two linear constraints with multipliers $v_1, v_2 \geq 0$ gives Lagrangian relaxation:

$$\begin{aligned}
 (SMALL_{v_1, v_2}) & \min && 3x_1 + 2x_2 + v_1(3 - 5x_1 - 2x_2) + v_2(3 - 2x_1 - 5x_2) \\
 & \text{s.t.} && (x_1, x_2) \in T \triangleq \{ \text{integer } (x_1, x_2) \geq 0 : \\
 & && 8x_1 + 8x_2 \geq 1, x_1 \leq 1, x_2 \leq 2 \}
 \end{aligned}$$

The corresponding Lagrangian dual is:

$$\begin{aligned}
 (SDual) & \max && \nu(SMALL_{v_1, v_2}) \\
 & \text{s.t.} && v_1, v_2 \geq 0
 \end{aligned}$$

Figure 13.2(a) solves the original model (SMALL) and its LP relaxation graphically. The integer-feasible solutions are $(x_1, x_2) = (0, 2), (1, 1),$ and $(1, 2)$, with optimal choice $\mathbf{x}^* = (0, 2)$ at value $\nu^* = 4$. The LP-relaxation feasible space of the model is shaded in part (a), solving at $\bar{\mathbf{x}} = (3/7, 3/7)$ with value $\bar{\nu} = 15/7 \approx 2.14$.

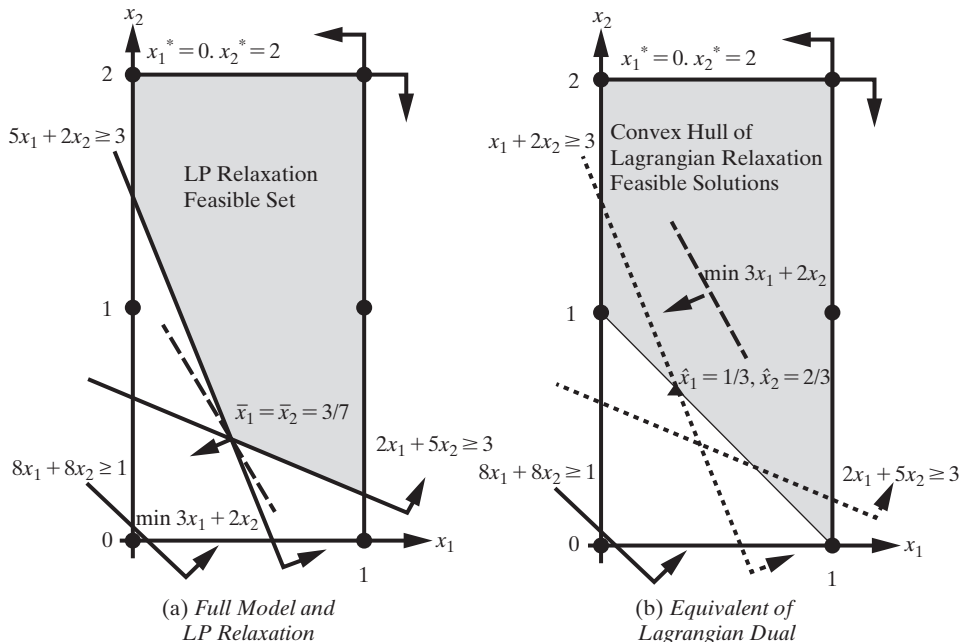


FIGURE 13.2 Graphic Solution of SMALL Lagrangian Example

With the first two linear constraints dualized (dotted lines in Figure 13.2(b)), Lagrangian relaxations are solved over feasible integer solutions meeting the still-enforced $8x_1 + 8x_2 \geq 1, 0 \leq x_1 \leq 1, \text{ and } 0 \leq x_2 \leq 2$.

$$T \triangleq \{(0, 1), (0, 2), (1, 0), (1, 1), (1, 2)\}$$

We know from Section 12.7 that any of the relaxations can be viewed as a linear program over their convex hull (the smallest polyhedral set containing all solutions in T). Shading in part (b) depicts the convex hull for this example, with optimum $\tilde{x} = (1/3, 2/3)$ and value = $7/3$.

Lagrangian versus Linear Programming Relaxation Bounds

The graphic solutions of Figure 13.2 demonstrate a useful formal characterization of what is achievable by solving the Lagrangian dual versus solving the linear programming relaxation of a given ILP.

Principle 13.10 For (P) , (P_v) , and (D_L) as defined above,

$$\nu(D_L) = \nu \left(\begin{array}{l} \min \quad \mathbf{c}\mathbf{x} \\ \mathbf{R} \geq \mathbf{r} \\ \mathbf{x} \in [T] \end{array} \right) \quad \text{and}$$

$$\nu(\bar{P}) = \nu \left(\begin{array}{l} \min \quad \mathbf{c}\mathbf{x} \\ \mathbf{R} \geq \mathbf{r} \\ \mathbf{x} \in T \end{array} \right)$$

where $[T]$ is the convex hull of T , (\bar{P}) is the LP relaxation of (P) , and \bar{T} is the LP relaxation of T .

The above characterization of LP relaxation bound $\nu(\bar{P})$ is no surprise. It simply reassembles all the original constraints after dropping integrality.

The characterization of Lagrangian dual bound $\nu(D_L)$ tells us more. What is achievable is exactly what would result from restricting an LP relaxation to dualized constraints and the convex hull $[T]$ of those maintained in Lagrangian relaxations.

Solution $\hat{\mathbf{x}} = (1/3, 2/3)$ at solution value $7/3 \approx 2.33$ illustrates in Figure 13.2(b). Notice that it still bounds integer optimal solution $\mathbf{x}^* = (0, 2)$ with value 4.00, confirming property [3.9]. Comparison with LP relaxation solution $\bar{\mathbf{x}} = (3/7, 3/7)$, which produces bound value $15/7 \approx 2.14$, establishes for this instance that $\nu(\bar{P}) < \nu(D_L)$; the Lagrangian strictly improves on the ordinary LP relaxation if an optimal choice or dual multipliers is used. This is no accident, because convex hull $[T]$ is always as tight as LP feasible \bar{T} and may be tighter.

Principle 13.11 For (P) , (P_v) , \bar{P} , and (D_L) above, $\nu(D_L) \geq \nu(\bar{P})$. Furthermore, if $[T] = \bar{T}$, that is, Lagrangian relaxations can be solved by linear programming, and $\nu(D_L) = \nu(\bar{P})$.

We see that the bound from a Lagrangian dual will always at least match that of the corresponding LP relaxation, but it can exceed it only if relaxations (P_v) are truly integer programs rather than being directly solvable as LPs. This leads to a refinement of Lagrangian tractability principle [13.4].

Principle 13.12 Per property [13.4], the choice of constraints to dualize in Lagrangian relaxations must leave subproblems significantly more tractable than the full model. Still, the Lagrangian dual bound for any dualization strategy cannot improve on the corresponding LP relaxation, unless subproblems cannot always be solved by LP alone.

Clearly, the choice of which constraints to dualize in Lagrangian relaxation is crucial. The generalized assignment model (CDOT) presented above illustrates the dilemma. We have seen that the Lagrangian relaxation $(CDOT_v)$ dualizing the $= 1$ constraints there decomposes into a series of Binary Knapsack Problems. Under principle [13.12], the extra integer-programming effort to solve them can (and does) yield a bound $\nu(D_L)$ stonger—sometimes much stonger—than the corresponding LP relaxation $(CDOT)$.

Contrast with the alternative $(CDOT_w)$ dualizing the \leq constraints with multipliers $w \leq 0$, while retaining the $= 1$ system:

$$\begin{aligned}
 \min \quad & 130x_{1,1} + 460x_{1,2} + 40x_{1,3} + 30x_{2,1} + 150x_{2,2} + 370x_{2,3} \\
 & 510x_{3,1} + 20x_{3,2} + 120x_{3,3} + 30x_{4,1} + 40x_{4,2} + 390x_{4,3} \\
 & 340x_{5,1} + 30x_{5,2} + 40x_{5,3} + 20x_{6,1} + 450x_{6,2} + 30x_{6,3} \\
 & +w_1(50 - 30x_{1,1} - 50x_{2,1} - 10x_{3,1} - 11x_{4,1} - 13x_{5,1} - 9x_{6,1}) \\
 & +w_2(50 - 10x_{1,2} - 20x_{2,2} - 60x_{3,2} - 10x_{4,2} - 10x_{5,2} - 17x_{6,2}) \\
 & +w_3(50 - 70x_{1,3} - 10x_{2,3} - 10x_{3,3} - 15x_{4,3} - 8x_{5,3} - 12x_{6,3}) \\
 \text{s.t.} \quad & x_{1,1} + x_{1,2} + x_{1,3} = 1 \\
 & x_{2,1} + x_{2,2} + x_{2,3} = 1 \\
 (CDOT_w) \quad & x_{3,1} + x_{3,2} + x_{3,3} = 1 \\
 & x_{4,1} + x_{4,2} + x_{4,3} = 1 \\
 & x_{5,1} + x_{5,2} + x_{5,3} = 1 \\
 & x_{6,1} + x_{6,2} + x_{6,3} = 1 \\
 & x_{i,j} = 0 \text{ or } 1 \quad i = 1, 6; \quad j = 1, 3
 \end{aligned}$$

As before, relaxations decompose into a series of single-constraint feasible sets such as

$$\begin{aligned}
 x_{1,1} + x_{1,2} + x_{1,3} &= 1 \\
 x_{1,1}, x_{1,2}, x_{1,3} &\geq 0 \text{ and integer}
 \end{aligned}$$

This time, however, each such problem can be solved as well by LP as IP. Whichever, of the variables in each system has the least objective function coefficient will $= 1$, and all the others will $= 0$, whether or not we explicitly enforce integrality. It follows (principle [13.12]) that even the best choice of multipliers w cannot produce a bound better than the LP relaxation of the full model.

Lagrangian Dual Objective Functions

The above graphic solution discussion provides intuition and insight. But as with most of the methods of this book, finding good Lagrangian dual multipliers for realistic applications typically requires a numerical search.

To simplify our discussion of search for Lagrange multipliers, assume that the feasible set T of relaxations (P_v) is bounded. Then, each relaxation optimum will occur at one of the finitely many extreme points of its convex hull. This leads to a useful insight into the form of the Lagrangian dual objective.

Principle 13.13 | If the feasible set T of a Lagrangian relaxation (P_v) is bounded, the relaxation optimal objective value $v(P_v)$ will be the piecewise-linear concave function of multipliers $\mathbf{v} \geq 0$ obtained as the minimum of linear objective functions of multipliers \mathbf{v} at alternative $\mathbf{x} \in T$.

Figure 13.3 illustrates why this must be true with the numerical example (*SMALL*) and its relaxation ($SMALL_{v_1, v_2}$) (and confirms the optimal dual solution value $7/3 \approx 2.33$ derived in Figure 13.2(b)). Given any nonnegative choice of the 2 multipliers v_1 and v_2 , computation of the relaxation optimum can be thought of as evaluating the resulting linear objective function at each of the 4 feasible solutions $\mathbf{x} \in T$, and then choosing the one with least objective value. For example, at $\mathbf{x} = (1, 2)$, the relaxation objective function in ($SMALL_{v_1, v_2}$) simplifies to the linear function

$$3(1) + 2(2) + v_1(3 - 5(1) - 2(2)) + v_2(3 - 2(1) - 5(2)) = 7 - 6v_1 - 9v_2$$

Each flat surface in the figure corresponds to the best choice at given values of multipliers \mathbf{v} . (There is no surface here for $\mathbf{x} = (1, 1)$ because it is dominated for all $\mathbf{v} \geq 0$.) That is, the solution value function is the minimum of linear functions in

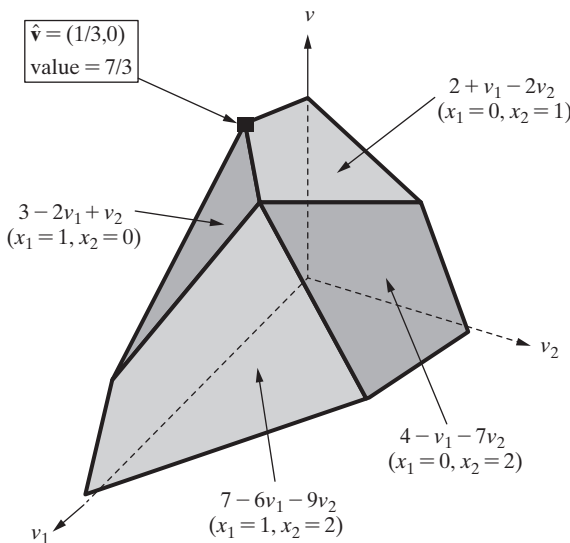


FIGURE 13.3 Dual Objective of *SMALL* Example

the dual multipliers. Its piecewise-linear shape follows from the fact that the best \mathbf{x} may be constant for a collection of \mathbf{v} s, then change abruptly when another one first matches and then surpasses the previous optimum. Linear boundaries between the surfaces identify \mathbf{v} for which there are alternative optima in the relaxation.

For the generic form (P) defined above, principle [13.13] establishes that dual objective functions $\nu(P_{\mathbf{v}})$ are always minima of linear functions in the multipliers, and thus concave (see definition [16.23]). Thus we can pursue an improving search such as those developed in Chapter 3 and 16 with confidence that when a local maximum is reached, that is, a \mathbf{v} that cannot be improved in its immediate neighborhood, then those multipliers are optimal for the Lagrangian dual (D_L) .

Subgradient Search for Lagrangian Bounds

A wide variety of improving search methods are known for at least approximately finding the best possible bounds for a given model and Lagrangian dualization strategy. We consider here only the simplest and most popular—**subgradient search**—which applies a generalization of steepest ascent **gradient—search** Algorithm 16D.

As usual, it will be helpful to gain insight from the Lagrangian dual of the simple numerical instance (*SMALL*) maximized graphically in Figure 13.3. Figure 13.4 shows a part of that dual surface projected from “above” to reduce it to 2 dimensions, including dual optimal $\hat{\mathbf{v}} = (0.333, 0.0)$.

Where a function is differentiable, the gradient (vector of partial derivatives) provides an improving direction in a maximize problem. For example, Point 1 = $(0.10, 0.50)$ in the figure is a solution for which the dual function is uniquely defined by surface $4 - v_1 - 7v_2$, because the corresponding relaxation $(P_{\mathbf{v}})$ has unique optimum $\mathbf{x} = (0.2)$. A gradient exists and yields the improving direction $\Delta\mathbf{v} (-1, -7)$ as shown.

The challenge in Lagrangian search arises at solutions like nearby Point 2 = $(0.262, 0.295)$, which lies on a boundary between piecewise linear segments of the dual function. There the relaxation has alternative optima $\mathbf{x} = (0, 1)$ and $\mathbf{x} = (0, 2)$.

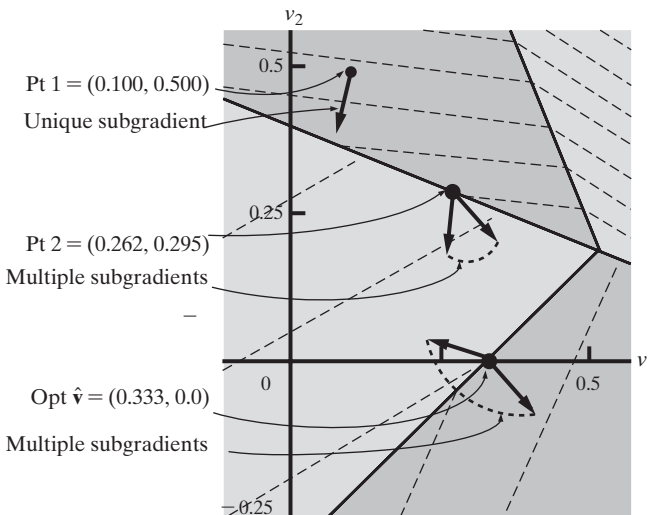


FIGURE 13.4 Lagrangian Dual Search of *SMALL* Numerical Example in 2D

Subgradients provide the generalization needed to deal with such cases by admitting convex combinations of all these alternatives.

Principle 13.14 For standard form (P) and (P_v) above, the subgradients of function $v(P_v)$ at a point v may be expressed as

$$\left\{ \Delta v = \sum_{\tilde{x} \in T_v} \alpha_{\tilde{x}} (\mathbf{r} - \mathbf{R}\tilde{x}) \text{ for } \alpha_{\tilde{x}} \geq 0 \text{ and } \sum \alpha_{\tilde{x}} = 1 \right\}$$

where $T_v \triangleq \{\tilde{x} \text{ optimal in } P_v\}$

We have already seen that at Point 1 in the figure, substitution of the unique optimum $(\tilde{x}_1, \tilde{x}_2) = (0, 2)$ gives improving direction $(\Delta v_1, \Delta v_2) = (-1, -7)$.

In the more complicated situation at figure Point 2, alternative relaxation optima $x = (0, 1)$, and $(0, 2)$, both with relaxation solution value = 1.673, lead to multiple subgradient directions.

$$\alpha_1(-2, 1) + \alpha_2(1, -2) \text{ for all } \alpha_1, \alpha_2 \geq 0, \alpha_1 + \alpha_2 = 1$$

All of the subgradients at Point 2 happen to be improving. Sadly, this will not always be the case when there are multiple subgradients at a point v . The range of subgradients at optimal point \hat{v} illustrate. They include $(1, -2)$, which decreases the objective, and $(-2, 1)$, which increases. Still, it can be shown that, at least for small steps, movement in any subgradient direction reduces the distance between a current solution and an optimum. Details of one step sequence known to still converge are given in the Step 4 of Algorithm 13C.

Algorithm 13C provides a full statement of subgradient search over Lagrangian duals. Before proceeding to the numerical example, it is worth noting some details of the algorithm. First, the search can pursue any subgradient defined in [13.14] at the current dual solution $v^{(\ell)}$. We adopt here the most readily available—dualized constraint “infeasibility” $(\mathbf{r} - \mathbf{R}x^{(\ell)})$ of the relaxation optimum.

Steps along subgradient directions (normalized to length = 1) are known to produce convergence if stepsizes converge to 0 but have sum that does not. Our example will use

$$\lambda_{\ell+1} \leftarrow 1/2(\ell + 1)$$

Stepizes λ_ℓ at Step 4 are chosen as if the optimization were unconstrained. Algorithm 13C Step 5 deals with the fact that the optimization is not strictly unconstrained because dual solutions may be subject to sign restrictions (here nonnegativity). However, it can be shown that the simplest possible correction for this keeps solutions nonnegative and maintains convergence; we have only to project (zero-out) any components of the solution becoming negative after the step.

Finally, there is no guarantee that subgradient search will produce monotonely improving dual solution values. Incumbents are tracked at Step 2 to record the best solution so far. When to satisfy that incumbent is an ad hoc decision. When the optimal $x^{(\ell)}$ of the Lagrangian relaxation happens to be feasible for dualized constraints, we can also track it as a primal incumbent.

ALGORITHM 13C: SUBGRADIENT LAGRANGIAN SEARCH

Let (P) , (P_v) , and (D_L) be as defined above.

Step 0: Initialization. Pick any starting dual solution $\mathbf{v}^{(0)} \geq 0$, begin $\ell \leftarrow 0$, and initialize primal and dual incumbent solution values $\hat{v}_D \leftarrow -\infty$ and $\hat{v}_P \leftarrow +\infty$.

Step 1: Lagrangian Relaxation. Solve Lagrangian Relaxation $(P_{\mathbf{v}^{(\ell)}})$ for relaxation optimum $\mathbf{x}^{(\ell)}$.

Step 2: Update Incumbents. If $v(P_{\mathbf{v}^{(\ell)}}) > \hat{v}_D$, update $\hat{v}_D \leftarrow v(P_{\mathbf{v}^{(\ell)}})$, and save $\hat{\mathbf{v}} \leftarrow \mathbf{v}^{(\ell)}$. Also if $(\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}) \leq 0$ and $\hat{v}_P < \mathbf{c}\mathbf{x}^{(\ell)}$, update $\hat{v}_P \leftarrow \mathbf{c}\mathbf{x}^{(\ell)}$.

Step 3: Stopping. If $(\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}) \leq 0$ and $\mathbf{v}^{(\ell)}(\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}) = 0$, stop; $\mathbf{x}^{(\ell)}$ solves primal problem (P) , and $v(D_L) = v(P)$. Otherwise, if further computation does not appear justified, stop and report incumbents \hat{v}_P , \hat{v}_D , $\hat{\mathbf{x}}$, and $\hat{\mathbf{v}}$ as approximate optima in (P) and (D_L) , respectively.

Step 4: Subgradient Step. Pick next stepsize $\lambda_{\ell+1} > 0$ from a sequence with $\lim_{\ell \rightarrow \infty} \lambda_{\ell} = 0$ and $\sum_{\ell=1}^{\infty} \lambda_{\ell} = \infty$. Then compute new dual solution

$$\mathbf{v}^{(\ell+1)} \leftarrow \mathbf{v}^{(\ell)} + \lambda_{\ell+1} \Delta \mathbf{v} \quad \text{where} \quad \Delta \mathbf{v} \leftarrow (\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}) / \|\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}\|$$

Step 5: Projection for Feasibility. Project the new dual solution on $\{\mathbf{v} \geq \mathbf{0}\}$ by setting

$$v_i^{(\ell+1)} \leftarrow \max \{0, v_i^{(\ell+1)}\} \quad \text{for all } i$$

Then advance $\ell \leftarrow \ell + 1$ and return to Step 1.

Application of Subgradient Search to Numerical Example

Table 13.4 and Figure 13.5 track application of Algorithm 13C on SMALL numerical example of (P) and (P_v) . The search begins at $\mathbf{v}^{(0)} = (0.50, 0.40)$, for which the relaxation solves at $\mathbf{x}^{(0)} = (1, 2)$ with objective value $v(P_{\mathbf{v}^{(0)}}) = 0.040$. These become the first incumbent dual solution. With $\mathbf{x} = (1, 2)$ feasible for both relaxed constraints, we can also save it as a primal incumbent.

Next we choose infeasibility vector $\mathbf{r} - \mathbf{R}\mathbf{x}^{(0)} = (-6, -9)$ as our first subgradient direction and normalize to length = 1 as $(-0.55, -0.83)$. A step of $\lambda_1 = 1/2$ advances the search to raw coordinates $\mathbf{v} = (0.22, -0.02)$. However, the second component is negative, so we project it to obtain next dual solution $\mathbf{v}^{(1)} = (0.22, 0.00)$.

TABLE 13.4 Progress of Algorithm 13C on SMALL Numerical Example

ℓ	$\mathbf{v}^{(\ell)}$	$\mathbf{x}^{(\ell)}$	$v(P_{\mathbf{v}^{(\ell)}})$	$\mathbf{r} - \mathbf{R}\mathbf{x}^{(\ell)}$	$\lambda_{\ell+1}$	Raw $\mathbf{v}^{(\ell+1)}$	$\hat{\mathbf{v}}$	\hat{v}_D	$\hat{\mathbf{x}}$	\hat{v}_P
0	(0.50, 0.40)	(1, 2)	0.40	(-6, -9)	1/2	(0.22, -0.02)	(0.50, 0.40)	0.40	(1, 2)	7
1	(0.22, 0.00)	(0, 1)	2.22	(1, -2)	1/4	(0.33, -0.22)	(0.22, 0.00)	2.22	(1, 2)	7
2	(0.33, 0.00)	(1, 0)	2.33	(-2, 1)	1/6	(0.19, 0.07)	(0.33, 0.00)	2.33	(1, 2)	7
3	(0.19, 0.07)	(0, 1)	2.04	(1, -2)	1/8	(0.24, -0.04)	(0.33, 0.00)	2.33	(1, 3)	7

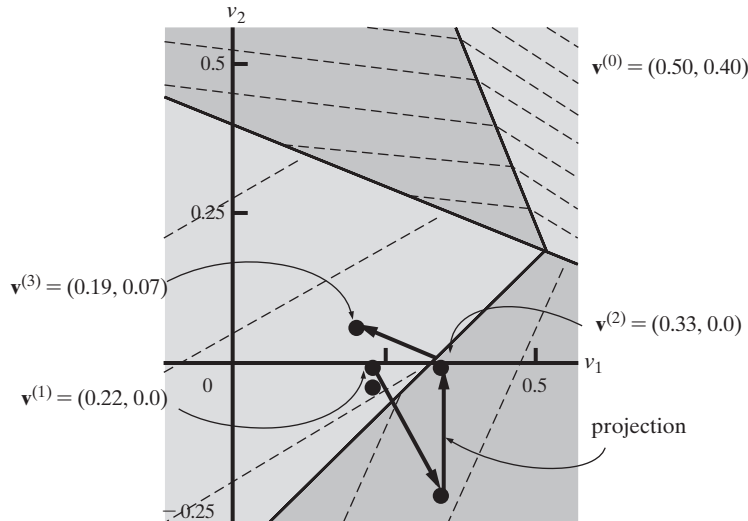


FIGURE 13.5 Progress of Subgradient Search on SMALL Numerical Example

Two further steps follow current value = 2.22 with dual solution values = 2.33, and 2.04, respectively. Not surprisingly, their solution values do not always improve. Although more iterations would be needed to assure convergence, we terminate this example upon the first decrease and report dual incumbent (0.33, 0.00) at value 2.33. From Figure 13.4 we know that this solution is optimal, but the subgradient search has established only that it may be an approximate optimum.

13.3 DANTZIG–WOLFE DECOMPOSITION

The Delayed Column Generation large-scale strategy of Section 13.1 is addressed to optimizations confronting combinatorially many decision options that can be represented by columns of a full formulation. Instead of explicitly addressing all the options at once, a partial master problem optimizes over just the subset of columns explicitly generated so far. The rest are delayed until identified by a column-generation subproblem as ones that can improve the current master problem solution. The overwhelming number of possible columns never need to be considered explicitly because the column-generation subproblem demonstrates that none not already considered can improve the current solution.

The **Dantzig–Wolfe decomposition** strategy of large-scale optimization, which is named after LP pioneers George Dantzig and Philip Wolfe (1960), adopts many of the same methods, but the target is different. The issue is not combinatorially many decision options that can be described by columns of a full optimization. Instead, we deal with very large linear programs by paralleling Lagrangian Relaxation of Section 13.2 in dividing the constraints as in Figure 13.6(a) between a modest-size collection of complicating constraints, and one or more LP subproblems with highly tractable structure such as network flows. Often the tractability of the subproblems stems from a **block-diagonal** collection of disjoint and

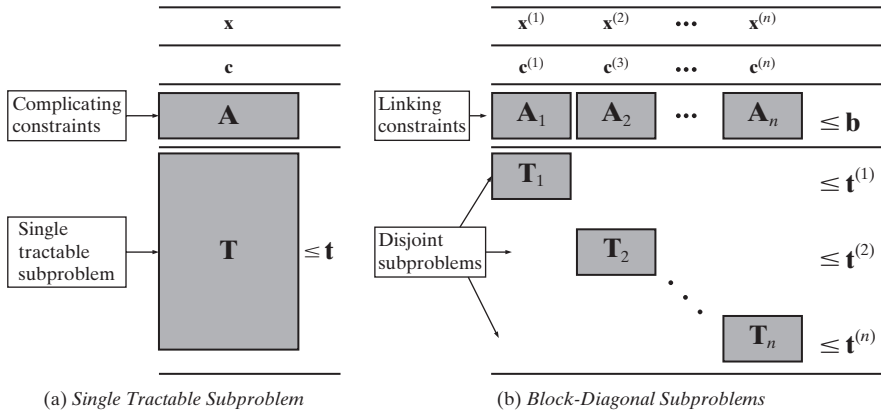


FIGURE 13.6 Structure of Attractive Dantzig–Wolfe Applications

smaller subproblems over subsets of the variables as illustrated in Figure 13.6(b). The subproblems may reflect parts of an organization, or time periods, or other modular elements that interact only through the set of **linking constraints**. The goal in all cases is to exploit the high tractability of subproblems by calling on them repeatedly while optimizing over a partial master problem of complicating/linking constraints.

As usual, it will be helpful to begin with an example application, albeit it tiny and fictional.

APPLICATION 13.4: GLOBAL BACKPACK (GB) NUMERICAL EXAMPLE

Consider a fictional company Global Backpack (GB) that manufactures and distributes school backpacks for children. GB has two sets of operations—one off-shore and one domestic—both with two production modes. The following table details parameters of the planning optimization they face in its block diagonal format:

	Off-shore		Domestic		
	x_1	x_2	x_3	x_4	
max	14	8	11	7	(profit)
s.t.	2.1	2.1	0.75	0.75	≤ 60 (shipping)
	0.5	0.5	0.5	0.5	≤ 25 (handling)
GBPdat	1	1			≥ 22 (site contract)
	1	0			≤ 20 (site mode 1 limit)
			1	1	≥ 12 (site contract)
			1	0	≤ 15 (site mode 1 limit)
		0	1	≤ 25 (site mode 2 limit)	

Decision variables x_1 and x_2 determine the number of bags (in thousands) produced, respectively, by each mode at the off-shore site. Variables x_3 and x_4 do the same for the domestic site.

Turning first to the subproblems, mode 1 production of bags at each site is subject to a limit because it uses components available at pre-contracted quantities

at a discounted price. Mode 2 is unlimited at the off-shore because it involves production with components bought more expensively on the open market. Mode 2 at the domestic site is limited by union agreements. Both sites are also subject to an agreed minimum production (in thousands) specified in contracts with their respective local government regulators.

For all sites and modes, GB has estimated a gross profit (\$ thousand per unit) reflecting the expected selling price less the cost of production. Profits are higher off-shore because of lower labor costs. The sum of all site and mode profits is to be maximized in the optimization.

Linking constraints show total shipping and handling budgets (in \$ thousand) provided by a separate subsidiary company for all GB sites and modes. As long as plans are within budget, no specific charge is billed to GB. Still, the budget \$-burdens per unit shown are higher for off-shore production because all finished goods must be shipped for sale in the domestic market.

Reformulation in Terms of Extreme Points and Extreme Directions

In the spirit of Delayed Column Generation we would like to focus on a partial master problem over just the linking constraints, with subproblems being called as needed to account for their separate constraint systems. Dantzig and Wolfe’s key insight is that this can be accomplished by exploiting as in (*) of Primer 3 the convex nature of subproblem feasible spaces. We represent values of the decision variables in the master problem as weighted sums of extreme points and extreme directions of the various subproblems.

Principle 13.15 A row-partitioned-form LP in the format of Figure 13.6(b), with master problem

$$\begin{aligned} \max \quad & \sum_s \mathbf{c}^{(s)} \mathbf{x}^{(s)} \\ \text{s.t.} \quad & \sum_s \mathbf{A}_s \mathbf{x}^{(s)} \leq \mathbf{b} \\ & \mathbf{x}^{(s)} \geq \mathbf{0} \quad \text{for all } s \end{aligned}$$

can be reformulated in terms of extreme points and extreme directions of subproblems s to capture the entire model as follows:

$$\begin{aligned} \max \quad & \sum_s \mathbf{c}^{(s)} \left(\sum_{j \in P_s} \lambda_{s,j} \mathbf{x}^{(s,j)} + \sum_{k \in D_s} \mu_{s,k} \Delta \mathbf{x}^{(s,k)} \right) \\ \text{s.t.} \quad & \sum_s \mathbf{A}_s \left(\sum_{j \in P_s} \lambda_{s,j} \mathbf{x}^{(s,j)} + \sum_{k \in D_s} \mu_{s,k} \Delta \mathbf{x}^{(s,k)} \right) \leq \mathbf{b} \\ & \sum_{j \in P_s} \lambda_{s,j} = \mathbf{1} \quad \text{for all } s \\ & \lambda_{s,j} \geq \mathbf{0} \quad \text{for all } s, j \in P_s, \mu_{s,k} \geq \mathbf{0} \quad \text{for all } s, k \in D_s \end{aligned}$$

where the P_s index the extreme points $\mathbf{x}^{(s,j)}$ and the D_s index the extreme directions $\Delta \mathbf{x}^{(s,k)}$ of subproblems s .

Reformulation from GB Application 13.4 Subproblems

Although typical applications would be much more complex, our GB Application 13.4 has been chosen so that both subproblems can be investigated graphically. Figure 13.7(a) shows that off-shore subproblem 1 has

$$\text{extreme points } \mathbf{x}^{(1,j)} = \begin{pmatrix} 0 \\ 22 \end{pmatrix}, \begin{pmatrix} 20 \\ 2 \end{pmatrix}$$

$$\text{and single extreme direction } \Delta \mathbf{x}^{(1,k)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Domestic subproblem 2 in part (b) has

$$\text{extreme points } \mathbf{x}^{(2,j)} = \begin{pmatrix} 0 \\ 12 \end{pmatrix}, \begin{pmatrix} 12 \\ 0 \end{pmatrix}, \begin{pmatrix} 15 \\ 0 \end{pmatrix}, \begin{pmatrix} 15 \\ 25 \end{pmatrix}, \begin{pmatrix} 0 \\ 25 \end{pmatrix}$$

and no extreme directions

All other points in either subproblem feasible set can be represented as in principle 13.15 and Primer 3. For example, depicted points $(x_1, x_2) = (20, 22)$ in subproblem 1 and $(x_3, x_4) = (15, 3, 4)$ in subproblem 2 can be expressed, respectively, as

$$\begin{pmatrix} 20 \\ 22 \end{pmatrix} \leftarrow \lambda_1 \begin{pmatrix} 20 \\ 2 \end{pmatrix} + \mu_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1 \begin{pmatrix} 20 \\ 2 \end{pmatrix} + 20 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} 15 \\ 3.4 \end{pmatrix} \leftarrow \lambda_1 \begin{pmatrix} 15 \\ 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} 15 \\ 25 \end{pmatrix} = 0.864 \begin{pmatrix} 15 \\ 0 \end{pmatrix} + 0.136 \begin{pmatrix} 15 \\ 25 \end{pmatrix}$$

Notice that in both subproblems, extreme-point weights λ_j are nonnegative and sum = 1 as required in the reformulation of 13.15. Extreme-direction weights need only be nonnegative.

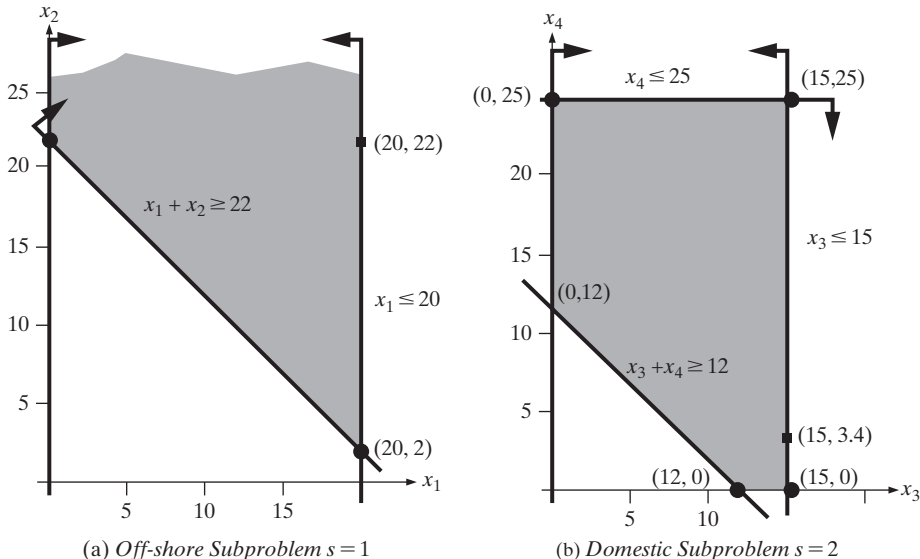


FIGURE 13.7 Subproblems of GB Application 13.4

Delayed Generation of Subproblem Extreme-Point and Extreme-Direction Columns

Although the full reformulation of principle [13.15] is completely equivalent to the original model, the numbers of extreme points and extreme directions it incorporates would be prohibitively huge for any real application. The heart of Dantzig–Wolfe is to realize that, just as with other models in Section 13.1, subproblem columns can be generated as needed in a partial master problem. Before presenting the full algorithm, we focus on the column-generation task, in principle [13.16].

Principle 13.16 To generate new improved columns for the reformulated master problem of [13.15], or prove that none exist, iterations ℓ of Dantzig–Wolfe decomposition attempt to solve LPs for each subproblem s of the following form:

$$\begin{aligned} \max \quad & \bar{\mathbf{c}}^{(s)} \mathbf{x}^{(s)} - \mathbf{q}^{(\ell)} \\ \text{s.t.} \quad & \mathbf{T}_s \mathbf{x}^{(s)} \leq \mathbf{t}^{(s)} \\ & \mathbf{x}^{(s)} \geq \mathbf{0} \end{aligned}$$

where $\bar{\mathbf{c}}^{(s)} \leftarrow (\mathbf{c}^{(s)} - \mathbf{v}^{(\ell)} \mathbf{A}_s)$, $\mathbf{v}^{(\ell)}$ is the optimal dual solution on linking constraints of the most recent partial master problem, and $\mathbf{q}_s^{(\ell)}$ is the corresponding dual value on the $= 1$ convexity constraint for subproblem s in that partial master problem.

We want to generate new columns that price out at favorable reduced costs in terms of the iteration ℓ partial master problem optimal dual solution $\mathbf{v}^{(\ell)}$ and $\mathbf{q}_s^{(\ell)}$. Objective function cost $\bar{\mathbf{c}}^{(s)} \leftarrow (\mathbf{c}^{(s)} - \mathbf{v}^{(\ell)} \mathbf{A}_s)$ builds in the part of that column pricing for linking constraints. Then, optimizing over subproblem variables $\mathbf{x}^{(s)}$ completes the task by optimizing the result over all possible feasible solutions to the subproblem. The term $-\mathbf{q}_s^{(\ell)}$ can be treated as constant in choosing an optimal extreme point or justifying unboundedness with an improving extreme direction (see Primer 3 and principle [5.27]). If an extreme point results, we have only to check that total column price including constant $-\mathbf{q}_s^{(\ell)}$ is positive for a maximize or negative for a minimize master problem.

One final observation is that processing of column generation subproblems is completely natural if some form of Simplex search (see Chapters 5–6 and Section 10.4) is applied.

Principle 13.17 If some form of Simplex search is applied to solve column generation subproblems [13.16], normal termination of any such algorithm produces exactly what is required to generate useful new columns. A finite optimum in simplex will be the best available extreme point to consider for addition to the master problem. Proof of unboundedness automatically yields the needed extreme direction for a new column.

We are now ready to proceed to a formal statement of the Dantzig–Wolfe method in the form of Algorithm 13D.

ALGORITHM 13D: DANTZIG-WOLFE DECOMPOSITION

Step 0: Initialization. Set iteration index $\ell \leftarrow 1$, and choose initial partial subproblem extreme point sets $P_{s,\ell} \subseteq P_s$ for all s such that the corresponding partial master version $\ell = 1$ of the formulation in [13.15] is feasible. Also initialize all partial extreme direction sets $D_{s,\ell} \leftarrow \emptyset$.

Step 1: Partial Master Solution. Solve partial master problem ℓ for reformulated primal optima $\lambda^{(s,\ell)}$, and $\mu^{(s,\ell)}$, along with corresponding dual optima $\mathbf{v}^{(\ell)}$ and $\mathbf{q}^{(\ell)}$.

Step 2: Column Generation Subproblems. Taking each subproblem s in turn, construct column generation LP of [13.16], and attempt to solve it. If the result is finite with solution value >0 for a maximize (<0 for minimize), return an extreme-point optimum $\bar{\mathbf{x}}^{(s)}$ (along with a $+1$ in the corresponding $= 1$ convexity constraint for s) as a new column in the next partial master problem. If the result is unbounded, return a corresponding extreme direction $\Delta \mathbf{x}^{(s)}$ as a new column. Otherwise, no additional columns from subproblem s can improve on the iteration ℓ partial master problem optimum.

Step 3: Stopping. If no additional columns were produced for any s in Step 2, stop. The most recent partial master problem ℓ solutions to primal and dual are optimal in the full model. Values for original primal variables can be recovered for all s by

$$\mathbf{x}^{(s)} \leftarrow \sum_{j \in P_{s,\ell}} \lambda_j^{(s,\ell)} \mathbf{x}^{(j)} + \sum_{k \in D_{s,\ell}} \mu_k^{(s,\ell)} \Delta \mathbf{x}^{(k)}$$

Otherwise update each $P_{s,\ell+1}$ and $D_{s,\ell+1}$ to include any new extreme-point and/or extreme-direction columns generated at Step 2. Then advance $\ell \leftarrow \ell + 1$, and return to Step 1.

Dantzig–Wolfe Solution of GB Application 13.4

Table 13.5 details application of Algorithm 13D to tiny GB Application 13.4.

Computation starts at iteration $\ell = 1$ with first sets of extreme points for the two subproblems that together yield a feasible starting partial master problem. Only one is needed from each subproblem to produce the subproblem optima shown for $\ell = 1$. Linking constraint duals are both $= 0$ because these particular extreme solutions leave slack in both main constraints.

Now consider the column generation subproblems at $\ell = 1$. With main duals $= 0$, $\bar{\mathbf{c}}^{(1)} = (14.0, 8.0)$, and $\bar{\mathbf{c}}^{(2)} = (11.0, 7.0)$, simply duplicate original objective function coefficients, but constants $q_1(1) = 176$ and $q_2^{(1)} = 84$. Both $\bar{\mathbf{c}}^{(s)}$ are > 0 in all components, meaning both subproblems seek large \mathbf{x} -values. It can be seen in Figure 13.7 that this yields an extreme direction $\Delta \mathbf{x}^{(1)} = (0, 1)$ in the first subproblem, which immediately leads to a new $D_{1,2}$ column in the next partial master problem. In subproblem 2, the largest extreme point $\bar{\mathbf{x}} = (15, 25)$ is optimal. Taking into account the constant $q_2^{(\ell)}$, total reduced cost for it has solution value $340 - 84 > 0$, qualifying the point to enter the next partial master problem set $P_{2,2}$.

Computation continues in the same way with iteration $\ell = 2$. This time linking constraint duals are not all $= 0$, and more complicated $\bar{\mathbf{c}}^{(1)} = (-11.60, -17.60)$

TABLE 13.5 Progress of Algorithm 13D on GB Application 13.4

ℓ	Partial Master Problem	Subproblem 1	Subproblem 2
0	Begin with extreme points $\bar{\mathbf{x}}^{(1)} = (0, 22), \bar{\mathbf{x}}^{(2)} = (0, 12)$		
1	$\lambda^{(1,1)} = (1), \lambda^{(2,1)} = (1)$ $\mu^{(1)} = \text{none}$ $\mathbf{v}^{(1)} = (0, 0), \mathbf{q}^{(1)} = (176, 84)$ partial master objval = 260.0 implied $\mathbf{x}^{(1,1)} = (0, 22), \mathbf{x}^{(2,1)} = (0, 12)$	$\bar{\mathbf{c}}^{(1)} = (14.0, 8.0)$ $\Delta \mathbf{x}^{(1)} = (0, 1)$ unbounded	$\bar{\mathbf{c}}^{(2)} = (11.0, 7.0)$ $\bar{\mathbf{x}}^{(2)} = (15, 25)$ objval = 256.0
2	$\lambda^{(2,1)} = (1, 0), \lambda^{(2,2)} = (0.771, 0.229)$ $\mu^{(1)} = (0)$ $\mathbf{v}^{(2)} = (12.191, 0), \mathbf{q}^{(2)} = (-387.2, -25.71)$ partial master objval = 318.51 implied $\mathbf{x}^{(1,2)} = (0, 22), \mathbf{x}^{(2,2)} = (3.43, 14.97)$	$\bar{\mathbf{c}}^{(1)} = (-11.60, -17.60)$ $\bar{\mathbf{x}}^{(1)} = (20, 2)$ objval = 120.0	$\bar{\mathbf{c}}^{(2)} = (1.857, -2.143)$ $\bar{\mathbf{x}}^{(2)} = (15, 0)$ objval = 53.57
3	$\lambda^{(1,3)} = (0, 1), \lambda^{(2,3)} = (0, 0.136, 0.864)$ $\mu^{(1)} = (0)$ $\mathbf{v}^{(3)} = (9.333, 0), \mathbf{q}^{(3)} = (-135.2, 60.0)$ partial master objval = 484.8 implied $\mathbf{x}^{(1,3)} = (20, 2), \mathbf{x}^{(2,3)} = (15, 3.4)$	$\bar{\mathbf{c}}^{(1)} = (-5.599, -11.599)$ $\bar{\mathbf{x}}^{(1)} = (20, 2)$ objval = 0.00	$\bar{\mathbf{c}}^{(2)} = (4.000, 0.000)$ $\bar{\mathbf{x}}^{(2)} = (15, 0)$ objval = 0.00

and $\bar{\mathbf{c}}^{(2)} = (1.857, -2.143)$ result. Each produces a new extreme point qualifying for the next partial master problem when constants $q_1^{(\ell)} = -135.2$ and $q_2^{(\ell)}$ are taken into account.

Finally, in iteration $\ell = 3$, updated $\bar{\mathbf{c}}^{(1)} = (-5.599, -11.599)$ and $\bar{\mathbf{c}}^{(2)} = (4.000, 0.000)$ along with constants $\mathbf{q}^{(3)} = (-135.2, 60)$. With those parameter values, both column generation subproblems produce optima with solution value = 0.0. We may conclude that no further columns are needed, and the most recent partial master problem is optimal.

For insight, the implied values of original variables $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are reported in Table 13.5 at every iteration ℓ . However, it is only necessary to recover the final optima when algorithm Step 3 terminates. Here this yields $\mathbf{x}^{(1)} = (20, 2)$, and $\mathbf{x}^{(2)} = (15, 3.4)$ with overall objective function value = 484.8. Notice in Figure 13.7(b) that the optimal solution for subproblem 2 is nonextreme-point solution (15, 3.4).

13.4 BENDERS DECOMPOSITION

Section 13.2 described the **Lagrangian Relaxation** large-scale strategy that formed more tractable subproblems by relaxing some of the main linear constraints of the given ILP. The relaxed constraints are not dropped entirely, but instead, they are weighted in the objective function with suitable dual multipliers to discourage their violation.

Benders Decomposition is the complementary strategy that fixes decision variable values on challenging columns to form more tractable LP subproblems over the rest. Then, like all the other methods of this chapter, a partial master problem processes subproblem results to either conclude that the current solution is optimal or pick a new set of fixed values on challenging variables to feed to the next subproblem. Most applications address mixed-integer linear programs (MILPs), treating their integer variables as the challenges to be fixed, and the continuous ones as the more tractable.

Principle 13.18 Benders decomposition is most attractive for MILP models taking the form

$$\begin{aligned}
 & \min \quad \mathbf{cx} + \mathbf{fy} \\
 (BP) \quad & \text{s.t.} \quad \mathbf{Ax} + \mathbf{Fy} \geq \mathbf{b} \\
 & \quad \quad \mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0} \text{ and integer}
 \end{aligned}$$

where a linear program results if values of integer variables \mathbf{y} are fixed.

APPLICATION 13.5: HEART GUARDIAN FACILITIES LOCATION

We can illustrate this with a small **uncapacitated facilities location problem** such as those of Section 11.6. The Heart Guardian (HG) corporation is establishing a logistic network to distribute its new single line of monitoring devices across 5 markets. Distribution centers from which trucks will deliver the monitors will need to be established at any or all of 3 already selected sites. Demands in each market (thousands), unit shipping costs from each site to each market (\$), and setup costs (\$ thousand) to establish each site have been estimated as shown in the following table.

		Markets					
		$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	
Demand d_j		75	90	81	26	57	
GHdat	Setup f_i	Unit Transportation Cost c_{ij}					
	Site $i = 1$	400	4	7	3	12	15
	Site $i = 2$	250	13	11	17	9	19
	Site $i = 3$	300	8	12	10	7	5

HG wishes to develop a minimum cost plan over these parameters.

Using decision variables $x_{ij} \triangleq$ the number (thousands) of units shipped from site i to market j and $y_i \triangleq 1$ if site i is opened and $= 0$ otherwise, the HG task can be modeled

$$\begin{aligned}
 (HG) \quad & \min \quad \sum_{ij} c_{ij} x_{ij} + \sum_i f_i y_i \\
 & \text{s.t.} \quad \sum_j x_{ij} - d_{sum} y_i \leq 0 \quad \text{for all } i \\
 & \quad \quad \sum_i x_{ij} = d_j \quad \text{for all } j \\
 & \quad \quad x_{ij} \geq 0 \quad \text{for all } i, j \\
 & \quad \quad y_i = 0 \text{ or } 1 \quad \text{for all } i
 \end{aligned}$$

With distribution site capacities treated as unlimited, constant $d_{sum} \triangleq \sum_j d_j$ is employed in the first system to implement the needed switching constraints.

Under the usual assumption that numbers of units shipped can be taken as continuous, this model (HG) partitions naturally into continuous and integer (binary) parts as in [13.18]. Once values for binary variable y_i have been fixed, the remaining LP over x_{ij} is a highly tractable transportation problem.

Benders Decomposition Strategy

The Benders method for models (BP) in [13.18](#) begins at each iteration ℓ by fixing a choice of integer variables at some $\mathbf{y}^{(\ell)}$, and reducing the formulation to one over continuous variables alone:

$$\begin{aligned}
 \min \quad & \mathbf{c}\mathbf{x} + \mathbf{f}\mathbf{y}^{(\ell)} \\
 (BP_\ell) \quad & \text{s.t.} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{F}\mathbf{y}^{(\ell)} \\
 & \mathbf{x} \geq \mathbf{0}
 \end{aligned}$$

The dual of $(BP_{\mathbf{y}^{(\ell)}})$ produces the **Benders subproblem** at the heart of each iteration ℓ .

Definition 13.19 Using dual variables \mathbf{v} on rows of primal models (BP) and $(BP_{\mathbf{y}^{(\ell)}})$ yields the dual subproblem.

$$\begin{aligned}
 \max \quad & \mathbf{v}(\mathbf{b} - \mathbf{F}\mathbf{y}^{(\ell)}) + \mathbf{f}\mathbf{y}^{(\ell)} \\
 (BD_\ell) \quad & \text{s.t.} \quad \mathbf{v}\mathbf{A} \leq \mathbf{c} \\
 & \mathbf{v} \geq \mathbf{0}
 \end{aligned}$$

For simplicity, we will assume that the given instance of (BP) has a finite optimum. This assures that reduced primal (BP_ℓ) will also have a finite optimum for at least an optimal $\mathbf{y}^{(\ell)}$. Then dual subproblem (BD_ℓ) is also feasible, and since the constraints are the same for all subproblems, every one must be feasible.

This leaves two kinds of outcomes that may result from attempting to solve subproblems $(BD_{\mathbf{y}^{(\ell)}})$. First, the subproblem may yield an extreme point optimal solution $\mathbf{v}^{(\ell)}$. Alternatively, the subproblem may prove unbounded along an extreme direction $\Delta \mathbf{v}^{(\ell)}$. The **Benders master problem** accounts for both of these subproblem possibilities in a formulation to decide the best choice of integer variables \mathbf{y} .

Definition 13.20 The complete Benders master problem (BM) optimizes variables \mathbf{y} over all possible subproblem extreme-point outcomes $\mathbf{v}^{(i)}$, $i \in P$, and extreme-direction outcomes $\Delta \mathbf{v}^{(j)}$, $j \in D$ as

$$\begin{aligned}
 \min \quad & z \\
 (BM) \quad & \text{s.t.} \quad z \geq \mathbf{f}\mathbf{y} + \mathbf{v}^{(i)}(\mathbf{b} - \mathbf{F}\mathbf{y}) \quad \text{for all } i \in P \\
 & \mathbf{0} \geq \Delta \mathbf{v}^{(j)}(\mathbf{b} - \mathbf{F}\mathbf{y}) \quad \text{for all } j \in D \\
 & \mathbf{y} \geq \mathbf{0} \text{ and integer}
 \end{aligned}$$

Analogously to the column-based reformulation of the Dantzig-Wolfe method in Section 13.3, Benders produces this row-based (BM) reformulation taking advantage of the fact that all Benders subproblems [13.19](#) for any model (BP) have the same constraints. The extreme-point system of constraints in (BM) assures that an optimal solution \mathbf{y} must equal or exceed its valuation at any subproblem extreme point by listing all the possibilities. On the other hand, infeasible \mathbf{y} lead to unbounded subproblems along some extreme direction. The second system of constraints in [13.20](#) assures that no such \mathbf{y} will be feasible in (BM) .

We can conclude that full formulation (BM) does compute an optimal \mathbf{y} . To be sure, there are likely to be exponentially many extreme points in full list P and perhaps numerous extreme directions in full list D . Still, as with all the other methods of this chapter, we can work with a much smaller partial master problem over just subproblem results derived so far.

Principle 13.21 Each iteration ℓ of Benders decomposition solves, at least approximately, a partial version (BM_ℓ) of full master problem (BM):

$$\begin{aligned}
 & \mathbf{min} \quad z \\
 (BM_\ell) \quad & \text{s.t.} \quad z \geq \mathbf{f}\mathbf{y} + \mathbf{v}^{(i)}(\mathbf{b} - \mathbf{F}\mathbf{y}) \quad \text{for all } i \in P_\ell \\
 & \quad \quad \mathbf{0} \geq \Delta\mathbf{v}^{(j)}(\mathbf{b} - \mathbf{F}\mathbf{y}) \quad \text{for all } j \in D_\ell \\
 & \quad \quad \mathbf{y} \geq \mathbf{0} \text{ and integer}
 \end{aligned}$$

where P_ℓ indexes any extreme-point subproblem solutions generated in iterations $1, \dots, \ell$ and D_ℓ is the corresponding index set of generated extreme directions.

We are now ready to state the full row-generating Benders Decomposition Algorithm 13E.

Optimality in Benders Algorithm 13E

One remaining issue is when can Benders Algorithm 13E end its sequence of solving subproblems and corresponding partial master problems, that is, when has an optimal solution been reached? Algorithm Step 2 shows the test.

Principle 13.22 If subproblem ℓ of Algorithm 13E produces an extreme-point solution $\mathbf{v}^{(\ell)}$ with subproblem optimal value $v(BD_\ell) \leq$ previous partial master problem optimal solution value $v(BM_{\ell-1})$, then $\mathbf{v}^{(\ell)}$ is optimal in both full master problem (BM) and given model (BP).

This criterion holds because every subproblem optimal value $v(BD_\ell)$ is an upper bound on $v(BP)$ for fixed $\mathbf{y}^{(\ell)}$ because (BD_ℓ) is a restriction of (BM) . On the other hand, most recent partial master problem solution value $v(BM_{\ell-1})$ is a lower bound because it is a relaxation of full master problem (BM) , which is equivalent to (BP) . Thus if criterion [13.22](#) is fulfilled, $v(BD_\ell) = v(BP) = v(BM)$ and $\mathbf{v}^{(\ell)}$ is optimal in all 3.

It is important to note that the test of [13.22](#) addresses provable optimal value $v(BD_{\ell-1})$ of the most recent partial master problem, not the $z_{\ell-1}$ that is part of that formulation. Partial master problems are integer programs, and thus it is often desirable to solve most of them only approximately. Algorithm 13E can continue adding new rows and computing bound values z_ℓ as long as some care is taken to avoid duplication. In particular, all that is needed is to generate extreme-point solutions $\mathbf{y}^{(\ell)}$ and/or extreme directions $\Delta\mathbf{v}^{(\ell)}$ not satisfied by the most recent partial master problem solution. Such new constraints will have to change the next partial master problem result. Still full optimality can be concluded only if, at some point, a partial master problem is solved to a provable optimum, and criterion [13.22](#) is satisfied.

ALGORITHM 13E: BENDERS DECOMPOSITION

Let (BP) , (BP_ℓ) , (BD_ℓ) , and (BM_ℓ) be as defined above.

Step 0: Initialization. Initialize extreme-point set $P_0 \leftarrow \emptyset$, extreme-direction set $D_0 \leftarrow \emptyset$, prior master problem solution value $z_0 = -\infty$, and pick any starting $\mathbf{y}^{(0)}$ consistent with formulation (BP) . Also, begin iteration counter $\ell \leftarrow 1$.

Step 1: Benders Subproblem. Attempt to solve Benders subproblem $(BD_{\mathbf{y}^{(\ell-1)}})$. If the result is a finite optimum at extreme point $\mathbf{v}^{(\ell)}$, go to Step 2, and if it is an extreme direction $\Delta \mathbf{v}^{(\ell)}$, go to Step 3.

Step 2: Stopping. If subproblem solution value $\mathbf{f}\mathbf{y}^{(\ell)} + \mathbf{v}^{(\ell)}(\mathbf{b} - \mathbf{F}\mathbf{y}^{(\ell)}) \leq$ the optimal solution value of previous partial master problem $v(BM_{\ell-1})$, stop; $\mathbf{y}^{(\ell)}$ is optimal in full model (BP) , and the corresponding continuous part of the optimal solution $\mathbf{x}^{(\ell)}$ can be obtained by solving reduced primal $(BP_{\mathbf{y}^{(\ell)}})$.

Step 3: Partial Master Problem Update. If an extreme point resulted at Step 2, include the new point by $P_\ell \leftarrow P_{\ell-1} \cup \ell$ in partial master problem (BM_ℓ) with new constraint

$$z \geq \mathbf{f}\mathbf{y} + \mathbf{v}^{(\ell)}(\mathbf{b} - \mathbf{F}\mathbf{y})$$

and retain $D_\ell \leftarrow D_{\ell-1}$. Otherwise, if an extreme direction resulted at Step 1, include the new direction by $D_\ell \leftarrow D_{\ell-1} \cup \ell$ in the partial master problem (BM_ℓ) with new constraint

$$0 \geq \Delta \mathbf{v}^{(\ell)}(\mathbf{b} - \mathbf{F}\mathbf{y})$$

and retain $P_\ell \leftarrow P_{\ell-1}$.

Step 4: Benders Partial Master Problem. Solve, at least approximately, master problem (BM_ℓ) for solution $\mathbf{y}^{(\ell)}$ and z_ℓ . Then advance $\ell \leftarrow \ell + 1$ and return to Step 1.

Solution of Heart Guardian Application 13.5 with Benders

Algorithm 13E

To apply Algorithm 13E to Heart Guardian facilities location 13.5, we begin by organizing its parameters in the format of primal subproblem (BP_ℓ) . Taking first the continuous, LP part, $\mathbf{c}\mathbf{x}$ and $\mathbf{A}\mathbf{x}$ can be arranged as follows:

$\mathbf{x} =$		x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	
$\mathbf{c} =$		4	7	3	12	15	13	11	17	9	19	8	12	10	7	5	
$\mathbf{A} =$	$i = 1$	1	1	1	1	1											
	$i = 2$						1	1	1	1	1						
	$i = 3$											1	1	1	1	1	
	$j = 1$	1					1					1					
	$j = 2$		1					1					1				
	$j = 3$			1					1					1			
	$j = 4$				1					1						1	
	$j = 5$					1					1						1

Next, we consider the right-hand side, integer parts \mathbf{fy} and $(\mathbf{b} - \mathbf{Fy})$. Then arrange as follows:

$\mathbf{y} =$		y_1	y_2	y_3
$\mathbf{f} =$		400	250	300
$i = 1$	\leq	0	329	
$i = 2$	\leq	0	329	
$i = 3$	\leq	0		329
$j = 1$	$=$	75		
$j = 2$	$=$	90		
$j = 3$	$=$	81		
$j = 4$	$=$	261		
$j = 5$	$=$	57		

Together, and using dual variables v_i on the three i -rows, and w_j on the five j rows, these lead to Benders subproblem:

	y_1	y_2	y_3	v_1	v_2	v_3	w_1	w_2	w_3	w_4	w_5	
max	400	250	300	$329y_1$	$329y_2$	$329y_3$	75	90	81	26	57	
s.t.				1			1					≤ 4
				1				1				≤ 7
				1					1			≤ 3
				1						1		≤ 12
				1							1	≤ 15
					1		1					≤ 13
					1			1				≤ 11
					1				1			≤ 17
					1					1		≤ 9
					1						1	≤ 19
						1	1					≤ 8
						1		1				≤ 12
						1			1			≤ 10
						1				1		≤ 7
						1					1	≤ 5
	y_i fixed			$v_i \leq 0$			w_j URS					

Notice that the variable types on dual variables have been adjusted for the \leq format of capacity constraints, and the $=$ demands.

Table 13.6 traces the progress of Algorithm 13E on these formulations. We begin somewhat arbitrarily with initial $\mathbf{y}^{(0)} = (0, 0, 0)$. It is easy to see that this choice makes subproblem (BD_1) unbounded. Extreme direction $\Delta\mathbf{v}^{(1)} = (-1, -1, -1)$ and $\Delta\mathbf{w}^{(1)} = (1, 1, 1, 1, 1)$ produces the first constraint of partial master (BM_1) (see parameters in the above table):

$$0 \geq \Delta\mathbf{v}^{(1)}(329y_1, 329y_2, 329y_3) + \Delta\mathbf{w}^{(1)}(75, 90, 81, 26, 57) \text{ or}$$

$$0 \geq -329y_1 - 329y_2 - 329y_3 + 329$$

which simply says at least one y_i must be positive in any feasible solution.

TABLE 13.6 Progress of Benders Algorithm 13E on Heart Guardian Application

ℓ	Benders Subproblem	Partial Master Problem
0		$\mathbf{y}^{(0)} = (0, 0, 0)$
1	$\Delta \mathbf{v}^{(1)} = (-1, -1, -1)$ $\Delta \mathbf{w}^{(1)} = (1, 1, 1, 1, 1)$ objval = $+\infty$	new row: $0 \geq -329y_1 - 329y_2 - 329y_3 + 329$ $\mathbf{y}^{(1)} = (1, 0, 0)$ objval indeterminant
2	$\mathbf{v}^{(2)} = (0, -3, -10)$ $\mathbf{w}^{(2)} = (4, 7, 3, 12, 15)$ objval = 2740	new row: $z \geq 400y_1 - 737y_2 - 2990y_3 + 2340$ $\mathbf{y}^{(2)} = (0, 1, 1)$ objval = -1387
3	$\mathbf{v}^{(3)} = (-7, 0, 0)$ $\mathbf{w}^{(3)} = (8, 11, 10, 7, 5)$ objval = 3417	new row: $z \geq -2003y_1 + 250y_2 + 300y_3 + 2867$ $\mathbf{y}^{(3)} = (1, 0, 1)$ objval = 1264
4	$\mathbf{v}^{(4)} = (0, 0, 0)$ $\mathbf{w}^{(4)} = (4, 7, 3, 7, 5)$ objval = 2340	new row: $z \geq 400y_1 + 250y_2 + 300y_3 + 2867$ $\mathbf{y}^{(4)} = (1, 1, 0)$ objval = 2290
5	$\mathbf{v}^{(5)} = (0, 0, -10)$ $\mathbf{w}^{(5)} = (4, 7, 3, 9, 15)$ objval = 2912	new row: $z \geq 400y_1 + 250y_2 - 2990y_3 + 2262$ $\mathbf{y}^{(5)} = (1, 0, 1)$ objval = 2340
6	objval = 2340	

Lacking any more guidance, we arbitrarily fix next $\mathbf{y}^{(1)} = (1, 0, 0)$. Then, subproblem (BD_2) yields an extreme-point optimum at $\mathbf{v}^{(2)} = (0, -3, -10)$, $\mathbf{w}^{(2)} = (4, 7, 3, 12, 15)$, and objective value = 2740. The implied new master problem constraint is

$$z \geq 400y_1 + 250y_2 + 300y_3 + \mathbf{v}^{(2)}(329y_1, 329y_2, 329y_3) + \mathbf{w}^{(2)}(75, 90, 81, 26, 57) \text{ or}$$

$$z \geq 400y_1 - 737y_2 - 2990y_3 + 2340$$

Now, the partial master problem (BM_2) solves at $\mathbf{y}^{(2)} = (0, 1, 1)$ and optimal value = -1387.

Continuing in this way, we solve subproblems for partial master problem optima $\mathbf{y}^{(2)}$, $\mathbf{y}^{(3)} = (1, 0, 1)$, and $\mathbf{y}^{(4)} = (1, 1, 0)$. When the constraint for subproblem (BD_5) is added to the partial master, however, repeat optimum $\mathbf{y}^{(5)} = (1, 0, 1)$ occurs at objective value = 2340. We already know that the subproblem at $\ell = 6$ will produce the same optimum as $\ell = 3$ for the same \mathbf{y} . Thus means criterion 13.22 has been fulfilled, and the algorithm has reached optimality at solution:

$$\mathbf{y}^* = (1, 0, 1) \text{ and from corresponding } (BP_{\mathbf{y}^*})$$

$$\mathbf{x}^* = (75, 90, 81, 0, 0 | 0, 0, 0, 0 | 0, 0, 0, 26, 57)$$

$$\text{optimal value} = 2340$$

EXERCISES

13-1 Consider solving an ILP of the following form by Delayed Column Generation Algorithm 13A.

$$\begin{aligned} \min \quad & \sum_j c_j x_j \\ \text{s.t.} \quad & \sum_j a_{i,j} x_j \geq b_i \quad i = 1, \dots, 5 \\ & \text{all } x_j \geq 0 \text{ and integer} \end{aligned}$$

where right-hand sides are given positive integers, $b_i \leq 5$, coefficients $a_{i,j}$ in any column j are non-negative integers summing to at most 5, and corresponding cost $c_j \leftarrow \sum_{i=1}^5 \log_{10}(a_{i,j} + 1)$.

- ✔ (a) Specify an initial partial problem using only columns j with a single $a_{i,j} \neq 0$ that admits a feasible primal solution.
- ✔ (b) Justify why it is appropriate to solve LP relaxations of each partial problem encountered by Algorithm 13A instead of requiring integer values of decision variables prior to overall algorithm termination.
- ✔ (c) Now suppose a partial problem is at hand with columns $j \in \bar{J}$, and its LP relaxation is solved to obtain optimal dual solution $\bar{v}_1, \dots, \bar{v}_5$. Formulate a INLP column generation subproblem model that seeks an attractive next new column g to enter (if there is one) with coefficients $a_{i,g}$ and cost c_g conforming to the rules above.
- ✔ (d) Explain why a generated column g that qualifies to enter the partial problem cannot already be one of the columns in \bar{J} .
- ✔ (e) Explain how subproblem generation in (c) will also detect when the algorithm should stop because no further columns qualify to enter.

13-2 Emergency relief agency ERNow is planning flights of small helicopters to deliver medical, food, and housing supplies for populations cutoff by a recent hurricane. The following table shows the fraction of each plane’s weight w_i and volume v_i capacity of shipping containers for different materials to be sent, along with the number that must be transported.

<i>i</i>	Material	Weight Fraction w_i	Volume Fraction v_i	Quantity Needed q_i
1	First aid supplies	0.04	0.10	30
2	Drinking water	0.20	0.14	20
3	Diesel Generators	0.40	0.24	12
4	Generator fuel	0.28	0.32	23
5	Tents	0.10	0.28	15
6	Cots	0.16	0.24	30
7	Blankets	0.03	0.18	40
8	Rain capes	0.08	0.14	25

ERNow wants to meet all these needs with the minimum number of flights.

- (a) Formulate ERNow’s challenge as an ILP over decision variables $x_j \triangleq$ the number of times load combination j is used, with columns for load combinations made up of $a_{i,j} \triangleq$ the number of containers of material i carried in each load j .
- (b) Discuss how the large number of feasible load mixes make Delayed Column Generation Algorithm 13A attractive for this application.
- (c) Show that the weight and volume constraints column coefficients $a_{i,j}$ are required to satisfy in terms of parameters w_i and v_i .
- (d) Construct an initial set of columns j for a first partial problem as ones for “pure” loads with $a_{i,j} = 0$ except on one product i where the maximum feasible number of that product is specified.
- (e) Justify why it is appropriate to solve LP relaxations of each partial problem encountered instead of requiring integer values of decision variables prior to algorithm termination.
- (f) Suppose now that the LP relaxation of a partial problem is solved as the algorithm proceeds and yields optimal dual values \bar{w}_i on product rows i . Use those results to formulate a column generation

subproblem to find a feasible new load mix g with coefficients $a_{g,i}$ having a reduced cost in the LP relaxation of the current partial problem that makes it attractive to enter.

- (g) What outcome from solving your generation subproblems of (f) would justify terminating Algorithm 13A? Explain.

- (h) Use class optimization software to solve the LP relaxation of the partial problem of (d). Then solve the resulting column generation subproblem (f) by inspection and continue Algorithm 13A until 5 new

columns have been added, or the termination condition of (g) is met.

- (i) Round up to derive an approximate integer optimal solution from the final results of part (h).

13-3 The Silo State IE faculty is dividing its 12 members into 4 teams of 3 to develop ideas for a long-term strategic plan. Like all faculties, the professors are not all equally compatible with each other. The table below provides a compatibility score (0 bad to 100 good) for each possible pair.

	Prof 2	Prof 3	Prof 4	Prof 5	Prof 6	Prof 7	Prof 8	Prof 9	Prof 10	Prof 11	Prof 12
Prof 1	21	61	99	54	75	27	64	70	99	58	88
Prof 2		45	14	81	55	78	73	58	5	13	9
Prof 3			44	27	41	31	53	3	36	0	10
Prof 4				11	3	46	29	26	30	6	11
Prof 5					43	56	52	93	73	0	87
Prof 6						14	53	34	92	13	18
Prof 7							38	27	68	27	18
Prof 8								7	49	15	94
Prof 9									35	32	53
Prof 10										8	12
Prof 11											10

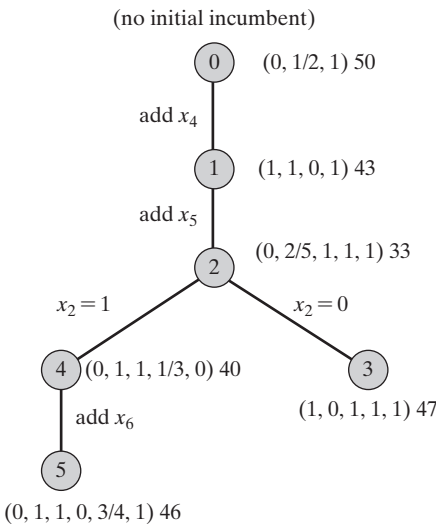
- (a) Formulate this team formation task as a binary ILP over professors $i = 1 \dots, 12$, and decision variables $x_j = 1$ if possible 3-prof team $T_j \triangleq \{i_1, i_2, i_3\}$ is included in the plan, and $= 0$ otherwise. The faculty seeks a maximum total compatibility collection of the 4 chosen teams, where the compatibility value of each possible team T_j denoted h_j , is the sum of the 3 table values above for pairs of its members.
- (b) How many columns (distinct teams) would the full formulation of (a) include?

- (c) Discuss the challenges in direct solution of the full model, especially if the faculty became much larger.
- (d) Explain the possible advantages of approaching the faculty's task indirectly with Delayed Column Generation Algorithm 13A, starting with a partial master problem including just the columns for teams of profs 1-3, 4-6, 7-9, and 10-12.

- (e) Use class optimization software to compute primal and dual optimal solutions to the LP relaxation of this first partial problem.

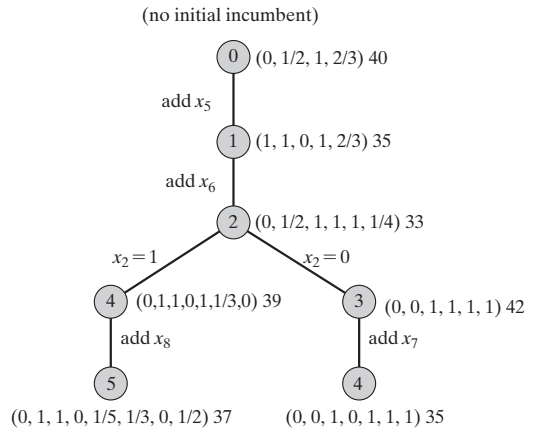
- (f) Outline an adhoc column generation subproblem using the dual LP optimum of (e) to produce additional columns that could improve the solution.
- (g) Manually apply your subproblem design of (f) to generate 3 new columns likely to improve the solution of (e), and justify your choices in terms of reduced objective values.
- (h) Add your new columns of (g) to the partial problem and re-solve its LP relaxation. Did the solution improve? Why or why not?
- (i) Do you think your solution of (h) is close to optimal? Explain.

13-4 The tree below shows the hypothetical evolution of a Branch and Price Algorithm 13B solution of an all-binary ILP in original variables $x_1, x_2,$ and x_3 . Relaxation solutions are shown next to nodes, and both fixed variables and new columns (variables) are identified on the branches.



- ✓ (a) Is the model minimizing or maximizing? Explain.
- ✓ (b) Assuming the process correctly followed Branch and Price Algorithm 13B, take the nodes in numerical order and briefly describe what apparently happened. Also, explain what solution proved optimal and why.

13-5 Do Exercise 13-4 for the Branch-and-Price below on an all-binary ILP over original variables x_1, x_2, x_3 and x_4 .



13-6 Consider the ILP

$$\begin{aligned} \max \quad & 30x_1 + 55x_2 + 20x_3 \\ \text{s.t.} \quad & 40x_1 - 12x_2 + 11x_3 \leq 55 \\ & 19x_1 + 60x_2 + 3x_3 \geq 20 \\ & 3x_1 + 2x_2 + 2x_3 = 5 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

Form the Lagrangian relaxations obtained by dualizing each of the following collections of main constraints, and show all sign restrictions that apply to Lagrange multipliers.

- ✓ (a) Dualize that first and second main constraints.
- (b) Dualize the first and third main constraints.

13-7 Consider the facilities location ILP

$$\begin{aligned} \min \quad & 3x_{1,1} + 6x_{1,2} + 5x_{2,1} + 2x_{2,2} \\ & + 250y_1 + 300y_2 \\ \text{s.t.} \quad & 30x_{1,1} + 20x_{1,2} \leq 30y_1 \\ & 30x_{2,1} + 20x_{2,2} \leq 50y_2 \\ & x_{1,1} + x_{2,1} = 1 \\ & x_{1,2} + x_{2,2} = 1 \\ & 0 \leq x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \leq 1 \\ & y_1 = 0 \text{ or } 1 \end{aligned}$$

- ✓ (a) Use total enumeration to compute all optimal solution.
- ✓ (b) Form a Lagrangian relaxation dualizing the third and fourth constraints with Lagrange multipliers v_1 and v_2 .

- ✔ (c) Explain how the dualization in part (b) leaves a relaxation that is easier to solve than the full ILP.
- ✔ (d) Use total enumeration to solve the Lagrangian relaxation of part (b) with $v_1 = v_2 = 0$, and verify that the relaxation optimal value provides a lower bound on the true optimal value computed in part (a).
- ✔ (e) Repeat part (d) with $v_1 = v_2 = -100$.
- ✔ (f) Repeat part (d) with $v_1 = 1000, v_2 = 500$.

13-8 Add to Exercise 13-7(a)-(f) the following:

- ✔ (g) Compute the subgradient direction of Algorithm 13C at your solution of (f) and take a step using $\lambda = 500$ to update the dual solution. Then re-solve the Lagrangian relaxation. Did the value improve?
- ✔ (h) Repeat (g) by taking a second step from the solution of (f) using stepsize $\lambda = 250$ to update the dual solution, and comment on the result.
- ✔ (i) Is it possible that the dualization of (b) can produce a bound better than the LP relaxation of the full model? Explain.
- (j) Now formulate an alternative Lagrangian relaxation dualizing the first two main constraints and keeping the 3rd and 4th.
- (k) Can that new dualization ever produce a bound better than the LP relaxation of the full model? Explain.

13-9 Do Exercise 13-7 for the generalized assignment model

$$\begin{aligned} \min \quad & 15x_{1,1} + 10x_{1,2} + 30x_{2,1} + 20x_{2,2} \\ \text{s.t.} \quad & x_{1,1} + x_{1,2} = 1 \\ & x_{2,1} + x_{2,2} = 1 \\ & 30x_{1,1} + 50x_{2,1} \leq 80 \\ & 30x_{1,2} + 50x_{2,2} \leq 60 \\ & x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} = 0 \text{ or } 1 \end{aligned}$$

Dualize the first two main constraints and solve with $\mathbf{v} = (0, 0)$, $\mathbf{v} = (10, 12)$, and $\mathbf{v} = (100, 200)$.

13-10 Consider the binary integer program

$$\begin{aligned} \min \quad & 5x_1 - 2x_2 \\ \text{s.t.} \quad & 7x_1 - x_2 \geq 5 \\ & x_1, x_2 \text{ binary} \end{aligned}$$

- (a) Formulate and justify a Lagrangian relaxation dualizing the only main constraint with multiplier v .

- (b) Briefly explain why your relaxation of (a) is indeed a relaxation of the full ILP model.
- (c) Comment on whether this relaxation can always be solved by LP ([13.11](#)).
- (d) There are four possible integer solutions to the relaxation. Develop four (linear) expressions for the relaxation objective function in terms of multiplier v , assuming in turn that each of the four is optimal.
- (e) Use results of (d) to develop a plot for the Lagrangian relaxation dual solution value as a function of multiplier v .
- (f) Identify an optimal choice of v on your plot of (e), and explain your choice.

13-11 Consider the following binary ILP:

$$\begin{aligned} \max \quad & 13x_1 + 22x_2 + 18x_3 + 17x_4 + 11x_5 + 19x_6 + 25x_7 \\ \text{s.t.} \quad & \sum_{j=1}^7 x_j \leq 3 \\ & 3x_1 + 7x_2 + 5x_3 \leq 15 \\ & 12x_4 + 9x_5 + 8x_6 + 6x_7 \leq 11 \\ & x_j \text{ binary, } j = 1, \dots, 7 \end{aligned}$$

- (a) State the corresponding Lagrangian relaxation dualizing only the first main constraint, and initializing its dual multiplier at $v = 20$ or $v = -20$, whichever is appropriate.
- (b) Briefly justify why your Lagrangian of (a) is indeed a relaxation of the original model.
- (c) Solve the relaxation of (a) by inspection, using multiplier $v = 10$ and identify an optimal solution and solution value for it. Justify your computations.
- (d) Beginning from your results of part (c), compute the subgradient direction Algorithm 13C would follow.

13-12 Consider the tiny LP constraint set

$$\begin{aligned} -w_1 + w_2 &\leq 1 \\ w_1 + 2w_2 &\geq 1 \\ w_1, w_2 &\geq 0 \end{aligned}$$

- (a) Sketch the feasible set of these constraints in a 2-dimensional plot.
- ✔ (b) Establish that the feasible set is convex.
- ✔ (c) List all the extreme points of the set.
- ✔ (d) List all the extreme directions.
- ✔ (e) Show that each of the following points can be expressed as a convex combination of the extreme points plus a nonnegative

combination of the extreme directions: $\mathbf{w}^{(1)} = (0, 1)$, $\mathbf{w}^{(2)} = (1, 1)$, $\mathbf{w}^{(3)} = (0, 1/2)$.

- ✓ (f) Is there any feasible point (w_1, w_2) for these constraints that cannot be expressed as a convex combination of the extreme points plus a nonnegative combination of the extreme directions? Explain.

13-13 Consider solving the following 4-variable LP by Dantzig-Wolfe Decomposition Algorithm 13D. Constraints 1 and 2 will constitute the linking problem, numbers 3-5 the first subproblem, and numbers 6-8 the second.

max	52x₁	+ 19x₂	+ 41x₃	+ 9x₄	
s.t.	12x₁	+ 20x₂	+ 15x₃	+ 8x₄	≤ 90
	9x₁	+ 10x₂	+ 13x₃	+ 18x₄	≤ 180
	+ x ₁				≤ 7
	+ x ₁	- x ₂			≥ 3
	x _{1, x₂} ≥ 0				
			+ 1x ₃		≤ 8
			+ 1x ₃	+ 1x ₄	≥ 10
			x _{3, x₄} ≥ 0		

- ✓ (a) Sketch the feasible set of the constraints for the first subproblem in a 2-dimensional plot, and establish that the feasible set is convex.
- ✓ (b) Identify all the extreme points and extreme directions of that first subproblem feasible set.
- ✓ (c) Do (a) for the second subproblem.
- ✓ (d) Do (b) for the second subproblem.
- ✓ (e) Show that Algorithm 13D can begin with extreme point $(x_1, x_2) = (0, 0)$ in the first subproblem, and $(x_3, x_4) = (0, 10)$ in the second subproblem. And construct the first partial master problem.
- ✓ (f) Solve your partial master problem of (e) with class optimization software.
- ✓ (g) Use results of part (f) to construct objective functions for the two subproblems and solve both graphically.
- ✓ (h) Update the partial master with results from the two subproblems.

13-14 Do Exercise 13-13 on the following LP starting with extreme points $(x_1, x_2) = (0, 0)$ and $(x_3, x_4) = (36, 0)$ in part (e).

min	200x₁	+ 350x₂	+ 450x₃	+ 220x₄	
s.t.	2x₁	+ 1x₂	+ 4x₃	+ 3x₄	≥ 100
	5x₁	+ 5x₂	+ 9x₃	+ 7x₄	≥ 177
	+x ₁				≤ 5
	+5x ₁	+4x ₂			≤ 32
	x _{1, x₂} ≥ 0				
			+1x ₃	+4x ₄	≤ 36
				+1x ₄	≤ 4
			x _{3, x₄} ≥ 0		

13-15 Consider solving the following MILP by Benders Decomposition Algorithm 13E. Treat the y variables as the complicating ones, and begin with $\mathbf{y}^{(0)} = (0, 0)$.

$$\begin{aligned}
 \max \quad & 60x_1 + 50x_2 - 25y_1 - 100y_2 \\
 \text{s.t.} \quad & 20x_1 + 17x_2 - 60y_1 - 30y_2 \leq 10 \\
 & 11x_1 + 13x_2 - 30y_1 - 60y_2 \leq 10 \\
 & x_1, x_2 \geq 0; \\
 & y_1, y_2 \in [0, 10] \text{ and integer}
 \end{aligned}$$

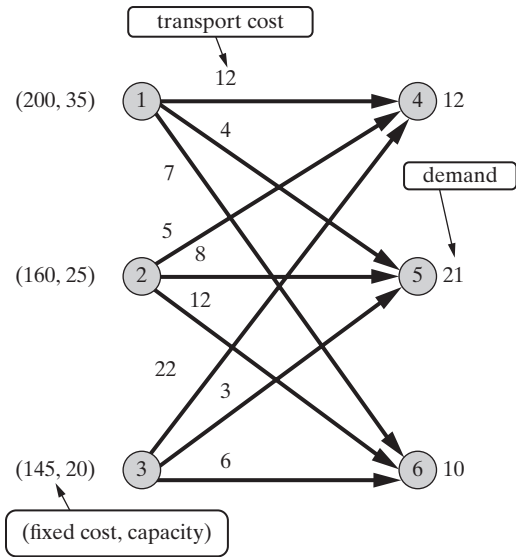
- ✓ (a) Treating y variables as the complicating ones, formulate the corresponding

Benders Primal and Benders Dual subproblems (definition 13.19).

- ✓ (b) Starting with all $y_i = 0$, optimize the instance by Algorithm 13E. Use class optimization software to solve sub- and master problems when they become inconvenient for solution by inspection.

13-16 Consider solving the the following minimum total cost facilities location model with Benders Decomposition Algorithm 13E, taking numbers on supply nodes as the fixed cost and capacity (if opened), those on demand nodes as

the required inflow, and numbers on arcs as unit costs of transportation.



- (a) Use nonnegative variables $x_{i,j}$ for flows on arcs (i, j) and binary variables $y_i = 1$ if supply i is opened ($= 0$ otherwise) to formulate the instance as a capacitated MILP parallel to the uncapacitated one of BG Application 13.5.
- (b) Treating y variables as the complicating ones, formulate the corresponding Benders Primal and Benders Dual subproblems (definitions [13.18] and [13.19]).
- (c) Starting with all $y_i = 1$, optimize the instance by Algorithm 13E. Be sure to provide details of sub and master problems solved at each step. Use class optimization software to solve sub and master problems when they become inconvenient for solution by inspection.

REFERENCES

Bertsimas, Dimitris and John N. Tsitsiklis (1997), *Introduction to Linear Optimization*. Athena Scientific, Nashua, New Hampshire.

Chvátal, Vašek (1980), *Linear Programming*, W.H. Freeman, San Francisco, California.

Lasdon, Leon S. (1970), *Optimization Theory for Large Systems*, Macmillan, London, England.

Martin, R. Kipp (1999), *Large Scale Linear and Integer Optimization*, Kluwer Academic, Boston, Massachusetts.

Parker, R. Gary and Ronald L. Rardin (1988), *Discrete Optimization*, Academic Press, San Diego, California.

Wolsey, Laurence (1998), *Integer Programming*, John Wiley, New York, New York.

Computational Complexity Theory

Prior chapters of this book have encountered many optimization model forms and the algorithms that address them. These include Linear Programs, Integer Linear Programs, Network Flow Problems, Shortest Path Problems, and Dynamic Programs. Some of these were seen to be highly tractable, that is, very large examples can be solved to global optimality. Others seemed dramatically harder to address.

Computational Complexity Theory seeks to define a rigorous and consistent way of thinking about how the tractability of problem forms should be classified and how algorithmic efficiency should be measured. This chapter presents the central concepts of that theory and illustrates their value in conducting optimization research and practice.

14.1 PROBLEMS, INSTANCES, AND THE CHALLENGE

Our investigation begins with a formal definition of an optimization problem.

Definition 14.1 In complexity theory, a **problem** is a general model form defined as an infinite collection of particular data sets termed **instances**.

Thus, although we sometimes informally call an example like the following a “problem”

$$\begin{array}{ll} \min & 2x_1 + 4x_2 \\ \text{s.t.} & 12x_1 + x_2 \geq 29 \\ & x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0 \end{array}$$

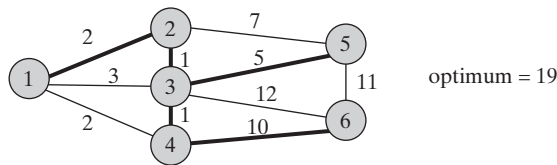
it is more properly considered one instance of the LP problem form characterized by its specific decision variables, objective function, and constraints.

The Challenge

Researchers or analysts confronted with instances of some problem form seek the most efficient algorithms for addressing the models before them. The challenge is to decide what sorts of methods to investigate. How can we identify some problems that are quickly and easily solved with clever special techniques, while much less efficient, or even heuristic methods are the best available tools for others? Two tree problems on graphs will illustrate.

APPLICATION 14.1: SPANNING TREE EXAMPLES OF THE COMPLEXITY CHALLENGE

We consider two similar problems defined on undirected graphs with weights/costs assigned to each edge. One instance is shown below.



An edge subgraph of any given instance is a **tree** if it is connected and contains no cycles, and it is a **spanning tree** if it includes all nodes. Heavy edges in the above offer an example.

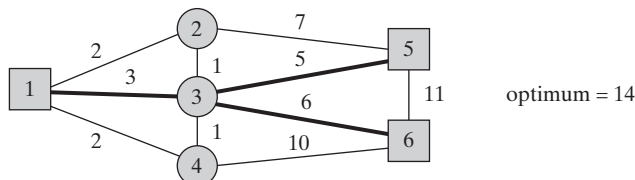
Minimum Spanning Tree Problem

The **Minimum Spanning Tree (MST)** problem treated in Section 10.10 seeks a minimum total weight spanning tree. The heavy edges in the above application define an optimum with total weight = 19.

As explained in Section 10.10, an extremely efficient “greedy” algorithm is available to compute an optimal spanning tree of any given instance. We simply select edges in lowest-to-highest weight sequence, skipping any which would create a cycle with those already chosen, and stopping when a spanning tree is a hand. For the above instance, this process could first choose weight = 1 edges (2, 3) and (3, 4), then choose weight = 2 edge (1, 2). Weight = 2 edge (1, 4) must now be skipped because it forms a cycle with the other three. Continuing, we skip weight = 3 edge (1, 3) because it too forms a cycle. Now weight = 5 edge (3, 5) can be added, but weight = 7 edge (2, 5) then creates a cycle. Adding weight = 10 edge (4, 6) completes a spanning tree. A solution obtained this way is provably optimal.

Minimum Steiner Tree Problem

An apparently very similar problem is the **Minimum Steiner Tree (Stein)** problem with instances like the following.



The new element is a subset of **Steiner nodes** (shown as squares in the above diagram). The minimum Steiner tree problem seeks a minimum total weight tree that spans all the Steiner nodes, but others may or may not be included. The above figure identifies an optimal solution in heavy edges, with total weight 14. All 3 Steiner nodes (1, 5, and 6) are included along with optional node 3.

What is startling about this Steiner variant of spanning tree problems is that despite its close resemblance to the (**MST**), the only known algorithms for computing an optimum are standard Integer Linear Programming methods of Chapter 12. Enumerative methods like Branch and Bound or Branch and Cut are the best tools available.

Prior to the full discovery of complexity theory for optimization problems in the 1970s, researchers and analysts had only their experience and intuition to decide what methods to pursue on a problem of interest. The remainder of this chapter introduces the rigorous, though still incomplete, tools introduced then that revolutionized the classification of problem complexity and the corresponding algorithm choices.

14.2 MEASURING ALGORITHMS AND INSTANCES

Any discussion of computation time in optimization begins with the observation that larger instances of any problem are likely to require more effort to solve than small ones. Efficiency must be defined relative to instance size.

Computational Orders

The **time** required by an algorithm can be defined informally as the number of elementary steps like additions, subtractions, multiplications, divisions, and comparisons needed to complete computation. We assume for the moment that all these operations can be done in unit time.

Recognizing that the effort required for any computation will grow with instance size, complexity theory describes algorithm efficiency as a function of size.

Definition 14.2 | The **computational order** of a given algorithm, denoted $O(\cdot)$, is a bound on the required time to complete computation on a problem instance as a function of its size.

Thus an $O(n^2)$ algorithm will require time growing at most with the square of instance size n , while an $O(2^n)$ algorithm consumes exponentially growing time.

The bound should be **worst case**, covering every instance, however perverse, to be sure no exceptions need to be treated. Otherwise, it would not really be a bound. Furthermore, much of the deep complexity theory of Sections 14.5–14.6 will be seen to depend on covering every case in establishing computation orders for given problems.

The following table illustrates why we need to focus only on computation time trends as n grows large. The ranking of time requirements at $n = 10$ changes dramatically as the trend emerges for even a modest $n = 100$.

Order	Time at $n = 10$	Time at $n = 100$
$\log n$	1 hour	2 hours
n^2	10 minutes	16 2/3 hours
n^5	1 minute	69+ days
2^n	1 second	10^{17} centuries

Recognizing that only the trends at large n really matter permits us to skip over a host of details that only cloud the real messages to be revealed.

- It is not important how computation times rank at small instance sizes (see $\log n$ vs. 2^n in the table). We are interested in the comparison only after instance size growth has revealed the trend.
- Only the **dominant** or highest-order term in the expression for the required number of steps needs to be considered (see n^2 vs. n^5 in the table). The effects of lower order terms will be overwhelmed as instances grow large.
- Constant multiples can be ignored in computing orders because effects of say doubling or tripling times will also be overwhelmed at larger sizes.
- Ignoring multiples also frees us from concerns about exactly what computer algorithms we are trying to bound or the units in which times are expressed. Speeding the time of computations, or changing the units, will only update the bound by the ratio of new vs. old times to execute arithmetic operations or the units in which they are expressed.

EXAMPLE 14.1: DETERMINING COMPUTATIONAL ORDERS

Consider the task of sorting instances defined by n numbers into nondecreasing sequence by their values. One algorithm (not the best) starts a sorted output list with one of the numbers and then inserts each of the others in turn by starting at the bottom of the list and stepping up one by one until the right place for the new number is identified. Determine the computational order of this algorithm.

Solution: Summing the number of steps that could be required before each insertion point is discovered, gives

$$1 + 2 + \dots + (n - 1) = \frac{1}{2} (n - 1)(n) = \frac{1}{2} (n^2 - n)$$

Notice that the insertion point might be discovered more quickly in some cases, but bounds need to reflect the worst case. Following the above conventions, attention can be focused on the dominant term n^2 of the order. Also, the constant $\frac{1}{2}$ can be ignored. The result is to score the given algorithm an $O(n^2)$ one.

Instance Size as the Length of an Encoding

To apply the notion of computational orders to the full range of problems and algorithms requires more precision about what is meant by **instance size**. In many optimization cases, it is satisfactory to rely on the normal intuitions about the number of elements in an instance. How many constraints and/or variables are in an instance? How many nodes and/or arcs are in an instance on graphs? Still, the issue becomes more subtle when the magnitudes of instance constants/parameters like costs, constraint coefficients, capacities, and right-hand sides materially affect the arithmetic effort to accomplish algorithm steps (e.g., in inverting matrices), or even change the number of arithmetic steps to be done (e.g., BKP, Examples 14.1 and 14.2 below).

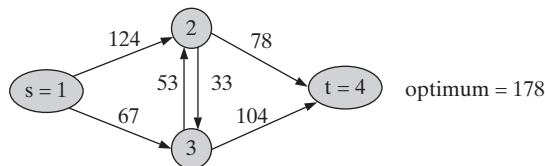
For a more robust definition of instance size, modern complexity theory falls back on concepts first developed by computing pioneer Dr. Alan Turing in the 1930s before computers even existed. It recognizes that to submit an instance for computer solution it must be reduced to a string of symbols, called an **encoding**, which provides a full description.

Definition 14.3 | The formal **size of an instance** of any given problem is the length of its encoding as a string of symbols drawn from a finite alphabet that fully defines the main structure of the instance and details its constant parameters.

Obviously, the encoding length can vary depending on the alphabet and other conventions used, but the difference will often be one of those constant multipliers which can be disregarded in assessing algorithm efficiency.

EXAMPLE 14.2: ENCODING MAXIMUM FLOW INSTANCES

To illustrate encoding, return to the **Maximum Flow Problem (MFlow)** of Section 10.8. An instance is a directed graph $G(V, A)$, with constants $u_{i,j}$ on the arcs in $(i, j) \in A$ specifying flow capacities, and two specific nodes in V chosen as the flow origin/source s and destination/sink t . The problem seeks a feasible network flow on the digraph that maximizes the total flow from s to t , treating all other nodes in V as transshipment. The following figure shows a simple instance on 4 nodes, with source $s = 1$ and sink $t = 4$.



- (a) Specify a finite alphabet of symbols sufficient to encode such instances.
- (b) Detail and justify an encoding of the above instance in symbols of the defined alphabet.

Solution:

- (a) We can use the digits $0, \dots, 9$ plus delimiters & within #.
- (b) One encoding of such instances in this alphabet would start by naming the source s and sink t , then enumerate the other members of vertex set V , and complete with a list of all arcs in $(i, j) \in A$ and their capacities $u_{i,j}$. For the above instance, the result is

1&4#2&3#1&2&124#1&3&67#2&3&33#2&4&78#3&2&33#3&4&104

Expressions for Encoding Length of All a Problem's Instances

The encoding length for any given problem instance can easily be determined by simple counting. Still, like most other things in complexity theory, what is important is how the length of the encoding grows with the various dimensions of instance input, including the size of constant parameters. For convenience, most complexity theory assumes such constants are integer, and if they are at least rationals the instance can be rescaled to make them integers.

Principle 14.4 Integer constants in problem instances are usually assumed to be encoded in a fixed-base number system so that their size is characterized by the logarithm of their magnitude.

Such encodings are often called **binary encodings**, thinking of numbers as base-2. Still, the base is irrelevant. The number of digits to express any integer q in the familiar base-10 system is $\lceil 1 + \log_{10} |q| \rceil$, which can be treated as simply $\log_{10} |q|$ in broad complexity analysis. Changing to binary only multiplies by a constant.

$$\log_2 |q| = \log_2 10 \cdot \log_{10} |q|$$

It is sufficient to simply refer to the size of parameter q as $\log |q|$.

EXAMPLE 14.3: LENGTH OF MAXIMUM FLOW ENCODINGS

Return to Maximum Flow Problems (**MFlow**) of Section 10.8 and the of Example 14.2 over directed graphs $G(V, A)$, with capacities $u_{i,j} > 0$ on the arcs in $(i, j) \in A$, and two specific nodes in V chosen as the origin/source s and destination/sink t . Develop and justify an expression bounding the length of the inputs for such (**MFlow**) instances.

Solution: Following the pattern of Example 14.2, and disregarding delimiters between items in the encoding, the total length of the encoding of the capacities is

$$\sum_{(i,j) \in A} \log u_{i,j}$$

This is bounded by $|A| \cdot \log u_{\max}$ where u_{\max} is the largest capacity. Other elements of size such as the lists the nodes and arcs grow with $|V|$ and $|A|$ alone. Thus a good description of the full length of Max Flow instance inputs is

$$O(|V| + |A| + |A| \cdot \log u_{\max})$$

14.3 THE POLYNOMIAL-TIME STANDARD FOR WELL-SOLVED PROBLEMS

Ideas of computational orders for algorithms and measures of instance size are valuable guides to the efficiency of solution procedures for most optimization problem forms. Knowing that a problem admits one algorithm that is say $O(n^2)$, for instance size n , and another that is $O(n^3)$, informs anyone trying to solve instances of that problem to at least give priority consideration to the first method.

Still, the challenge outlined in Section 14.1 is far grander. When should a problem form be classified well-solved? The remarkably simple answer that has emerged from modern complexity theory is elegantly concise.

Principle 14.5 An optimization problem is considered well-solved in the complexity theory sense if it admits a **polynomial-time** algorithm for every instance, that is, an algorithm running in computation time bounded by a polynomial (constant power) function of the instance size (length of its input).

EXAMPLE 14.4: RECOGNIZING POLYNOMIALLY BOUNDED COMPUTATION

For n the number of entities in an instances of a given problem, and q a constant parameter for instances, determine whether an algorithm with time bounded by each of the following orders is polynomial-time:

$$O(n^2), O(n^{25}), O(n^2 \log n), O(n^2 \sqrt{n}), O(2^n), O(n \cdot q)$$

Solution: $O(n^2)$ is clearly polynomial with time bounded by constant power 2, as is $O(n^{25})$, although the latter has computation that grows very rapidly with instance size. $O(n^2 \log n)$ and $O(n^2 \sqrt{n})$ also qualify as polynomial-time because computation growth in both is bounded by constant-power $O(n^3)$. Exponential growth, with size in the exponent like $O(2^n)$, is definitely not polynomial-time because there is no constant power that bounds every case. More subtly, $O(n \cdot q)$ is also not polynomial-time because in the binary-encoding length of the parameter q , such an algorithm is $O(n \cdot 2^{\log q})$ which is exponential in the length of q .

The polynomial-time standard was another of the theoretical computing contributions of pioneer Alan Turing. It was extended to optimization in the 1970s by other pioneers Jack Edmonds, Stephen Cook, and Richard Karp. Table 14.1 shows how it aligns so well with findings about best algorithms for many of the problems treated in this book.

TABLE 14.1 Well-Known Problem Forms and Polynomial Solvability

Polynomially Solvable	Believed Not
Linear Programs (LP)	Integer Linear Programs (ILP)
Spanning Tree Problem (MST)	Steiner Tree Problem (Stein)
Network Flow Problem (NetFlo)	Fixed Charge Network Flow Problem (FCNP)
Linear Assignment (Asmt)	Quadratic Assignment (QAsmt)
Maximum Flow (MFlow)	Generalized Assignment (GAsmt)
2-Matching (2Match)	3-Matching (3Match)
Shortest Path (SPath)	Longest Path (LPath)
CPM Scheduling (CPM)	Minimum Makespan Scheduling (MSpan)
	Travelling Salesman Problem (TSP)
	Set Packing (SPack)
	Set Covering (SCover)
	Set Partitioning (SPartn)
	Vertex Covering (VCover)
	Knapsack Problem (KP)
	Binary Knapsack Problem (BKP)
	Multi-dimension Knapsack Problem (MKP)
	Capital Budgeting Problem (CapBud)
	Facilities Location Problem (FLP)
	Vehicle Routing Problem (VRP)

14.4 POLYNOMIAL AND NONDETERMINISTIC-POLYNOMIAL SOLVABILITY

Having recognized the polynomial-time solvability standard for well-solved problems (principle [14.5](#)), the big question is, “How broad is the family of polynomially solvable problems?” Table 14.1 showed that many of the most familiar optimization models meet the standard, but many more remain formally uncertain; no polynomial-time algorithm is known, but there is no definitive proof that none is possible. This section introduces formal families of problems that provide a foundation for investigating those uncertainties.

Decision versus Optimization Problems

Most of this formal complexity theory is developed in terms of **decision problems**, which are problems that can be answered simply “yes” or “no.” For example the problem of determining whether the given instance of a directed graph is acyclic (free of dicycles) is a decision problem. Algorithms like those of Section 9.6 can determine whether or not the instance is acyclic with an easy polynomial-time search.

Most optimization problems demand more. An optimal choice of the decision variables is required, or at least the optimal value of the objective function, rather than a simple “yes” or “no.”

Principle 14.6 No typical optimization model is a decision problem.

Still, some restricted forms of optimizations do fit the definition.

Principle 14.7 | Given an instance of an optimization problem (**Opt**), the more limited **feasibility version** asking whether the instance admits any feasible solutions (denoted (**Opt^{feas}**)), is a decision problem.

Principle 14.8 | Given an instance of an optimization problem (**Opt**) and a **threshold v** , the more limited **threshold version** asking whether the instance admits a feasible solution with objective value at least as good as v (denoted (**Opt[≤]**) for a minimize or (**Opt[≥]**) for a maximize) is a decision problem.

APPLICATION 14.2: SET PARTITIONING AND DECISION PROBLEMS

The **Set Partitioning Problem (SPartn)** is introduced in Section 11.3. It seeks a minimum total cost sub-collection of subsets S_j creating an exact partition of a target set S . Using decision variables $x_j \triangleq 1$ if subset j is chosen and $= 0$ otherwise, and parameter matrix

$$\mathbf{A} \triangleq \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The following is an instance over $S = \{1, 2, 3\}$, with $S_1 = \{1, 2, 3\}$ at cost 12, $S_2 = \{1\}$ at cost 3, $S_3 = \{1, 3\}$ at cost 7, $S_4 = \{2\}$ at cost 10, and $S_5 = \{2, 3\}$ at cost 5:

$$\begin{aligned} \min \quad & 12x_1 + 3x_2 + 7x_3 + 10x_4 + 5x_5 \\ \text{s.t.} \quad & \sum_{j=1}^5 a_{ij}x_j = 1 \quad \text{for all } i \in S \\ & x_1, \dots, x_5 \text{ binary} \end{aligned}$$

An optimal solution is $\mathbf{x}^* = (0, 1, 0, 0, 1)$ with total cost 8.

Like most optimizations (principle [14.6](#)) problem (**SPartn**) is clearly not a decision problem. It is answered by an optimal solution, not a simple “yes” or “no.”

Contrast with the feasibility variant (**SPartn^{feas}**) to determine whether the given instance of (**SPartn**) has any feasible solution. Possible answers are limited to “yes” or “no,” and (**SPartn^{feas}**) is indeed a decision problem (principle [14.7](#)). The above instance is a “yes” case.

The threshold version (**SPartn[≤]**), which is also a decision problem (principle [14.8](#)), adds a new objective value target v to the encoding. For the minimize instance above, the answer is “yes” for $v \geq 8$ because there are feasible solutions with values as low as 8. For $v < 8$ the answer is “no.”

Class P - Polynomially Solvable Decision Problems

The first major class of problems in complexity theory to consider are the polynomially solvable decision problems.

Definition 14.9 Class $\mathbf{P} \triangleq \{\text{decision problems solvable in polynomial time}\}$

None of the polynomially solvable optimization problems identified in Table 14.1 belongs to \mathbf{P} because none is a decision problem. Still, all of their feasibility and threshold version are members of \mathbf{P} . For example, for instances of minimizing linear programming problem (\mathbf{LP}), corresponding instances of feasibility version ($\mathbf{LP}^{\text{feas}}$) can be answered in polynomial-time by applying the known polynomial-time algorithm for computing a full optimal solution, which will provide a feasible solution if there is one, or stop with a provable conclusion of infeasibility.

Similarly, minimizing instances of threshold version (\mathbf{LP}^{\leq}) can be resolved in polynomial time by applying the polynomial-time optimization algorithm. It is only necessary to compare the threshold ν to the computed optimal solution value in order to decide “yes” or “no.”

Class NP - Nondeterministic-Polynomially Solvable Decision Problems

A subtle extension of the idea of polynomial-solvability is needed to deal with the more difficult optimization problems not known to be polynomially solvable in the usual sense (e.g., right-hand column of Table 14.1). It focuses on whether given or guessed solutions can be **verified** in polynomial-time. Specifically, a decision problem is said to be **nondeterministic-polynomial time solvable** if “yes” instances can be verified in polynomial time with the aid of a polynomial-length hint. Then the set of such models forms the second major building block of complexity theory.

Definition 14.10 Class $\mathbf{NP} \triangleq \{\text{decision problems nondeterministically solvable in polynomial time}\}$

APPLICATION 14.3: NONDETERMINISTIC SOLVABILITY OF ILP THRESHOLD

Nondeterministic solvability (verifiability) is actually much more straightforward in the optimization context than its clumsy name suggests. To see the idea, consider the general maximizing **Integer Linear Program (ILP)** and its threshold version (\mathbf{ILP}^{\geq}). An instance is defined by an m by n integer matrix \mathbf{A} , an n -vector of objective coefficients, \mathbf{c} , an m -vector of right-hand-sides, \mathbf{b} , and a threshold ν . Each instance of problem (\mathbf{ILP}^{\geq}) asks whether there exists an n -vector $\bar{\mathbf{x}}$ such that

$$\begin{aligned} \mathbf{c} \cdot \bar{\mathbf{x}} &\geq \nu \\ \mathbf{A}\bar{\mathbf{x}} &\leq \mathbf{b} \\ \bar{\mathbf{x}} &\geq \mathbf{0} \text{ and integer} \end{aligned}$$

If the answer is “yes,” how can that be proved in polynomial time? One way would be to begin from scratch and fully solve the corresponding optimization instance, checking the threshold at the end. But no polynomial-time algorithm is known for optimizing general ILPs.

What if instead a sufficiently good (objective at least ν) feasible solution \bar{x} could be guessed or found in some other nondeterministic manner? Confirming a “yes” conclusion would then require only $O(n)$ additions to verify the objective threshold, $O(mn)$ effort to check satisfaction of all main constraints, and $O(n)$ inspections to assure all $\bar{x}_j \geq 0$ and integer. This combines to $O(n) + O(mn) = O(mn)$ effort. Verifiability of “yes” cases is doable in polynomial-time if a solution is given.

Now suppose the right answer for an instance of (**ILP**[≠]) is “no.” No way is known to efficiently prove that, even with the help of a polynomial-length guess. At least for worst cases, an enumerative algorithm requiring exponentially growing time would probably be required. Proving the negative is just a lot harder. This is why class **NP** asks only about “yes” cases in defining nondeterministic-polynomial solvability.

Membership in **NP** is easy to establish for “yes” cases of all the familiar optimization problems treated in this book using exactly the kind of argument applied in Application 14.3. The same does not hold for every decision problem. The great Alan Turing established that a class of **Undecidable** problems exists for which not even “yes” instances can be verified in polynomial time. Fortunately, such problems fall outside the scope of this book.

Polynomial versus Nondeterministic Polynomial Problem Classes

Figure 14.1 displays many connections among different complexity classes of problems considered so far. The most important regards **P** vs. **NP**.

Principle 14.11 Complexity class **P** is a subset of **NP**.

This must be true because both “yes” and “no” instances of (decision) problems in **P** can be solved in polynomial time. The same algorithm would verify “yes” cases in polynomial time as required for **NP**.

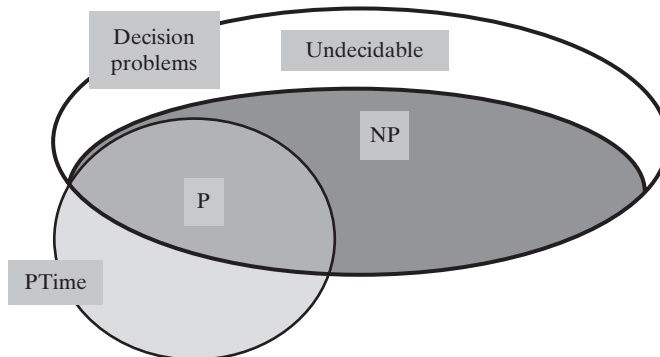


FIGURE 14.1 Polynomial vs. Nondeterministic Polynomial Problem Classes

It is also useful to include **PTime**, the collection of all polynomially solvable problems in the discussion. **PTime** includes **P** and much more.

Principle 14.12 Complexity class **P** is the decision-problem subset of **PTime**.

One last observation mentioned above is that the collection of decision problems contains more than **NP**, including the famous **Undecidable** class.

EXAMPLE 14.5: DISTINGUISHING THE COMPLEXITY CLASS OF PROBLEMS

Refer again to Table 14.1 and Applications 14.1–14.2 with the Minimum Spanning Tree problem (**MST**), the Minimum Steiner Tree problem (**Stein**), and the Set Partitioning problem (**SPartn**).

Decide where each of the following should be placed in the diagram of Figure 14.1: (**MST**), (**MST**[≤]), (**Stein**[≤]), (**SPartn**^{feas}).

Solution: We know optimization problem (**MST**) is polynomially solvable, so it would be placed in the part of **PTime** outside **P**. Its threshold version (**MST**[≤]) belongs to **P**. Although its optimization version is more difficult, the threshold version of Steiner Tree (**Stein**[≤]) still belongs to **NP**, but probably not **P**. The feasibility version of Set Partitioning (**SPartn**^{feas}) is similar. It belongs to **NP**, but probably not to **P**.

14.5 POLYNOMIAL-TIME REDUCTIONS AND NP-HARD PROBLEMS

The complexity classification of Figure 14.1 is still incomplete because it remains vague about problems not known to belong to **PTime** or **P**.

Polynomial Reductions between Problems

A relational operator that connects complexity of different problems can vastly enrich the investigation.

Definition 14.13 Complexity theory problem (**Q**₁) **polynomially reduces** to another (**Q**₂) (denoted (**Q**₁) ∝ (**Q**₂)) if a polynomial-time algorithm for (**Q**₂) provides one for (**Q**₁).

The notion of instances of one problem being solved by an algorithm for another is common in optimization. For example, Chapter 10 shows that Network Flow problems can be solved by Linear Programming, that is, (**NetFlo**) ∝ (**LP**). Similarly, (**LP**) ∝ (**ILP**) because an algorithm for Integer Linear Programs could be used to solve instances of LP. Notice that the more general or harder-to-solve problem falls on the open, right side of the ∝ operator, much like a larger number falls on the

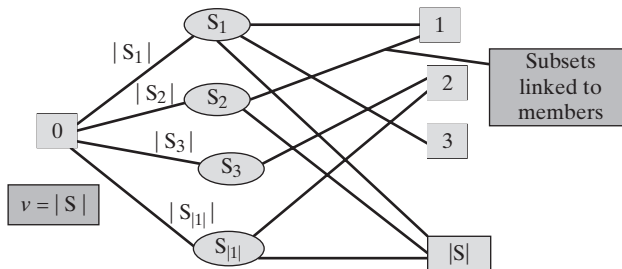
open side of a standard $<$. Problems (**NetFlo**) and (**LP**) on the left are the less complex of the two problems to solve in each reduction.

Earlier discussion of feasibility and threshold versions of optimization problems fits nicely in the reduction context.

Principle 14.14 For any optimization problem (**Opt**), feasibility version (**Opt^{feas}**) and threshold version (**Opt[≤]**) (or (**Opt[≥]**)) reduce to the full optimization version, that is, (**Opt^{feas}**) \propto (**Opt**), and (**Opt[≤]**) \propto (**Opt**) (or (**Opt[≥]**) \propto (**Opt**)).

APPLICATION 14.4: REDUCTION OF SET PARTITION TO STEINER TREE

For a less straight forward example of a polynomial reduction, return to the Steiner Tree problem of Application 14.1 and the Set Partitioning problem of Application 14.2. The figure below illustrates a proof that Set Partitioning feasibility version (**SPartn^{feas}**) polynomially reduces to a threshold version of Steiner Tree, that is, (**SPartn^{feas}**) \propto (**Stein[≤]**).



The argument begins with a typical instance of (**SPartn^{feas}**): a ground set of objects S , and a family of subsets $\{S_i \subseteq S : i \in I\}$. The question is whether any collection of those subsets provides an exact partition of S . To establish a reduction, a corresponding instance of Steiner Tree will be constructed that fulfills the task of checking feasibility of the given Set Partition instance by meeting a specified objective function threshold in the corresponding Steiner Tree.

The figure illustrates how this can be done. One source, Steiner-node is connected by edges to non-Steiner nodes for each of the subsets S_i with cost equal to subset size $|S_i|$. Then each such subset node i is joined to Steiner nodes for elements of the ground set that belong to S_i . Finally the threshold for the Steiner case is set at $|S|$ the number of objects in the ground set.

Clearly any Steiner Tree of the constructed instance will use subsets covering every element of $|S|$. If its cost, which will be the sum of the used subset sizes, satisfies the threshold, there can be no duplication among the selected subsets, and they provide the needed proof of Set Partition feasibility. Otherwise, there is no feasible solution.

Why is this a polynomial reduction? That is because the constructed Steiner instance has a size polynomially related to the length of the Set Partitioning one. If, for example, the construction required building an exponentially larger Steiner Tree

instance, a polynomial algorithm in the length of its massive size would not provide one for the Set Partition instance being investigated. The reduction must keep the sizes of both instances in sync.

Another insight is that nothing has been presented so far about the threshold version (\mathbf{Stein}^{\leq}) of Steiner Tree to suggest there actually is a polynomial-time algorithm to solve it; there probably is not. Still, the above reduction shows (\mathbf{Stein}^{\leq}) is at least as hard to solve as ($\mathbf{SPartn}^{\text{feas}}$). Whatever is known about the solvability of ($\mathbf{SPartn}^{\text{feas}}$) bounds what is possible for (\mathbf{Stein}^{\leq}).

Another important property of polynomial reductions is that they can be chained together to establish a reduction indirectly.

Principle 14.15 Polynomial reductions among problems are **transitive**. That is, if $(\mathbf{Q}_1) \propto (\mathbf{Q}_2)$ and $(\mathbf{Q}_2) \propto (\mathbf{Q}_3)$, then $(\mathbf{Q}_1) \propto (\mathbf{Q}_3)$.

For example, $(\mathbf{NetFlo}) \propto (\mathbf{ILP})$ because $(\mathbf{NetFlo}) \propto (\mathbf{LP})$, and in turn, $(\mathbf{LP}) \propto (\mathbf{ILP})$.

NP-Complete and NP-Hard Problems

Canadian researcher Stephen Cook revolutionized modern complexity theory by discovering that there are problems in **NP** to which every one of its members reduces. They form a new hardest subclass called **NP-Complete**.

Definition 14.16 Complexity class **NP-Complete** $\triangleq \{(\mathbf{Q}) \in \mathbf{NP} : \text{every member of } \mathbf{NP} \text{ reduces to } (\mathbf{Q})\}$.

The idea can be extended beyond decision problems by capturing those as hard as any member of **NP-Complete**.

Definition 14.17 Complexity class **NP-Hard** $\triangleq \{(\mathbf{Q}) : \text{some member of } \mathbf{NP-Complete} \text{ reduces to } (\mathbf{Q})\}$.

All that is required to place a problem in this hardest of complexity families relevant to optimization is to show a known member of **NP-Complete** reduces to it.

EXAMPLE 14.6: PROVING PROBLEMS NP-COMPLETE OR NP-HARD

Recall Application 14.4's proof that Set Partitioning feasibility ($\mathbf{SPartn}^{\text{feas}}$) \propto (\mathbf{Stein}^{\leq}), and assume it is known that ($\mathbf{SPartn}^{\text{feas}}$) is **NP-Complete**. Use these results to establish that (\mathbf{Stein}^{\leq}) is **NP-Complete** and full Steiner Tree optimization model (\mathbf{Stein}) is **NP-Hard**.

Solution: To show (\mathbf{Stein}^{\leq}) is **NP-Complete** requires first establishing that it belongs to **NP**, then finding a known **NP-Complete** problem that reduces to it.

As a “yes”/“no” problem for which given “yes” solutions can easily be verified in polynomial time, (**Stein**[≡]) does indeed belong to **NP**. Furthermore, Application 14.4 provides the needed reduction from an **NP-Complete** problem assuming (**SPartn**^{feas}) is **NP-Complete**.

Full optimization Steiner Tree problem (**Stein**) is not a decision problem (principle 14.6), so it cannot belong to any part of **NP**. Still principle 14.14 establishes that the threshold version of the problem reduces to the full optimization. Having just proved the threshold version **NP-Complete**, this assures full version (**Stein**) is **NP-Hard**.

Figure 14.2 adds **NP-Complete** and **NP-Hard** to the known complexity framework. Class **NP-Hard** includes all the problems provably as hard as any in **NP**, and **NP-Complete** is its decision problem subset.

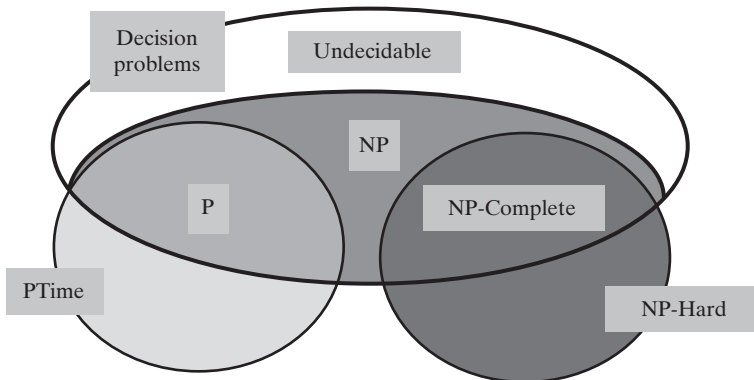


FIGURE 14.2 Believed Complexity Class Structure

14.6 P VERSUS NP

Thousands of well-studied optimization problems (including all those on the right side of Table 14.1) are by now known to be **NP-Hard**, with corresponding feasibility and threshold versions **NP-Complete**. The full import of this is revealed by considering what would follow if a polynomial-time algorithm were found for any member of either class.

Principle 14.18 If any single problem in **NP-Complete** or **NP-Hard** can be solved in polynomial time, then every member of **NP** is polynomially solvable, and **P = NP**. Consequently, unless **P = NP**, there can exist no polynomial-time algorithm for any **NP-Complete** or **NP-Hard** problem.

Every member of **NP-Hard** reduces from a member of **NP-Complete**, and every member of **NP** reduces to every member of **NP-Complete**. A polynomial algorithm for any problem in either class implicitly provides one for all of **NP**, making **P = NP**, because every other problem could be solved in polynomial time by reducing it to the one with a known algorithm.

The $P \neq NP$ Conjecture

The cartoons in Figure 14.3 depict the revolutionary impact of principle 14.18 for researchers seeking efficient (polynomial-time) algorithms for one or another **NP-Hard** problem of interest. As of this writing, all have failed, and principle 14.18 now reveals that a team seeking a polynomial algorithm for any **NP-Hard** problem is, in effect, simultaneously trying to achieve what generations of researchers have not been able to do – find a polynomial algorithm for any problem in **NP**. An algorithm for any member of **NP** would yield one for all.

Recognition of this almost breathtaking burden has led to a broadly accepted conjecture that $P \neq NP$.



(a) Team Failing on Their **NP-Hard** Problem



(b) Countless Others Sharing Their Failure on Equivalent Problems in **NP**

FIGURE 14.3 Argument for $P \neq NP$

Principle 14.19 Although it has yet to be proved, discovery that an optimization or related problem is **NP-Hard** makes it all but certain that no polynomial-time algorithm will be discovered that can deal with every instance.

It is worth emphasizing that many once broadly held beliefs in science and mathematics have ultimately been shown to be false. Until that happens with P vs. **NP**,

however, readers are well advised to first attempt to determine whether a problem of interest is **NP-Hard**, and if so, to look beyond polynomial-time exact algorithms to find useful tools that can address instances of interest.

14.7 DEALING WITH NP-HARD PROBLEMS

A finding that a problem is **NP-Hard** does not imply it is hopeless to investigate. After all, many of the most important applications of optimization fall on the right, **NP-Hard** side of Table 14.1. What is indicated is that more limited standards for effective solution need to be considered.

Special Cases

One of the first ways to seek solutions to **NP-Hard** optimization problems is to focus on tractable **special cases**.

Principle 14.20 | Special cases of optimization problem – subsets of instances – can often be solved in polynomial-time even though the full problem is **NP-Hard**.

Classification of a problem must consider its worst case – the most difficult instances to solve – if it is going to be up to the task of dealing with other hard problems that reduce to it. Difficult instances of one problem are unlikely to reduce to easy cases of another.

This does not preclude efficient solution of subsets of instances important in particular applications. For example, the Linear Assignment problem (**Asmt**) of Section 10.7 is a binary integer linear program with widespread application. Every instance of (**Asmt**) is an instance of (**ILP**). Still, efficient polynomial-time algorithms are known for even the worst case of (**Asmt**) although they probably do not exist for all cases of (**ILP**).

Pseudo-Polynomial Algorithms

Some important optimization models are **NP-Hard** because no algorithm is known to solve them in time polynomial in the formal length of instance input. Still, methods may be available with weaker standards of efficiency.

Definition 14.21 | An algorithm is said to be **pseudo-polynomial** if its computation is bounded by a polynomial in the number of main entities in the instance, and the magnitudes of its constant parameters.

The distinction comes in the treatment of constant parameter lengths. A truly polynomial-time bound would use the number of digits or logarithm of its magnitude for such constants (principle [14.4]). Pseudo-polynomial bounds consider the magnitude of constants, which grows much more quickly in the worst case. Although good

performance may result for instances with moderate-size constants, exponential time growth will eventually come to dominate.

An example of a pseudo-polynomial algorithm is the dynamic programming method for Binary Knapsack Problems (**BLP**) treated in Section 9.9. It runs in $O(nb)$ time, where n is the number of variables in the instance and b is the right-hand-side. Good performance could be expected for modest values of b , but (**BKP**) is **NP-Hard** when input length is measured in the standard way $O(n \log b)$.

Average Case Performance

Although worst-case bounds track well with algorithm time growth on instances of most optimization problems, actual performance is sometimes better represented by average times.

Principle 14.22 | Average-case bounds on computation times may be useful in predicting actual performance on instances of some problems, although they do not change the formal complexity classification of the problem.

A classic example of the distinction between worst- and average-case bounds arises with Linear Programs (**LP**). As shown in Table 14.1, (**LP**) is known to be polynomial-time solvable. Still, the polynomial bounds come from the interior-point methods of Chapter 7, not the Simplex algorithms of Chapter 5 and 6. Simplex performs well on a vast array of actual (**LP**) instances, but a small and unrepresentative number of cases are known to require an exponentially growing numbers of steps (see Section 7.6). This has not precluded Simplex from being widely used in practice or having its effectiveness documented by research on average-case bounds.

Stronger Relaxations and Cuts for B&B and B&C

Enumerative methods like Branch-and-Bound (B&B) and Branch-and-Cut (B&C) treated in Chapter 12 have exponentially growing worst-case computation times. Still enhanced relaxation and cutting planes of Sections 12.2 and 12.6 can greatly enhance their performance.

Principle 14.23 | Strong relaxations, cutting planes, and other methods of integer programming can often contain the ultimately exponential-time explosion of enumerative methods enough to permit solution of modest-size instances found in important applications.

One example is the strong valid inequalities for Set Covering problems (**SCover**) investigated in Section 12.6. They render instances of (**SCover**) among the better solved of ILPs.

Specialized Heuristics with Provable Worst-Case Performance

Another class of approaches to difficult **NP-Hard** problems are specialized heuristic algorithms with provable objective function performance.

Definition 14.24 **Guaranteed Performance** heuristics produce a feasible solution to the given instance of a hard optimization problem that has an objective function value within a provable multiple of optimal.

Application 14.5 illustrates for the famous **Travelling Salesman Problem (TSP)** of Section 11.5.

APPLICATION 14.5: TWICE-AROUND HEURISTIC FOR TSP WITH TRIANGLE INEQUALITY

The Traveling Salesman Problem seeks a minimum total length closed route or **tour** visiting each node of a given complete graph. Most forms are known to be **NP-Hard**.

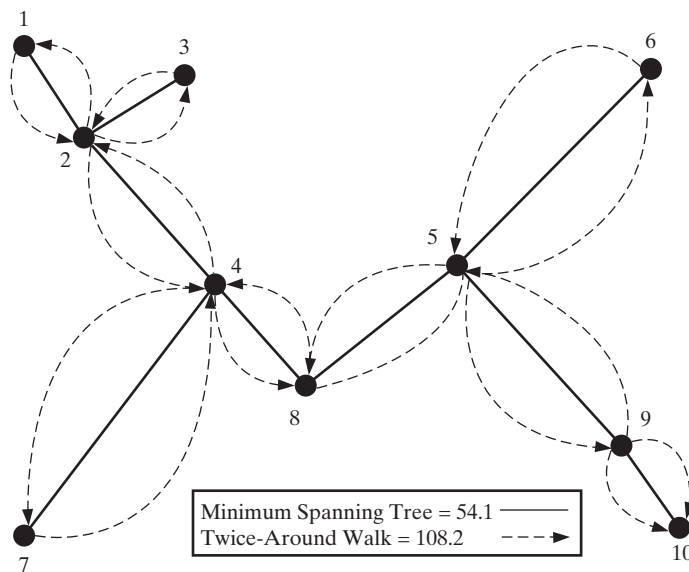
This holds even for instances with the special property that point-to-point distances $d_{i,j}$ satisfy the famous **triangle inequality**:

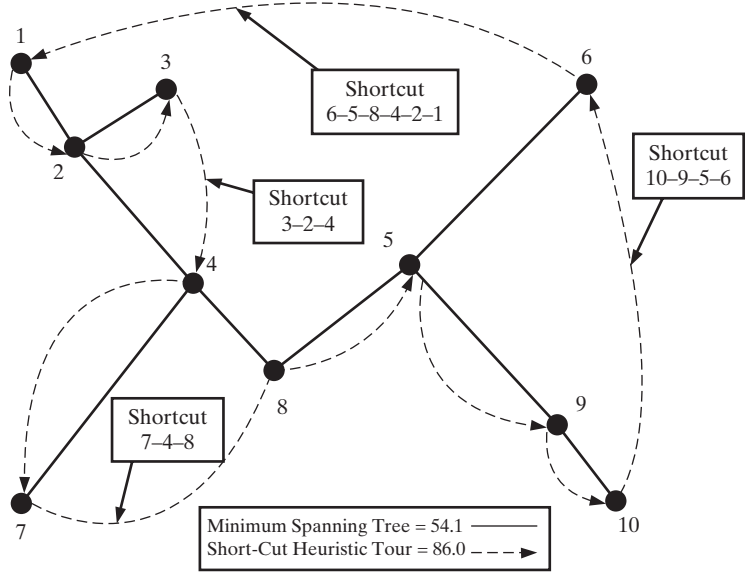
$$d_{i,k} \leq d_{i,j} + d_{j,k} \text{ for all nodes } i, j, k$$

That is, going directly from any node i to any other k is always at least as short as going via an intermediate node j .

NCB Application 11.8 of Section 11.5 is just such a case. Recall that it seeks a shortest route for a drilling machine to visit all 10 holes needed in a given circuit board. Distances in Table 11.7 are Euclidean and thus satisfy the triangle inequality. An optimal solution has total length 81.8 inches.

The following figures illustrate the famous **Twice-Around** approximate algorithm for such cases.





- The algorithm begins by computing the Minimum Spanning Tree of the given graph shown in heavy lines. Section 10.10 shows how that can be done in time polynomially bounded in the number of edges and nodes.
- Next a closed walk around the outside of the spanning tree is constructed by doubling each spanning tree edge (dashed arcs). The walk does visit every node as required for a TSP tour, but it duplicates some nodes.
- Finally, this walk is converted to a TSP tour by stepping from node to adjacent node, short-cutting past any already visited as in the second figure to avoid duplication. For example, progress goes from node 1, to 2, to 3, then bypasses a return to 3 by skipping to node 4 etc.

How far from optimal can the tour produced by this heuristic be? First, notice that the created tour's length is no more that two times the length of the minimum spanning tree. The original walk doubled each tree edge, and several were short-cutted in the tour, which, under the triangle inequality, can only reduce the total length. On the other hand, an optimal tour must be a connected subgraph visiting every node. This makes the length of the shortest spanning tree a lower bound on the TSP optimum. Taken together, we can conclude the twice-around heuristic solution is at most two times the length of an optimum. Although this 100% over optimum limit is not a very attractive guarantee, worst-case methods do give a satisfying certainty that the method will meet expectations for even the most perverse of instances.

General Purpose Approximate/Heuristic Algorithms

Finally, for the most difficult of **NP-Hard** optimization problems, the most attractive approach may be **heuristic or approximate** algorithms.

Principle 14.25 Heuristic or approximate algorithms, which produce at least feasible solutions with objective function values likely to be attractive, are often the only viable approach to instances of hard optimization problems big enough for practical application.

Chapter 15 offers many alternatives.

EXERCISES

14-1 Consider the following Binary Knapsack Problem (**BKP**) instance

$$\begin{aligned} \max \quad & 7x_1 + 9x_2 + 21x_3 + 15x_4 \\ \text{s.t.} \quad & 8x_1 + 4x_2 + 12x_3 + 7x_4 \leq 19 \quad (\text{BKP}) \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

- (a) Construct the corresponding Dynamic Programming digraph (like Figure 9.16), including showing objective function coefficients on all arcs needed.
- (b) What are the stages and what are the states of your model in (a)? Explain.
- (c) Starting with state 19, stage 1 initialized with objective value $v[19, 1] = 0$, and taking each later stages in turn, compute optimal subproblem values $v[\text{state}, \text{stage}]$ for each reachable state and state.
- (d) Use your results of (c) to identify an optimal solution to the full instance and determine its objective function value.

14-2 Return to the (**BKP**) of Exercise 14-1.

- ✓ (a) Define a finite alphabet of symbols, and then show a binary encoding of that instance in terms of your alphabet.
- ✓ (b) Establish that your encoding of (a) has length proportional to the number of variables n , and the logarithms (rounded up) of objective coefficients c_j , main constraint coefficients a_j , and RHS b .
- ✓ (c) Explain why the formulation of Exercise 14-1 is an instance of the Binary Knapsack Problem (**BKP**) form

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n a_j x_j \leq b \\ & x_1, \dots, x_n = 0 \text{ or } 1 \end{aligned}$$

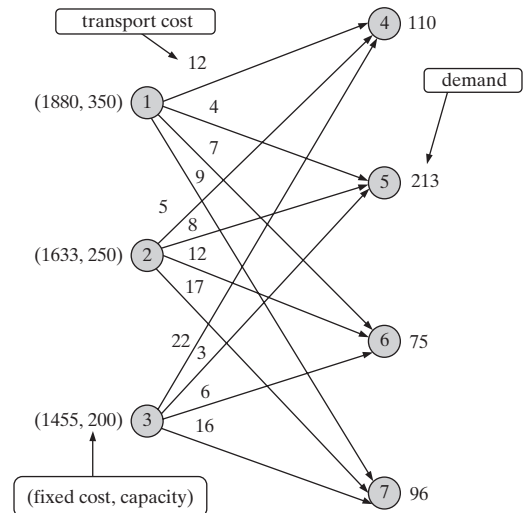
- ✓ (d) The Dynamic Programming algorithm of 14-1 solves instances of (**BKP**) by

computing a longest path in a network across n stages with at most b states each. Explain why those computations are polynomial in the magnitudes of the constants, but exponential in the standard binary encoding.

- ✓ (e) Explain why (d) makes (**BKP**) pseudo-polynomially solvable.
- ✓ (f) Comment on what (d) and (e) tell us about how easy or hard (**BKP**) instances are to solve, at least those of modest size.
- ✓ (g) (**BKP**) is known to be **NP-Hard**, so all problems in **NP** reduce to it. Would you expect manageable instances like those of (f) to be the kinds that result from reduction of a very hard problem form in **NP**? Explain.

14-3 Return to the Facilities Location Problem (**FLP**) of Chapter 11, definition 11.29, and assume all parameters are integer.

- (a) ✓ Consider the following instance:



Define a finite alphabet of symbols, and then show a binary encoding of the instance in terms of your alphabet.

- ✔ (b) Establish that your encoding of (a) has length proportional to the number of facilities m , demand points n , and the logarithms (rounded up) of objective and constraint coefficients.
- ✔ (c) State the threshold version (\mathbf{FLP}^{\leq}) of the full problem for given threshold v .
- ✔ (d) Explain why the threshold model of (c) belongs to class \mathbf{NP} but the full optimization model (\mathbf{FLP}) does not.
- ✔ (e) Threshold version (\mathbf{FLP}^{\leq}) is known to be $\mathbf{NP-Complete}$. Explain why this implies the full optimization form (\mathbf{FLP}) is $\mathbf{NP-Hard}$.
- ✔ (f) What would be the consequences of discovering a polynomial-time algorithm for either the full optimization model or its threshold analog? Explain.

14-4 Now return to the Fixed Charge Network Flow Problem (\mathbf{FCNP}) of Chapter 11, definition [11.31](#), and assume all data are integer.

- (a) Develop and justify an expression for the length of a binary encoding for an instance in terms of the dimensions and parameters of the model.
- (b) State the threshold version (\mathbf{FCNP}^{\leq}) for given threshold v .
- (c) Explain why the threshold problem of (b) belongs to \mathbf{NP} .
- ✔ (d) Detail a polynomial reduction from the (\mathbf{FLP}^{\leq}) of Exercise 14-3 (c) to your threshold problem (\mathbf{FCNP}^{\leq}) of (b).
- ✔ (e) How do (b), (c), and $\mathbf{NP-Completeness}$ of (\mathbf{FLP}^{\leq}) (Exercise 14-3 (e)) lead to the conclusion that the threshold version (\mathbf{FCNP}^{\leq}) also belongs to $\mathbf{NP-Complete}$.
- ✔ (f) Explain why (e) implies the full optimization model (\mathbf{FCNP}) is $\mathbf{NP-Hard}$.
- ✔ (g) What would be the consequences of finding a polynomial-time algorithm for either the full optimization problem or its threshold analog? Explain

14-5 The Capital Budgeting Problem (\mathbf{CapBud}) of Section 11.2 over a set of n proposed projects j using decision variables $x_j = 1$ if project j is

chosen and $= 0$ otherwise can be formulated as follows:

$$\begin{aligned} \max \quad & \sum_{j=1}^n r_j x_j && \text{(maximize total return)} \\ \text{s.t.} \quad & \sum_{j=1}^n a_{t,j} x_j \leq b_t \text{ for all } t && \text{(budget limits in times } t) \\ & x_j \leq x_k \text{ for all } j, k \in P && \text{(project pairs subject to precedence)} \\ & x_j + x_k \leq 1 \text{ for all } j, k \in M && \text{(mutually exclusive project pairs)} \\ & x_j \text{ binary } j = 1, \dots, n \end{aligned}$$

- (a) Develop and justify an expression for the length of a binary encoding for an instance in terms of the dimensions and parameters of the model.
- (b) State the threshold version (\mathbf{CapBud}^{\geq}) of the problem for given threshold v , and establish that it belongs to complexity class \mathbf{NP} .
- (c) The threshold version (\mathbf{BKP}^{\geq}) of the Binary Knapsack Problem discussed in Exercise 14-2 above is known to be $\mathbf{NP-Complete}$. Use that fact along with part (b) to establish that (\mathbf{CapBud}^{\geq}) is also $\mathbf{NP-Complete}$. Be sure to fully detail the required reduction among instances of the two problems.
- (d) Explain what the result of (c) implies for the prospects of finding a polynomial time (binary encoding) algorithm for either (\mathbf{CapBud}^{\geq}) or the full optimization version (\mathbf{CapBud}), and why.

14-6 The Multi-dimensional Knapsack Problem (\mathbf{MKP}) over n decision variables $x_j = 1$ is object j is chosen and $= 0$ otherwise, can be formulated:

$$\begin{aligned} \max \quad & \sum_{j=1}^n r_j x_j && \text{(maximize total return)} \\ \text{s.t.} \quad & \sum_{j=1}^n a_{ij} x_j \leq b_i \text{ } i = 1, \dots, m \text{ (capacities } i) \\ & x_j \text{ binary } j = 1, \dots, n \end{aligned}$$

- (a) Develop and justify an expression for the length of a binary encoding for an instance in terms of the dimensions and parameters of the model.

- (b) Explain why the **(BKP)** instance in Exercise 14-1 is an instance of **(MKP)**.
- (c) State the threshold version **(MKP[≥])** of **(MKP)** for given threshold v , and establish that it belongs to complexity class **NP**.
- (d) The threshold version **(BKP[≥])** of the Binary Knapsack Problem treated in Exercise 14-2 is known to be **NP-Complete**. Use that fact along with part (c) to establish that **(MKP[≥])** is also **NP-Complete**. Be sure to fully detail the required reduction among instances of the two problems.
- (e) Explain what the result of (d) implies for the prospects of finding a polynomial time (binary encoding) algorithm for either **(MKP[≥])** or the full optimization version **(MKP)**, and why.

14-7 Given a digraph $G(V, E)$, the Vertex Cover problem **(VCover)** seeks a minimum cardinality subset of vertices that together touch every edge of E .

- (a) Show that the problem can be modeled in terms of decision variables $x_i = 1$ if vertex i is in the solution and $= 0$ otherwise as ILP

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i \\ \text{s.t.} \quad & \sum_{i \in I_e} x_i \geq 1 \quad \text{for all } e \in E \\ & x_i \text{ binary} \quad \text{for all } i \in V \end{aligned}$$

where $I_e \triangleq \{i \in V\}$ end points of edge e .

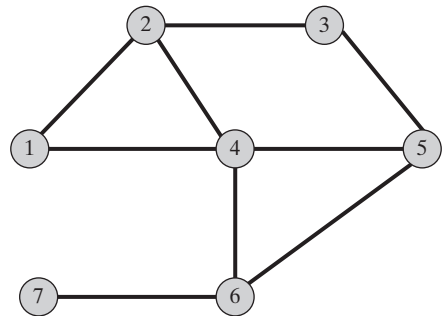
- (b) Now recall the weighted Set Covering problem **(SCover)** of Section 11.3 seeking a minimum total weight subcollection of subsets $S_j, j \in J$, which together include each element of union $S \triangleq \cup_{j \in J} S_j$ at least once. In terms of decision variables $x_j = 1$ if subset j is included and $= 0$ otherwise, plus objective function coefficients c_j , the problem can be modeled as ILP:

$$\begin{aligned} \min \quad & \sum_{j \in J} c_j x_j \\ \text{s.t.} \quad & \sum_{\{j \text{ with } i \in S_j\}} x_j \geq 1 \quad \text{for all } i \in S \\ & x_j \text{ binary} \quad \text{for all } j \in J \end{aligned}$$

- (c) Show that every instance of **(VCover)** can be viewed as an instance of **(SCover)**. Be sure to fully detail how elements of the two formulations correspond.
- (d) Problem **(VCover)** is known to be **NP-Hard**. Use that fact and your result of part (c) to show that **(SCover)** is **NP-Hard**.

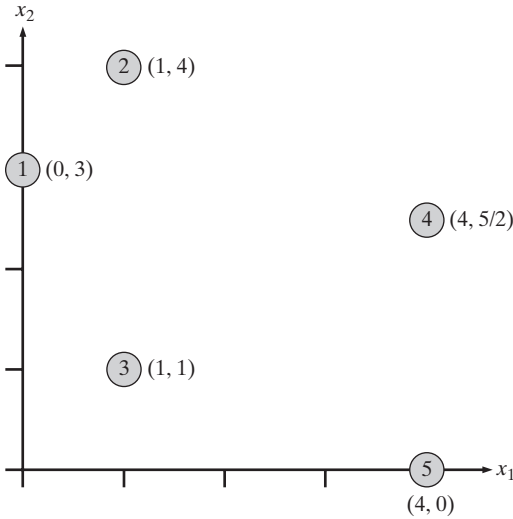
14-8 Return to the cardinality Vertex Cover problem **(VCover)** of Exercise 14-7(a). One algorithm to construct an approximately optimum set $\bar{V} \triangleq \{i \in V: x_i = 1\}$ for the problem can be stated as follows: (i) start with $\bar{V} \leftarrow \emptyset$; then (ii) as long as there exists any edge $e \in E$ with $I_e \cap \bar{V} = \emptyset$, update $\bar{V} \leftarrow \bar{V} \cup I_e$. That is, add both end nodes of any edge not covered by the current solution \bar{V} until no more such edges exist.

- ✓ (a) Apply the algorithm to the following instance beginning with $\bar{V} = \{1, 2\}$.



- ✓ (b) Explain why the algorithm is guaranteed to produce a feasible cover of all edges in any given instance.
- ✓ (c) Justify that this approximate algorithm runs in time polynomial in the numbers of vertices and edges.
- (d) Explain why an optimal vertex cover must contain at least one end of every edge e encountered at step (ii) of the algorithm. Then show how this assures the carnality of the approximate solution obtained above is no more than twice optimal.
- (e) Explain how existence of a guaranteed-performance heuristic like the one above is not inconsistent with the fact that **(VCover)** is **NP-Hard**.

14-9 Return to the Traveling Salesman Problem (**TSP**) and the **Twice-Around** heuristic of Application 14.5. Then consider an instance on the 5 points in the following plot:



Arcs can be assumed to exist between all pairs of points, and distances are Euclidean.

- (a) Compute a minimum spanning tree of the points in the plot using Algorithm 10F and determine its total length.

- (b) Start from point 1 and sketch the corresponding twice-around walk of the points. Also show its total length.
- (c) Starting again from point 1, shorten the walk of part (b) into a TSP tour, and compute the tour's total length.
- (d) Use results of parts (a) and (c) to compute an upper and a lower bound on the length of an optimal TSP tour for the given instance. Then verify that the approximate solution of (c) is no more than twice the length of an optimal tour.
- (e) Explain how existence of a guaranteed-performance heuristic like Twice-Around is not inconsistent with the fact that (**TSP**) is **NP-Hard**, even when arc lengths are Euclidean.

14-10 Return to the lists of problems in Table 14.1. For each of the following pairs of problems in the table, establish that the first, polynomially solvable one is a special case of the **NP-Hard** second, and identify the special properties of the first that prevent it from yielding worst-case instances of the second.

- ✓ (a) (**MST**) vs. (**Stein**).
- (b) (**Asmt**) vs. (**GAmt**).
- (c) (**CPM**) vs. (**LPath**).

REFERENCES

Garey, Michael R. and David S. Johnson (1979), *Computers and Intractability - A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, California.

Hochbaum, Dorit S., editor (1997), *Approximation Algorithms for NP-Hard Problems*, PWS Publishing, Boston, Massachusetts.

Martin, R. Kipp (1999), *Large Scale Linear and Integer Optimization*, Kluwer Academic, Boston, Massachusetts.

Parker, R. Gary and Ronald L. Rardin (1988), *Discrete Optimization*, Academic Press, San Diego, California.

Schrijver, Alexander (1998), *Theory of Linear and Integer Programming*, John Wiley, Chichester, England.

Wolsey, Laurence (1998), *Integer Programming*, John Wiley, New York, New York.

Heuristic Methods for Approximate Discrete Optimization

Almost all of the methods considered in previous chapters for dealing with ILPs and INLPs are at least nominally addressed to **exact optimization**—assuring a provably optimal solution if allowed to run long enough. Unfortunately, the main message of the complexity theory developed in Chapter 14 (principle [14.19]) is that a huge number of important ILP and INLP problem forms will probably never admit the sort of polynomial-time algorithm, assuring that arbitrarily large instances can be solved to exact optimality.

The result is that the overwhelming majority of large-scale discrete optimization applications settle for some form of **heuristic/approximate optimization** method seeking good and feasible, but not necessarily optimal solutions within manageable computational effort. Many heuristic methods merely adapt one of the exact methods—stopping searches early and accepting the best feasible solution discovered. As Branch and Bound, Branch and Cut, and Large-Scale technologies evolve, larger and larger instances can be successfully addressed in this way. Still, underlying exponential growth will eventually overwhelm.

This chapter introduces some important, but strictly heuristic alternatives that follow strategies bearing little more than surface similarity to exact methods. Instead they pursue opportunistic and intuitive drives for hopefully good feasible solutions, exploiting problem structures where available. The results may be very good indeed, but their quality can usually be assessed only empirically by experiments over a variety of instances. Such strictly heuristic methods rarely stop with any mathematical guarantee of how nearly optimal their results may be.

15.1 CONSTRUCTIVE HEURISTICS

The first category of heuristics to consider, **constructive searches**, build an approximate optimum incrementally. Like the partial solutions of Chapter 12, values for the discrete decision variables are chosen one-by-one, terminating when a full feasible solution is completed. They proceed through partial solutions, choosing values

for decision variables one at a time and (often) stopping upon completion of a first feasible solution.

Rudimentary Constructive Search Algorithm

Constructive searches typically begin with every discrete component of the decision vector free. At each iteration, one previously free variable is fixed at a value feasible with decisions fixed so far. That is, the chosen value for the new component should not produce constraint violation when previously fixed values are substituted and optimistic assumptions about free variable values are adopted.

In the simplest case, the process terminates when no free variables remain. Algorithm 15A gives a more formal statement.

ALGORITHM 15A: RUDIMENTARY CONSTRUCTIVE SEARCH

Step 0: Initialization. Start with all-free initial partial solution $\mathbf{x}^{(0)} = (\#, \dots, \#)$ and set solution index $t \leftarrow 0$.

Step 1: Stopping. If all components of current solution $\mathbf{x}^{(t)}$ are fixed, stop and output $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(t)}$ as an approximate optimum.

Step 2: Step. Choose a free component x_p of partial solution $\mathbf{x}^{(t)}$ and a value for it that plausibly leads to good feasible completions. Then, advance to partial solution $\mathbf{x}^{(t+1)}$ identical to $\mathbf{x}^{(t)}$ except that x_p is fixed at the chosen value.

Step 3: Increment. Increment $t \leftarrow t + 1$, and return to Step 1.

Greedy Choices of Variables to Fix

Obviously, the bulk of the effort in constructive searches goes to choosing the next free variable to fix and picking its value. Most common procedures accomplish these tasks in a **greedy** or **myopic** fashion.

Definition 15.1 Greedy constructive heuristics elect the next variable to fix and its value that does least damage to feasibility and most helps the objective function, based on what has already been fixed in the current partial solution.

That is, greedy rules choose the fix that seems most likely, on the basis of what is presently known, to lead to a good feasible completions.

In very rare cases (e.g., the Spanning Tree problems of Section 10.10) such greedy approaches are guaranteed to produce an exact optimum. Much more commonly, they risk suffering from looking only at local information about the next choice. Quite possibly, a decision that appears very good with only a few variables fixed will actually end up forcing the search into a very poor part of the feasible space. Still, if the procedure is to be computationally efficient, compromises have to be made.

Greedy Rule for NASA Application

We can illustrate the idea of constructive search with the NASA capital budgeting model (11.7). There, the decision variables

$$x_j \triangleq \begin{cases} 1 & \text{if mission } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

It is natural to construct a solution for capital budgeting models like this one by successively adding missions until constraints block further inclusions. That is, we will seek to fix a previously free x_j at value 1 as long as any free mission can be selected without violating a constraint.

To implement this natural constructive search we need a greedy selection criterion of the type described in [15.1]. Certainly, we should prefer missions with high objective function “value” coefficients. But we also want to consider constraints. One high-value mission might consume so much of the various budgets that it would block all further decisions. Also, precedence constraints change the implicit value of a mission; choosing the mission gains its value and makes successors feasible.

Our search will trade-off these objective and constraint considerations in a common way, by comparing missions according to the ratios

$$r_j \triangleq \frac{\left(\begin{array}{c} \text{project} \\ j \text{ value} \end{array} \right) + \left(\begin{array}{c} \text{allowance for} \\ \text{enabled successor} \\ \text{values} \end{array} \right)}{\sum_{i=1}^8 \left(\begin{array}{c} \text{fraction of remaining} \\ \text{constraint } i \text{ right-hand} \\ \text{side consumed by } j \end{array} \right)} = \frac{c_j + \sum_{\text{free } k \text{ preceded by } j} \left(\frac{c_k}{2} \right)}{\sum_{i=1}^8 \left(\frac{a_{i,j}}{b_i^{(t)}} \right)} \quad (15.1)$$

where

$c_j \triangleq$ objective function coefficient for mission j

$a_{i,j} \triangleq$ coefficient for mission j in the i th main constraint

$b_i^{(t)} \triangleq$ right-hand side remaining in the i th main constraint after fixing variable values as in partial solution $\mathbf{x}^{(t)}$

Among the free projects for which all predecessors have been scheduled in partial solution $\mathbf{x}^{(t)}$, we will fix the x_j with maximum ratio r_j . If there remains room for j in all applicable budgets, it is fixed = 1. Otherwise, we set $x_j = 0$.

Like many such greedy indices, ratio (15.1) seems rather complicated at first glance. The numerator tries to account for both the immediate value of selecting a mission and the potential it opens up to select missions of which it is a predecessor. Half the value of all enabled successors is arbitrarily added to the mission’s direct value. The denominator of (15.1) sums the fractions of remaining constrained “resources” that a mission would consume if selected. Thus we favor missions using relatively little of now-scarce resources.

Ratioing value to resource use combines objective and constraint considerations. The highest r_j will correspond to a mission j high in value, or low in resource consumption, or both. Selecting that mission may not be the best long-term decision, but it does reflect about all we can know without looking more than one step into the future.

EXAMPLE 15.1: DEVISING GREEDY HEURISTIC RULES

Recall from Section 11.3 that set cover models seek a minimum cost collection of columns or subsets that together include or cover every element of a given set. One instance is

$$\begin{array}{llllll} \min & 15x_1 & + & 18x_2 & + & 6x_3 & + & 20x_4 \\ \text{s.t.} & + x_1 & & & & & + & x_4 & \geq & 1 \\ & + x_1 & + & x_2 & & & + & x_4 & \geq & 1 \\ & & & + & x_2 & + & x_3 & + & x_4 & \geq & 1 \\ & & & & & & & & & & x_1, \dots, x_4 = 0 \text{ or } 1 \end{array}$$

Explain why it would make sense to choose free x_j to fix $= 1$ by picking one with least ratio

$$r_j \triangleq \frac{\text{cost coefficient of column } j}{\text{number of uncovered elements that } j \text{ covers}}$$

Solution: The proposed ratio explicitly seeks minimum cost by including the objective function coefficient in its numerator. Still, it also considers feasibility in dividing by the number of still uncovered rows or elements each free j could resolve. The effect is to seek the most efficient next choice of x_j to fix $= 1$, the best in the short-term or myopic sense.

Constructive Heuristic Solution of NASA Application

Starting from the completely free partial solution

$$\mathbf{x}^{(0)} = (\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#)$$

all $b_i^{(0)}$ equal initial right-hand sides. Ratios for the first two j 's are

$$r_1 = \frac{200}{6/10} = 333.33$$

$$r_2 = \frac{3 + (18/2)}{(2/10) + (3/12)} = 26.67$$

and similar arithmetic yields

$$\begin{array}{llll} r_3 = 129.07, & r_4 = 29.17, & r_5 = 35.21, & r_6 = 21.54 \\ r_7 = 6.52, & r_8 = 7.37, & r_9 = 110.09, & r_{10} = 157.50 \\ r_{11} = 10.96, & r_{12} = 7.38, & r_{13} = 586.05, & r_{14} = 87.30 \end{array}$$

The highest of these ratios is 586.05 for mission 13. Since this mission fits within remaining right-hand sides $b_i^{(t)}$ and has no predecessors, we fix $x_{13} = 1$ to produce

$$\mathbf{x}^{(1)} = (\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, 1, \#)$$

Table 15.1 provides an abridged summary of the rest of the search. Processing of $t = 1$ parallels the first iteration, selecting and fixing $= 1$ additional mission 1.

TABLE 15.1 Constructive Search of NASA Application

<i>t</i>	Computation	Choice
0	$\mathbf{x}^{(0)} = (\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#)$ <hr/> $b_1^{(0)} = 10, b_2^{(0)} = 12, b_3^{(0)} = 14, b_4^{(0)} = 14.$ $b_5^{(0)} = 14, b_6^{(0)} = 1, b_7^{(0)} = 1, b_8^{(0)} = 1$ <hr/> $r_1 = 333.33, r_2 = 26.67, r_3 = 129.07, r_4 = 29.17,$ $r_5 = 35.21, r_6 = 21.54, r_7 = 6.52, r_8 = 7.37,$ $r_9 = 110.09, r_{10} = 157.50, r_{11} = 10.96, r_{12} = 7.38,$ $r_{13} = 586.05, r_{14} = 87.30$	Select $j = 13$ and fix $x_{13} = 1$
1	$\mathbf{x}^{(1)} = (\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, 1, \#)$ <hr/> $b_1^{(1)} = 10, b_2^{(1)} = 11, b_3^{(1)} = 10, b_4^{(1)} = 13.$ $b_5^{(1)} = 13, b_6^{(1)} = 1, b_7^{(1)} = 1, b_8^{(1)} = 1$ <hr/> $r_1 = 333.33, r_2 = 25.38, r_3 = 122.59, r_4 = 28.26,$ $r_5 = 31.05, r_6 = 19.55, r_7 = 6.05, r_8 = 7.22,$ $r_9 = 107.84, r_{10} = 133.06, r_{11} = 10.35, r_{12} = 7.04,$ $r_{13} = \text{N/A}, r_{14} = 79.56$	Select $j = 1$ and fix $x_1 = 1$
⋮	⋮	⋮
3	$\mathbf{x}^{(3)} = (1, \#, \#, \#, \#, \#, \#, \#, 1, \#, \#, 1, \#)$ <hr/> $b_1^{(3)} = 4, b_2^{(3)} = 3, b_3^{(3)} = 6, b_4^{(3)} = 13.$ $b_5^{(3)} = 13, b_6^{(3)} = 1, b_7^{(3)} = 1, b_8^{(3)} = 1$ <hr/> $r_1 = \text{N/A}, r_2 = 8.00, r_3 = 38.28, r_4 = 28.26,$ $r_5 = 17.50, r_6 = 18.35, r_7 = 1.71, r_8 = 7.22,$ $r_9 = 54.54, r_{10} = \text{N/A}, r_{11} = 9.61, r_{12} = 2.23,$ $r_{13} = \text{N/A}, r_{14} = 50.99$	Select $j = 9$ and fix $x_9 = 0$ because violates constraint 2
⋮	⋮	⋮
9	$\mathbf{x}^{(9)} = (1, \#, 0, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0)$ <hr/> $b_1^{(9)} = 4, b_2^{(9)} = 3, b_3^{(9)} = 6, b_4^{(9)} = 13.$ $b_5^{(9)} = 13, b_6^{(9)} = 1, b_7^{(9)} = 1, b_8^{(9)} = 1$ <hr/> $r_1 = \text{N/A}, r_2 = 8.00, r_3 = \text{N/A}, r_4 = \text{N/A},$ $r_5 = \text{N/A}, r_6 = \text{N/A}, r_7 = 1.71, r_8 = 7.22,$ $r_9 = \text{N/A}, r_{10} = \text{N/A}, r_{11} = 9.61, r_{12} = 2.23,$ $r_{13} = \text{N/A}, r_{14} = \text{N/A}$	Select second best $j = 2$ and fix $x_2 = 1$ because $j = 11$ has free predecessor
10	$\mathbf{x}^{(10)} = (1, 1, 0, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0)$ <hr/> $b_1^{(10)} = 2, b_2^{(10)} = 0, b_3^{(10)} = 6, b_4^{(10)} = 13.$ $b_5^{(10)} = 13, b_6^{(10)} = 1, b_7^{(10)} = 1, b_8^{(10)} = 1$ <hr/> $r_1 = \text{N/A}, r_2 = \text{N/A}, r_3 = \text{N/A}, r_4 = \text{N/A},$ $r_5 = \text{N/A}, r_6 = \text{N/A}, r_7 = 0.0000, r_8 = 0.0001,$ $r_9 = \text{N/A}, r_{10} = \text{N/A}, r_{11} = 0.002, r_{12} = 0.0000,$ $r_{13} = \text{N/A}, r_{14} = \text{N/A}$	Select $j = 11$ and fix $x_{11} = 1$
⋮	⋮	⋮
14	$\mathbf{x}^{(14)} = (1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)$	

Something different occurs at $t = 3$. The mission with the maximum r_j there is number 9. But mission 9 requires \$5 billion in the 2000–2004 budget period, and projects already chosen use all but $b_2^{(3)} = \$3$ billion. We have to fix $x_9 = 0$ to maintain feasibility.

Another peculiarity arises at iteration $t = 9$. There the mission with the best ratio is $j = 11$. However, mission 11 cannot be selected before predecessor mission 2. Thus we pass to the second best ratio, which happens to be $j = 2$, and fix $x_2 = 1$.

Our constructive search terminates when all 14 components of the decision vector have been fixed. The heuristic optimal solution produced is

$$\hat{\mathbf{x}} \triangleq \mathbf{x}^{(14)} = (1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)$$

flying missions 1, 2, 10, 11, and 13 at a total value of 671. (Compare with the branch and bound results in Table 12.3.)

EXAMPLE 15.2: EXECUTING CONSTRUCTIVE HEURISTICS

Return to the set covering problem of Example 15.1 and its suggested greedy ratio r_j . Use this ratio to apply Algorithm 15A, fixing selected $x_j = 1$ as long as rows remain uncovered and $= 0$ thereafter.

Solution: The search begins with all-free partial solution $\mathbf{x}^{(0)} = (\#, \#, \#, \#)$. All rows of the set covering model are uncovered, so ratios compute

$$r_1 = \frac{15}{2}, \quad r_2 = \frac{18}{2}, \quad r_3 = \frac{6}{1}, \quad r_4 = \frac{20}{3}$$

The least of these values occurs at $j = 3$, so we fix x_3 to obtain $\mathbf{x}^{(1)} = (\#, \#, 1, \#)$.

The third row of the model is now satisfied. This leads to revised ratios

$$r_1 = \frac{15}{2}, \quad r_2 = \frac{18}{1}, \quad r_4 = \frac{20}{2}$$

Choosing the least fixes x_1 in $\mathbf{x}^{(2)} = (1, \#, 1, \#)$.

All rows are covered by partial solution $\mathbf{x}^{(2)}$. Thus the least cost choice for remaining components is zero. We stop with heuristic optimum $\hat{\mathbf{x}} = (1, 0, 1, 0)$ at cost $6 + 15 = 21$. Notice that this solution is not as good as optimal $\mathbf{x}^* = (0, 0, 0, 1)$ with cost 20.

Need for Constructive Search

Many project selection and capital budgeting models are approached by greedy constructive heuristics such as the one just illustrated. Still, we have seen in Chapter 12 that more exact Branch and Bound, and Branch and Cut methods can also be effective.

The real need for constructive search methods becomes clear only with large, often nonlinear, highly combinatorial discrete models such as the KI truck routing application of Section 11.5 or cases where we need an answer fast.

Principle 15.2 In large, especially nonlinear, discrete models, or when time is limited, constructive search is often the only effective optimization-based approach to finding good solutions.

If tractable and strong relaxations are available, Branch and Bound is preferred. When natural neighborhoods exist, improving search can be effective. If neither applies, constructive heuristics provide the method of last resort.

Constructive Search of KI Truck Routing Application

To illustrate constructive search in such highly combinatorial cases, we will develop an algorithm for the KI routing application. Recall that stops $i = 1, \dots, 20$ are to be organized into the smallest possible list of routes j originating and terminating at a single central depot. Each route is then sequenced by an improving search to minimize travel distance. Figure 15.1 shows stop locations, and Table 15.2 provides the fractions of a load to be delivered at each stop.

Our constructive search for KI begins each route with a “seed” stop. We will choose the free stop farthest from the depot—number $i = 9$ in the first route. Figure 15.1 shows that the idea is to create a starting, out-and-back route with a general direction anchored by the seed location.

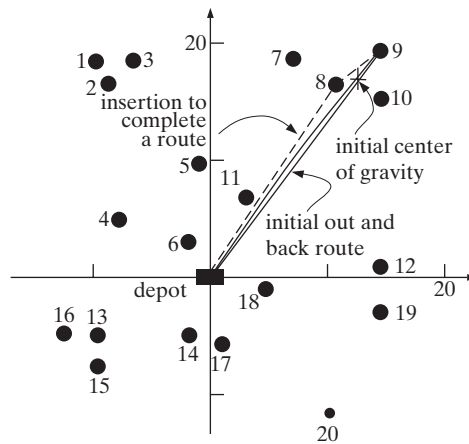


FIGURE 15.1 Locations and First Route in KI Application

TABLE 15.2 Fractions of Truckloads to Be Delivered in KI Application

Stop, i	Fraction, f_i	Stop, i	Fraction, f_i	Stop, i	Fraction, f_i	Stop, i	Fraction, f_i
1	0.25	6	0.70	11	0.21	16	0.38
2	0.33	7	0.28	12	0.68	17	0.26
3	0.39	8	0.43	13	0.16	18	0.29
4	0.40	9	0.50	14	0.19	19	0.17
5	0.27	10	0.22	15	0.22	20	0.31

As long as capacity remains in the truck for a route, we will insert new stops. As usual, our approach is greedy. A “center of gravity” is computed for stops so far fixed into the route, and the closest stop to that center still fitting on the truck is added to the route.

Figure 15.1 shows that a 1-stop route’s center of gravity is set arbitrarily 80% of the way from the depot to the seed location. This initial center of gravity for the route started by stop 9 has coordinates

$$0.8(x_9, y_9) = 0.8(15, 20) = (12, 16)$$

We hope to grow a cluster of stops near that point to form a compact route.

Stop 9 already uses $f_9 = 0.50$ truck. The nearest stop to the center of gravity is $i = 8$, with load $f_8 = 0.43$ within the remaining capacity. It becomes the first insertion.

After the route has more than one stop, the new selection is averaged into its center of gravity as

$$\frac{1}{2} [0.8(15, 20) + (11, 17)] = (11.5, 16.5)$$

However, the capacity fixed on this route already sums to

$$f_8 + f_9 = 0.50 + 0.43 = 0.93$$

and no remaining load will fit in the residual 0.07 truckload capacity. Route $j = 1$ is complete.

The next seed location is the farthest free stop from the depot—number $i = 10$. In turn, stop $i = 7$ is fixed in the route, then stop $i = 11$, and finally $i = 5$. Continuing in this way produces a total of 7 routes covering all stops.

15.2 IMPROVING SEARCH HEURISTICS FOR DISCRETE OPTIMIZATION INLPS

Many large combinatorial optimization models, especially INLPs with nonlinear objective functions, are too large for enumeration and lack strong relaxations that are tractable. Still, much can be done. Suitable adaptations of **improving search** methods introduced in Chapter 3 can often yield very effective **heuristic** algorithms. That is, we can still find good feasible solutions even though we will not be able to guarantee their optimality or even be sure about how close they come to optimal.

Rudimentary Improving Search Algorithm

Algorithm 15B shows a rudimentary adaptation of improving search to discrete models. Like the continuous cases of Chapter 3, the process begins with an initial feasible solution $\mathbf{x}^{(0)}$. Each iteration t considers neighbors of current solution $\mathbf{x}^{(t)}$ and tries to advance to one that is feasible and superior in objective value. If no feasible neighbor is improving, the process stops with **local optimum** and **heuristic optimum** $\mathbf{x}^{(t)}$.

Discrete Neighborhoods and Move Sets

What is new about the discrete form of improving search is that we must explicitly define the **neighborhood** of a current solution. Unlike the continuous case, where there are infinitely many points near a current solution, discrete search must advance to a binary or integer point. Explicit **move sets** (denoted \mathcal{M}) control what solutions are considered neighbors of current $\mathbf{x}^{(t)}$.

Principle 15.3 Improving searches over discrete variables define neighborhoods by specifying a move set \mathcal{M} of moves allowed. The current solution and all reachable from it in a single move $\Delta\mathbf{x} \in \mathcal{M}$ comprise its neighborhood.

ALGORITHM 15B: DISCRETE IMPROVING SEARCH

Step 0: Initialization. Choose any starting feasible solution $\mathbf{x}^{(0)}$, and set solution index $t \leftarrow 0$.

Step 1: Local Optimum. If no move $\Delta\mathbf{x}$ in move set \mathcal{M} is both improving and feasible at current solution $\mathbf{x}^{(t)}$, stop. Point $\mathbf{x}^{(t)}$ is a local optimum.

Step 2: Move. Choose some improving feasible move $\Delta\mathbf{x} \in \mathcal{M}$ as $\Delta\mathbf{x}^{(t+1)}$.

Step 3: Step. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$

Step 4: Increment. Increment $t \leftarrow t + 1$, and return to Step 1.

EXAMPLE 15.3: DEFINING MOVE SETS

Consider the discrete optimization model

$$\begin{aligned} \max \quad & 20x_1 - 4x_2 + 14x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + 4x_3 \leq 5 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

and assume that an improving search begins at $\mathbf{x}^{(0)} = (1, 1, 0)$.

(a) List all neighbors of $\mathbf{x}^{(0)}$ under move set

$$\mathcal{M} \triangleq \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \right\}$$

(b) Determine which members of the neighborhood are both improving and feasible.

Solution:

(a) Following principle [15.3](#), the neighbors of $\mathbf{x}^{(0)}$ under the specified move set are

$$\begin{aligned} (1, 1, 0) + (1, 0, 0) &= (2, 1, 0) \\ (1, 1, 0) + (-1, 0, 0) &= (0, 1, 0) \\ (1, 1, 0) + (0, 1, 0) &= (1, 2, 0) \\ (1, 1, 0) + (0, -1, 0) &= (1, 0, 0) \\ (1, 1, 0) + (0, 0, 1) &= (1, 1, 1) \\ (1, 1, 0) + (0, 0, -1) &= (1, 1, -1) \end{aligned}$$

(b) Of the neighbors in part (a), only $\mathbf{x} = (0, 1, 0)$, which has objective value -4 , and $\mathbf{x} = (1, 0, 0)$, which has objective value 20 , are feasible for all constraints of the model. Current point $\mathbf{x}^{(0)} = (1, 1, 0)$ has objective value 16 . Thus $\mathbf{x} = (1, 0, 0)$ is the only neighbor that is both improving and feasible (i.e., the one to which improving search would advance).

NCB Application Revisited

We will use the NCB application of Section 11.5 to illustrate improving search in discrete optimization. Recall that we seek a shortest-distance routing through 10 points in a printed circuit board that must be drilled. Table 15.3 details hole-to-hole travel distances.

TABLE 15.3 Distances between Holes in NCB Application

$i \backslash j$	1	2	3	4	5	6	7	8	9	10
1	—	3.6	5.1	10.0	15.3	20.0	16.0	14.2	23.0	26.4
2	3.6	—	3.6	6.4	12.1	18.1	13.2	10.6	19.7	23.0
3	5.1	3.6	—	7.1	10.6	15.0	15.8	10.8	18.4	21.9
4	10.0	6.4	7.1	—	7.0	15.7	10.0	4.2	13.9	17.0
5	15.3	12.1	10.6	7.0	—	9.9	15.3	5.0	7.8	11.3
6	20.0	18.1	15.0	15.7	9.9	—	25.0	14.9	12.0	15.0
7	16.0	13.2	15.8	10.0	15.3	25.0	—	10.3	19.2	21.0
8	14.2	10.6	10.8	4.2	5.0	14.9	10.3	—	10.2	13.0
9	23.0	19.7	18.4	13.9	7.8	12.0	19.2	10.2	—	3.6
10	26.4	23.0	21.9	17.0	11.3	15.0	21.0	13.0	3.6	—

For improving search it will be most convenient to employ the quadratic assignment formulation [11.27](#) of Section 11.5:

$$\begin{aligned} \min \sum_{k=1}^{10} \sum_{i=1}^{10} \sum_{j=1}^{10} d_{i,j} y_{k,i} y_{k+1,j} & \quad \text{(total distance)} \\ \sum_{i=1}^{10} y_{k,i} = 1 & \quad \text{for all } k = 1, \dots, 10 \quad \text{(some hole each } k) \end{aligned} \quad (15.2)$$

$$\sum_{k=1}^{10} y_{k,i} = 1 \quad \text{for all } i = 1, \dots, 10 \quad (\text{each } i \text{ assigned})$$

$$y_{k,i} = 0 \text{ or } 1 \quad \text{for all } k = 1, \dots, 10; \quad i = 1, \dots, 10$$

where

$$y_{k,i} \triangleq \begin{cases} 1 & \text{if } k\text{th hole drilled is } i \\ 0 & \text{otherwise} \end{cases}$$

and $y_{10+1,j}$ is understood to mean $y_{1,j}$ in objective function summations.

We will abuse notation in the usual way to think of solutions as vectors \mathbf{y} even though components have two subscripts. The NCB optimal solution \mathbf{y}^* , which is depicted in Figure 15.2, has length 81.8 inches and nonzero components

$$y_{1,1}^* = y_{2,3}^* = y_{3,6}^* = y_{4,10}^* = y_{5,9}^* = y_{6,5}^* = y_{7,8}^* = y_{8,7}^* = y_{9,4}^* = y_{10,2}^* = 1$$

We begin our improving searches with initial feasible solution

$$\mathbf{y}^{(0)} = (1, 0, \dots, 0; 0, 1, 0, \dots, 0; \dots; 0, \dots, 0, 1) \quad (15.3)$$

corresponding to $y_{1,1} = y_{2,2} = \dots = y_{10,10} = 1$. That is, hole 1 is drilled first, then hole 2, and so on. The total length is

$$\begin{aligned} & d_{1,2} + d_{2,3} + d_{3,4} + d_{4,5} + d_{5,6} + d_{6,7} + d_{7,8} + d_{8,9} + d_{9,10} + d_{10,1} \\ &= 3.6 + 3.6 + 7.1 + 7.0 + 9.9 + 25.0 + 10.3 + 10.2 + 3.6 + 26.4 \\ &= 106.7 \text{ inches} \end{aligned}$$

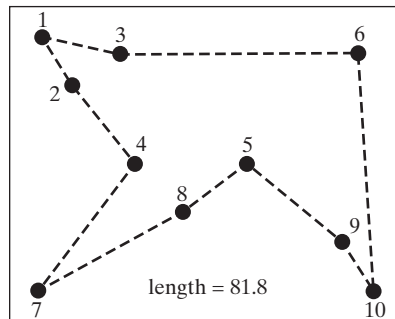


FIGURE 15.2 Optimal Drill Path in the NCB Example

Choosing a Move Set

The critical element of a discrete improving search heuristic is its move set. If it were possible, we would make every solution a neighbor of every other. Then the search would yield global optima because a stop implies that no solution at all is feasible and superior in objective value to the current.

In a practical search, however, we must accept much less.

Principle 15.4 The move set \mathcal{M} of a discrete improving search must be compact enough to be checked at each iteration for improving feasible neighbors.

On the other hand, we would not want too limited a move set.

Principle 15.5 The solution produced by a discrete improving search depends on the move set (or neighborhood) employed, with larger move sets generally resulting in superior local optima.

If the move set \mathcal{M} of a discrete improving search is too restrictive, very few solutions will be considered at each iteration, and poor quality local optima will result.

We will adopt one of the simplest move sets for our NCB application. Specifically, our \mathcal{M} will consist of **pairwise interchanges** swapping one position k with another ℓ . Corresponding move vectors $\Delta \mathbf{y}$ have two -1 components at deleted assignments, two $+1$ components at revised ones, and all other components 0. For example, if the hole in route position $k = 3$ is now number 7, and the hole in position $\ell = 5$ is now number 1, the corresponding interchange move direction has

$$\Delta y_{3,7} = -1, \quad \Delta y_{5,1} = -1, \quad \Delta y_{3,1} = +1, \quad \Delta y_{5,7} = +1$$

changing hole 1 to position 3 and hole 7 to position 5.

In all, this pairwise interchange \mathcal{M} contains a $\Delta \mathbf{y}$ for $(10 \cdot 9)/2 = 45$ choices of k and ℓ , each with $10 \cdot 9 = 90$ possible current hole assignment pairs—a total of $45 \cdot 90 = 4050$ moves. However, at any particular solution \mathbf{y} , only 45 moves interchanging its specific assignments lead to a feasible neighbor. In all searches in this section we adopt the one such move most improving the objective function.

EXAMPLE 15.4: COMPARING MOVE SETS

Return to the discrete model of Example 15.3 at initial point $\mathbf{x}^{(0)} = (1, 1, 0)$.

- (a) Show that $\mathbf{x}^{(0)}$ is not locally optimal under the move set of Example 15.3.
 (b) Show that $\mathbf{x}^{(0)}$ is locally optimal over smaller move set

$$\mathcal{M} \triangleq \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$$

Solution:

- (a) Example 15.3(b) established that $\mathbf{x} = (1, 0, 0)$ is a feasible neighbor of $\mathbf{x}^{(0)}$ with superior objective value. Thus $\mathbf{x}^{(0)}$ is not best in its neighborhood, and so not locally optimal. Algorithm 15B would advance to $\mathbf{x}^{(1)} = (1, 0, 0)$ and repeat.
 (b) Over this more restricted move set, neighbors are

$$(1, 1, 0) + (1, 0, 0) = (2, 1, 0)$$

$$(1, 1, 0) + (0, 1, 0) = (1, 2, 0)$$

$$(1, 1, 0) + (0, 0, 1) = (1, 1, 1)$$

None is feasible, so $\mathbf{x}^{(0)}$ is locally optimal.

Rudimentary Improving Search of the NCB Application

We can illustrate Algorithm 15B on our NCB application of Section 11.5. Table 15.4 displays results starting from initial solution $t = 0$. The objective function impact of the 45 feasible pairwise interchanges available at that solution are as follows:

$k \backslash \ell$	2	3	4	5	6	7	8	9	10
1	-1.9	-1.6	-0.3	6.5	-0.5	3.1	12.1	20.1	38.8
2		0.8	11.5	26.3	18.2	18.0	30.2	54.0	34.4
3			6.4	13.6	14.6	3.0	17.8	41.8	22.8
4				9.3	-7.1	1.6	5.1	19.5	13.0
5					-1.0	-2.3	4.8	11.5	8.2
6						10.0	-3.1	8.2	-0.6
7							-1.1	4.4	-2.1
8								18.3	-1.5
9									-0.6

For example, the best swap, position $k = 4$ for $\ell = 6$, implies savings of

$$d_{3,4} + d_{4,5} + d_{5,6} + d_{6,7} = 7.1 + 7.0 + 9.9 + 25.0 = 49 \text{ inches}$$

for delinking holes 4 and 6 from their current fourth and sixth tour positions, plus costs

$$d_{3,6} + d_{6,5} + d_{5,4} + d_{4,7} = 15.0 + 9.9 + 7.0 + 10.0 = 41.9 \text{ inches}$$

for relinking in their new positions. The net change is $41.9 - 49 = -7.1$ inches.

Table 15.4 displays the tour sequence resulting from this best interchange. It also details, swaps, move directions, and solutions visited to reach local optimality at $t = 5$. Local optimum \hat{y} has length 92.8 inches with

$$\hat{y}_{1,2} = \hat{y}_{2,1} = \hat{y}_{3,3} = \hat{y}_{4,6} = \hat{y}_{5,5} = \hat{y}_{6,4} = \hat{y}_{7,8} = \hat{y}_{8,9} = \hat{y}_{9,10} = \hat{y}_{10,7} = 1$$

Initial length 106.7 inches has been reduced by 13%, but locally optimal value 92.8 leaves us well above globally shortest tour length 81.8 inches.

TABLE 15.4 Rudimentary Improving Search of the NCB Application

t	Drill Sequence	Length	Interchange	Nonzero Move Components
0	1-2-3-4-5-6-7-8-9-10	106.7	4th for 6th	$\Delta y_{4,4} = \Delta y_{6,6} = -1, \Delta y_{4,6} = \Delta y_{6,4} = 1$
1	1-2-3-6-5-4-7-8-9-10	99.6	1st for 2nd	$\Delta y_{1,1} = \Delta y_{2,2} = -1, \Delta y_{1,2} = \Delta y_{2,1} = 1$
2	2-1-3-6-5-4-7-8-9-10	97.7	8th for 10th	$\Delta y_{8,8} = \Delta y_{10,10} = -1, \Delta y_{8,10} = \Delta y_{10,8} = 1$
3	2-1-3-6-5-4-7-10-9-8	96.0	7th for 10th	$\Delta y_{7,7} = \Delta y_{10,8} = -1, \Delta y_{7,8} = \Delta y_{10,7} = 1$
4	2-1-3-6-5-4-8-10-9-7	93.8	8th for 9th	$\Delta y_{8,10} = \Delta y_{9,9} = -1, \Delta y_{8,9} = \Delta y_{9,10} = 1$
5	2-1-3-6-5-4-8-9-10-7	92.8	Local optimum	

EXAMPLE 15.5: PERFORMING DISCRETE IMPROVING SEARCH

Consider the knapsack model (Section 11.2)

$$\begin{aligned} \max \quad & 18x_1 + 25x_2 + 11x_3 + 14x_4 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

under the **single complement** move set admitting moves that change one 0-component to 1 or one 1-component to 0. Compute an approximate optimal solution by discrete improving search Algorithm 15B beginning at $\mathbf{x}^{(0)} = (1, 0, 0, 0)$. If more than one neighbor is feasible and improving at any move, choose the one that improves the objective the most.

Solution: Feasible neighbors of the given $\mathbf{x}^{(0)}$ are $(0, 0, 0, 0)$, $(1, 0, 1, 0)$ and $(1, 0, 0, 1)$ with objective values 0, 29, and 32, respectively. The last improves the objective function the most, so the search advances to $\mathbf{x}^{(1)} = (1, 0, 0, 1)$.

At $\mathbf{x}^{(1)}$, feasible neighbors are $(0, 0, 0, 1)$ and $(1, 0, 0, 0)$. Since neither improves the objective function, the search stops with local optimum $\hat{\mathbf{x}} = \mathbf{x}^{(1)} = (1, 0, 0, 1)$ at objective value $\hat{v} = 32$.

Multistart Search

For our tiny NCB example we were able to know how far heuristic optimum sequence 2–1–3–6–5–4–8–9–10–7 is from globally optimal. In a larger instance it would be difficult to tell. Thus it is natural to at least try for further improvement.

One obvious approach is multistart—repeating the search from several different initial solutions $\mathbf{y}^{(0)}$.

Principle 15.6 **Multistart** or keeping the best of several local optima obtained from searches from different starting solutions is one way to better the heuristic solutions produced by improving.

Table 15.5 details search of the NCB application from 3 different starts. The first is the search of Table 15.4 with local minimum 92.8. Search 2 yields an improved local minimum with length 84.7 inches after one 1 iteration. Search 3 terminates with the same solution after a longer sequence. Multistart would report the best of these as an approximate optimum.

EXAMPLE 15.6: PERFORMING MULTISTART SEARCH

Return to the knapsack problem of Example 15.4:

$$\begin{aligned} \max \quad & 18x_1 + 25x_2 + 11x_3 + 14x_4 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

and its single complement search neighborhood. Perform a multistart discrete improving search starting from initial solutions (1, 0, 0, 0), (0, 1, 0, 0), and (0, 0, 1, 0).

Solution: Example 15.4 already established that $\mathbf{x}^{(0)} = (1, 0, 0, 0)$ leads to local maximum $\hat{\mathbf{x}} = (1, 0, 0, 1)$ with value $\hat{\nu} = 32$. Now restarting with $\mathbf{x}^{(0)} = (0, 1, 0, 0)$, the best feasible neighbor is $\mathbf{x}^{(1)} = (0, 1, 0, 1)$ with value 39.

This solution is locally optimal because no neighbor is both feasible and improving. Since it also improves on the current best (or incumbent) solution, our approximate optimum becomes $\hat{\mathbf{x}} = (0, 1, 0, 1)$, $\hat{\nu} = 39$.

The third search starts with $\mathbf{x}^{(0)} = (0, 0, 1, 0)$. Its feasible neighbor that most improves the objective function is $\mathbf{x}^{(1)} = (0, 1, 1, 0)$ with value 36. Again, Algorithm 15B stops at a local maximum. However, this one is not better than incumbent value $\hat{\nu} = 39$, so it is not retained.

TABLE 15.5 Multistart Search of the NCB Application

<i>t</i>	Drill Sequence	Length	Interchange	Nonzero Move Components
Search 1				
0	1-2-3-4-5-6-7-8-9-10	106.7	4th for 6th	$\Delta y_{4,4} = \Delta y_{6,6} = -1, \Delta y_{4,6} = \Delta y_{6,4} = 1$
1	1-2-3-6-5-4-7-8-9-10	99.6	1st for 2nd	$\Delta y_{1,1} = \Delta y_{2,2} = -1, \Delta y_{1,2} = \Delta y_{2,1} = 1$
2	2-1-3-6-5-4-7-8-9-10	97.7	8th for 10th	$\Delta y_{8,8} = \Delta y_{10,10} = -1, \Delta y_{8,10} = \Delta y_{10,8} = 1$
3	2-1-3-6-5-4-7-10-9-8	96.0	7th for 10th	$\Delta y_{7,7} = \Delta y_{10,8} = -1, \Delta y_{7,8} = \Delta y_{10,7} = 1$
4	2-1-3-6-5-4-8-10-9-7	93.8	8th for 9th	$\Delta y_{8,10} = \Delta y_{9,9} = -1, \Delta y_{8,9} = \Delta y_{9,10} = 1$
5	2-1-3-6-5-4-8-9-10-7	92.8	Local optimum	Incumbent value = 92.8
Search 2				
0	1-2-7-3-4-8-5-6-9-10	100.8	1st for 3rd	$\Delta y_{1,1} = \Delta y_{3,7} = -1, \Delta y_{1,7} = \Delta y_{3,1} = 1$
1	7-2-1-3-4-8-5-6-9-10	84.7	Local optimum	Incumbent value = 84.7
Search 3				
0	1-10-2-9-3-8-4-7-5-6	157.7	1st for 4th	$\Delta y_{1,1} = \Delta y_{4,9} = -1, \Delta y_{1,9} = \Delta y_{4,1} = 1$
1	9-10-2-1-3-8-4-7-5-6	97.5	6th for 8th	$\Delta y_{6,8} = \Delta y_{8,7} = -1, \Delta y_{6,7} = \Delta y_{8,8} = 1$
2	9-10-2-1-3-7-4-8-5-6	92.2	3rd for 6th	$\Delta y_{3,2} = \Delta y_{6,7} = -1, \Delta y_{3,7} = \Delta y_{6,2} = 1$
3	9-10-7-1-3-2-4-8-5-6	86.8	5th for 6th	$\Delta y_{5,3} = \Delta y_{6,2} = -1, \Delta y_{5,2} = \Delta y_{6,3} = 1$
4	9-10-7-1-2-3-4-8-5-6	86.0	4th for 5th	$\Delta y_{4,1} = \Delta y_{5,2} = -1, \Delta y_{4,2} = \Delta y_{5,1} = 1$
5	9-10-7-2-1-3-4-8-5-6	84.7	Local optimum	Incumbent value = 84.7

15.3 TABU AND SIMULATED ANNEALING METAHEURISTICS

Discrete improving search Algorithm 15B can be quite effective on many models, especially if it is applied several times with multistart. Still, the drive to an improving search local optimum can prove so narrowly focused that longer paths leading to better solutions are never investigated.

Metaheuristic algorithms apply one of several high-level strategies to obtain a better diversified search. This section presents two searches—**tabu search** and **simulated annealing**—that enrich the way moves are selected at each iteration

but continue the paradigm of moving from single solution to single solution. Then Section 15.4 extends even further to algorithms moving through evolving families of solutions.

Difficulty with Allowing Nonimproving Moves

An alternative to restarting improving search when no improving feasible move remains is to escape a local optimum by allowing nonimproving feasible moves. A few such retrograde moves might very well take the search to a region where progress can resume.

Unfortunately, this appealing strategy has a fatal flaw (unless new technology is introduced). Consider, for example, the locally optimal point where the first search of Table 15.5 terminated. No improving move was available at final drilling sequence 2–1–3–6–5–4–8–9–10–7 with value 92.8 inches. However, the best nonimproving move, which swaps the 9 in the eighth position for the 10 in the ninth, increases the objective by only 1.0 inch.

Why not take it and hope it leads to a better local optimum? Try. Sequence 2–1–3–6–5–4–8–10–9–7 results with length 93.8 inches. There is now a feasible interchange that improves (perhaps only one): swap back the 10 and 9 in the eighth and ninth tour positions to reduce length to 92.8 inches. Adopting that move returns us to exactly where we began, and the search will cycle forever.

Principle 15.7	Nonimproving moves will lead to infinite cycling of improving search unless some provision is added to prevent repeating solutions.
----------------	---

Tabu Search

Several schemes for incorporating nonimproving moves in improving search without undo cycling have proved effective on a variety of discrete optimization problems. One is called **tabu search** because it proceeds by classifying some moves “tabu” or forbidden. To be more specific,

Definition 15.8	Tabu search deals with cycling by temporarily forbidding moves that would return to a solution recently visited.
-----------------	--

The effect is to prevent short-term cycling, although solutions can repeat over a longer period.

Algorithm 15C gives a formal statement of this tabu modification to improving search Algorithm 15B. A **tabu list** records forbidden moves, and each iteration chooses a non-tabu feasible move. After each step, a collection of moves that includes any returning immediately to the previous point is added to the tabu list. No such move is allowed for a few iterations, but eventually all are removed from the tabu list and again available.

ALGORITHM 15C: TABU SEARCH

Step 0: Initialization. Choose any starting feasible solution $\mathbf{x}^{(0)}$ and an iteration limit t_{\max} . Then set incumbent solution $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(0)}$ and solution index $t \leftarrow 0$. No moves are tabu.

Step 1: Stopping. If no non-tabu move $\Delta\mathbf{x}$ in move set \mathcal{M} leads to a feasible neighbor of current solution $\mathbf{x}^{(t)}$, or if $t = t_{\max}$, then stop. Incumbent solution $\hat{\mathbf{x}}$ is an approximate optimum.

Step 2: Move. Choose some non-tabu feasible move $\Delta\mathbf{x} \in \mathcal{M}$ as $\Delta\mathbf{x}^{(t+1)}$.

Step 3: Step. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$

Step 4: Incumbent Solution. If the objective function value of $\mathbf{x}^{(t+1)}$ is superior to that of incumbent solution $\hat{\mathbf{x}}$, replace $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(t+1)}$.

Step 5: Tabu List. Remove from the list of tabu of forbidden moves any that have been on it for a sufficient number of iterations, and add a collection of moves that includes any returning immediately from $\mathbf{x}^{(t+1)}$ to $\mathbf{x}^{(t)}$.

Step 6: Increment. Increment $t \leftarrow t + 1$, and return to Step 1.

Since steps may either improve or degrade the objective function value, an incumbent solution $\hat{\mathbf{x}}$ tracks the best feasible point found so far. When the search stops, which is usually when user-supplied iteration limit t_{\max} is reached, incumbent $\hat{\mathbf{x}}$ is reported as an approximate optimum.

Tabu Search of the NCB Application

Table 15.6 illustrates an implementation of tabu search on the NCB application. Figure 15.3 tracks the objective function values of points encountered.

The initial solution and move set of this tabu search are identical to those of the ordinary improving search in Table 15.4. This time, however, $t_{\max} = 50$ points were visited, with search proceeding to the feasible, non-tabu neighbor with best objective function value, whether or not it improves.

The design of tabu searches requires some judgment in deciding what moves to make tabu at each iteration. Too few will lead to cycling; too many inordinately restricts the search.

Table 15.6's search of the NCB application fixed the first of each two positions interchanged for a period of 6 iterations. For example, after the first interchange of fourth and sixth positions, no move again changing the fourth position was allowed for 6 steps. Tabu positions are underlined. Such a policy maintains a relatively rich set of available moves, yet prevents immediate reverses.

Figure 15.3 shows clearly how tabu search improves as long as improving moves are available, then begins controlled wandering. In the beginning the incumbent solution $\hat{\mathbf{y}}$ improves rapidly. (Incumbent values are recorded in Table 15.6.) Later, progress slows. Still, the global optimum was discovered on iteration $t = 44$.

TABLE 15.6 Tabu Search of NCB Application

<i>t</i>	Drill Sequence ^a	Length	Incumbent	Interchange	Δ Obj.
0	1-2-3-4-5-6-7-8-9-10	106.7	106.7	4th for 6th	7.1
1	1-2-3-6-5-4-7-8-9-10	99.6	99.6	1st for 2nd	1.9
2	2-1-3-6-5-4-7-8-9-10	97.7	97.7	8th for 10th	1.7
3	2-1-3-6-5-4-7-10-9-8	96.0	96.0	7th for 10th	2.2
4	2-1-3-6-5-4-8-10-9-7	93.8	93.8	6th for 10th	-2.3
5	2-1-3-6-5-7-8-10-9-4	96.1	93.8	5th for 10th	.1
6	2-1-3-6-4-7-8-10-9-5	96.2	93.8	4th for 10th	2.9
7	2-1-3-5-4-7-8-10-9-6	93.3	93.3	1st for 3rd	1.6
8	3-1-2-5-4-7-8-10-9-6	91.7	91.7	8th for 9th	-0.2
9	3-1-2-5-4-7-8-9-10-6	91.9	91.7	2nd for 3rd	-1.7
10	3-2-1-5-4-7-8-9-10-6	93.6	91.7	6th for 7th	-3.2
11	3-2-1-5-4-8-7-9-10-6	96.8	91.7	5th for 7th	-3.0
12	3-2-1-5-7-8-4-9-10-6	99.8	91.7	4th for 7th	15.9
13	3-2-1-4-7-8-5-9-10-6	83.9	83.9	1st for 3rd	-2.1
14	1-2-3-4-7-8-5-9-10-6	86.0	83.9	8th for 9th	-0.5
15	1-2-3-4-7-8-5-10-9-6	86.5	83.9	2nd for 3rd	-0.8
⋮	⋮	⋮	⋮	⋮	⋮
40	1-2-7-8-5-9-10-4-6-3	96.3	83.9	1st for 2nd	-1.3
41	2-1-7-8-5-9-10-4-6-3	97.6	83.9	8th for 9th	9.9
42	2-1-7-8-5-9-10-6-4-3	87.7	83.9	2nd for 9th	0.9
43	2-4-7-8-5-9-10-6-1-3	86.8	83.9	9th for 10th	5.0
44	2-4-7-8-5-9-10-6-3-1	81.8	81.8	6th for 7th	-0.5
45	2-4-7-8-5-10-9-6-3-1	82.3	81.8	5th for 7th	-3.1
46	2-4-7-8-9-10-5-6-3-1	85.4	81.8	1st for 10th	-2.1
47	1-4-7-8-9-10-5-6-3-2	87.5	81.8	7th for 8th	0.7
48	1-4-7-8-9-10-6-5-3-2	86.8	81.8	2nd for 3rd	0.1
49	1-7-4-8-9-10-6-5-3-2	86.7	81.8	9th for 10th	-3.0
50	1-7-4-8-9-10-6-5-2-3	89.7	81.8	Stop	

^aUnderlining indicates components not allowed to change.

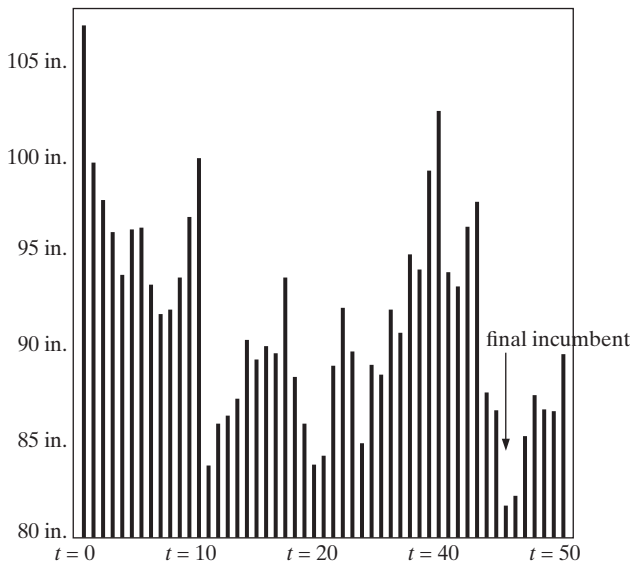


FIGURE 15.3 Solution Values in Tabu Search of NCB Application

Naturally, it would be the final incumbent reported when iteration limit $t_{\max} = 50$ was reached.

Results will vary with models and details of tabu policy, but there is good reason to believe that such performance is typical. Suitable implementations of the tabu variation on improving search can greatly enhance the quality of heuristic solutions obtained.

EXAMPLE 15.7: APPLYING TABU SEARCH

Return to the knapsack problem

$$\begin{aligned} \max \quad & 18x_1 + 25x_2 + 11x_3 + 14x_4 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + x_3 + x_4 \leq 3 \\ & x_4, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

of Examples 15.5 and 16.6 and assume that we will use the same single-complement move set.

(a) Explain how making it tabu to complement any component x_j for the next 2 iterations after it is changed by the search prevents short-term cycling.

(b) Begin from solution $\mathbf{x}^{(0)} = (1, 0, 0, 0)$ and use this tabu rule in executing Algorithm 15C through $t_{\max} = 5$ steps.

Solution:

(a) Once a component is changed for 0 to 1, or vice versa, the only way to return to the immediately preceding solution is to complement the same component again.

(b) The required search is summarized in the following table:

t	$\mathbf{x}^{(t)}$	Value	Incumbent Value	Complemented	$\Delta \text{obj.}$
0	(1, 0, 0, 0)	18	18	$j = 4$	14
1	(1, 0, 0, <u>1</u>)	32	32	$j = 1$	-18
2	(<u>0</u> , 0, 0, 1)	14	32	$j = 2$	25
3	(<u>0</u> , <u>1</u> , 0, 1)	39	39	$j = 4$	-14
4	(0, <u>1</u> , 0, <u>0</u>)	25	39	$j = 3$	11
5	(0, 1, <u>1</u> , <u>0</u>)	36	39	Stop	

Each iteration begins by selecting an x_j to complement that is not among those marked tabu (underlined) and does preserve feasibility. The best such move produces the next solution. The result is also saved as a new incumbent if it is superior to any feasible solution encountered so far. Computation stops with approximate (here exact) optimum $\hat{\mathbf{x}} = (0, 1, 0, 1)$ at $t = t_{\max} = 5$.

Simulated Annealing Search

Another method of introducing nonimproving moves into improving search is termed **simulated annealing** because of its analogy to the annealing process of slowly cooling metals to improve strength.

Definition 15.9 Simulated annealing algorithms control cycling by accepting nonimproving moves according to probabilities tested with computer-generated random numbers.

Improving and accepted nonimproving moves are pursued; rejected ones are not.

Algorithm 15D provides details. The move selection process at each iteration begins with random choice of a provisional feasible move, totally ignoring its objective function impact. Next, the net objective function improvement Δobj (nonpositive for nonimproving moves) is computed for the chosen move. The move is always accepted if it improves ($\Delta\text{obj} > 0$), and otherwise

$$\text{probability of acceptance} = e^{\Delta\text{obj}/q} \quad (15.4)$$

That is, all improving moves and some nonimproving ones are accepted. The probability of accepting a nonimproving move declines as net objective improvement Δobj becomes more negative.

ALGORITHM 15D: SIMULATED ANNEALING SEARCH

Step 0: Initialization. Choose any starting feasible solution $\mathbf{x}^{(0)}$, an iteration limit t_{\max} , and a relatively large initial temperature $q > 0$. Then, set incumbent solution $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(0)}$ and solution index $t \leftarrow 0$.

Step 1: Stopping. If no move $\Delta\mathbf{x}$ in move set \mathcal{M} leads to a feasible neighbor of current solution $\mathbf{x}^{(t)}$, or if $t = t_{\max}$, then stop. Incumbent solution $\hat{\mathbf{x}}$ is an approximate optimum.

Step 2: Provisional Move. Randomly choose a feasible move $\Delta\mathbf{x} \in \mathcal{M}$ as a provisional $\Delta\mathbf{x}^{(t+1)}$, and compute the (possibly negative) net objective function improvement Δobj for moving from $\mathbf{x}^{(2)} = (0, 0, 0, 1)$ to $(\mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)})$ (increase for a maximize, decrease for a minimize).

Step 3: Acceptance. If $\Delta\mathbf{x}^{(t+1)}$ improves, or with probability $e^{\Delta\text{obj}/q}$ if $\Delta\text{obj} \leq 0$, accept $\Delta\mathbf{x}^{(t+1)}$ and update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta\mathbf{x}^{(t+1)}$$

Otherwise, return to Step 2.

Step 4: Incumbent Solution. If the objective function value of $\mathbf{x}^{(t+1)}$ is superior to that of incumbent solution $\hat{\mathbf{x}}$, replace $\hat{\mathbf{x}} \leftarrow \mathbf{x}^{(t+1)}$.

Step 5: Temperature Reduction. If a sufficient number of iterations have passed since the last temperature change, reduce temperature q .

Step 6: Increment. Increment $t \leftarrow t + 1$, and return to Step 1.

Parameter q in (15.4) is a **temperature** controlling the randomness of the search. If q is large, the exponent in (15.4) approaches 0, implying that the probability of accepting nonimproving moves approximates $e^0 = 1$. If q is small, the probability of accepting very bad moves decreases dramatically. Simulated annealing searches usually begin with q relatively large and decrease it every few iterations.

As with tabu and other searches that can make nonimproving moves, an incumbent solution $\hat{\mathbf{x}}$ must be maintained to keep track of the best feasible solution found so far. When computation stops, $\hat{\mathbf{x}}$ is output as an approximately optimal solution.

Simulated Annealing Search of NCB Application

Table 15.7 provides an abridged summary of a simulated annealing search of our NCB application. Figure 15.4 shows the complete history of accepted solutions through iteration limit $t_{\max} = 50$. Like all other searches of this section, move set \mathcal{M} included all single interchanges, and initial solution $\mathbf{y}^{(0)}$ is the one in (15.3).

Temperatures in this simulated annealing example began at $q = 5.0$. Every 10 iterations they were reduced by a factor of 0.8, so that

$$\begin{aligned} q &= 5.0 && \text{for iterations } t = 0, \dots, 9 \\ q &= 0.8(5.0) = 4.0 && \text{for iterations } t = 10, \dots, 19 \\ q &= 0.8(4.0) = 3.2 && \text{for iterations } t = 20, \dots, 29 \\ &\vdots && \end{aligned}$$

The first few iterations of Table 15.7 show most randomly chosen moves being accepted. Interchange of the seventh and tenth tour positions improves on solution $\mathbf{y}^{(0)}$ and is accepted immediately to produce $\mathbf{y}^{(1)}$. The first move generated from solution $\mathbf{y}^{(1)}$ increases length by 20.1 inches. Thus its probability of acceptance was

$$e^{-20.1/5} \approx 0.018$$

Not surprisingly, it was rejected.

The next try produced a move that increased length by only 3.1 inches. With better probability

$$e^{-3.1/5} \approx 0.538$$

it was adopted.

The latter part of Table 15.7 shows the impact of reducing temperature q as the search proceeds. More and more nonimproving moves are rejected, because the probability of acceptance has declined with q .

Results in Figure 15.4 show the wide-ranging evolution of the full simulated annealing search. As with tabu, the final incumbent solution happens to be the global optimum, but this is not guaranteed.

Again this behavior is typical of reported simulated annealing applications, although many more iterations would normally be required. Suitable implementations of the simulated annealing variation on improving search can greatly enhance the quality of heuristic solutions obtained.

TABLE 15.7 Simulated Annealing Search of NCB Application

<i>t</i>	Drill Sequence	Length	Incumbent	Temp <i>q</i>	Interchange	Δ Obj.	Outcome
0	1-2-3-4-5-6-7-8-9-10	106.7	106.7	5.00	7th for 10th	2.1	Accepted
1	1-2-3-4-5-6-10-8-9-7	104.6	104.6	5.00	1st for 9th	-20.1	Rejected
					1st for 4th	-3.1	Accepted
2	4-2-3-1-5-6-10-8-9-7	107.7	104.6	5.00	7th for 10th	1.3	Accepted
3	4-2-3-1-5-6-7-8-9-10	106.4	104.6	5.00	5th for 6th	5.0	Accepted
4	4-2-3-1-6-5-7-8-9-10	101.4	101.4	5.00	9th for 10th	0.3	Accepted
5	4-2-3-1-6-5-7-8-10-9	101.1	101.1	5.00	9th for 10th	-0.3	Accepted
6	4-2-3-1-6-5-7-8-9-10	101.4	101.1	5.00	2nd for 7th	-12.9	Rejected
					1st for 7th	3.6	Accepted
7	7-2-3-1-6-5-4-8-9-10	97.8	97.8	5.00	6th for 7th	-6.6	Rejected
					7th for 8th	-1.7	Accepted
8	7-2-3-1-6-5-8-4-9-10	99.5	97.8	5.00	2nd for 5th	-9.0	Rejected
					2nd for 4th	-0.9	Accepted
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
40	3-1-2-4-7-8-5-9-10-6	81.8	81.8	2.05	6th for 10th	-13.4	Rejected
					5th for 6th	-4.5	Rejected
					2nd for 5th	-24.2	Rejected
					9th for 10th	-15.3	Rejected
					9th for 10th	-15.3	Rejected
					8th for 9th	-0.5	Accepted
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
46	3-1-2-4-7-8-9-10-5-6	85.4	81.8	2.05	5th for 10th	-16.5	Rejected
					8th for 10th	-15.3	Rejected
					8th for 10th	-15.3	Rejected
					3rd for 6th	-20.8	Rejected
					2nd for 9th	-39.2	Rejected
					2nd for 9th	-39.2	Rejected
					4th for 9th	-22.5	Rejected
					3rd for 9th	-32.2	Rejected
					5th for 7th	-21.3	Rejected
					1st for 3rd	-3.8	Rejected
					2nd for 8th	-59.6	Rejected
					9th for 10th	0.7	Accepted
47	3-1-2-4-7-8-9-10-6-5	84.7	81.8	2.05	3rd for 8th	-52.6	Rejected
					8th for 9th	-9.8	Rejected
					4th for 5th	-0.7	Rejected
					7th for 10th	-12.4	Rejected
					7th for 10th	-12.4	Rejected
					5th for 10th	-12.0	Rejected
					1st for 7th	-34.0	Rejected
					4th for 10th	-13.3	Rejected
					2nd for 10th	-18.6	Rejected
					6th for 10th	-7.8	Rejected
					6th for 7th	-18.3	Rejected
					3rd for 5th	-12.7	Rejected
					9th for 10th	-0.7	Accepted
48	3-1-2-4-7-8-9-10-5-6	85.4	81.8	2.05	9th for 10th	0.7	Accepted
49	3-1-2-4-7-8-9-10-6-5	84.7	81.8	2.05	7th for 8th	0.2	Accepted
50	3-1-2-4-7-8-10-9-6-5	84.5	81.8	1.64	Stop		

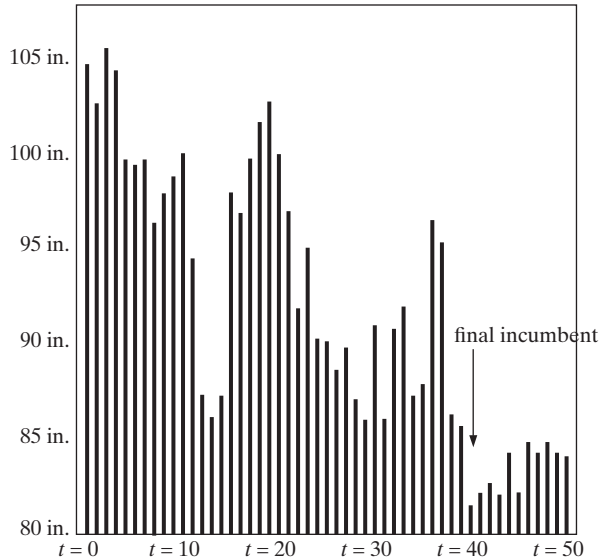


FIGURE 15.4 Solution Values in Simulated Annealing Search of NCB Application

EXAMPLE 15.8: APPLYING SIMULATED ANNEALING

Return again to the knapsack model of Example 15.5 to 15.7 with its single-complement move set and initial solution $\mathbf{x}^{(0)} = (1, 0, 0, 0)$. Using temperature $q = 10$, apply simulated annealing Algorithm 15D through $t_{\max} = 3$ steps. Where random decisions are required, use the following random numbers (uniform between 0 and 1): 0.72, 0.83, 0.33, 0.41, 0.09, 0.94.

Solution: Required computations are summarized in the following table.

t	$\mathbf{x}^{(t)}$	Value	Incumbent Value	q	Complement	Δobj	Outcome
0	(1, 0, 0, 0)	18	18	10	$j = 4$	14	Accepted
1	(1, 0, 0, 1)	32	32	10	$j = 4$	-14	Rejected
					$j = 1$	-18	Accepted
2	(0, 0, 0, 1)	14	32	10	$j = 3$	11	Accepted
3	(0, 0, 1, 1)	25	32	10	Stop		

The process begins by randomly selecting between feasible moves complementing $j = 3$ and $j = 4$ at $\mathbf{x}^{(0)} = (1, 0, 0, 0)$. Since the first random number 0.72 is in the upper half of interval $[0, 1]$, $j = 4$ is provisionally selected. The corresponding move is improving, so the search advances to $\mathbf{x}^{(1)} = (1, 0, 0, 1)$ with value 32.

Feasible complements at $\mathbf{x}^{(1)}$ are $j = 1$ and $j = 4$. The next random number 0.83 selects $j = 4$. The corresponding move has $\Delta \text{obj} = -14$, so we test whether random number 0.33 is at most probability

$$e^{\Delta\text{obj}/q} = e^{-14/10} \approx 0.247$$

It is not, and the provisional move is rejected.

The next randomly generated feasible move complements $j = 1$ with $\Delta\text{obj} = -18$. This time

$$e^{\Delta\text{obj}/q} = e^{-18/10} \approx 0.165 \geq 0.09$$

We accept the move and advance to $\mathbf{x}^{(2)} = (0, 0, 0, 1)$.

Three moves are feasible at $\mathbf{x}^{(2)}$ and complementation of $j = 3$ is selected by random number 0.94. The corresponding improving move advances to $\mathbf{x}^{(3)} = (0, 0, 1, 1)$.

We now stop at $t = t_{\max} = 3$ and report incumbent solution $\hat{\mathbf{x}} = (1, 0, 0, 1)$ with value 32.

15.4 EVOLUTIONARY METAHEURISTICS AND GENETIC ALGORITHMS

Evolutionary Metaheuristics broaden the reach of heuristic search beyond any sequence of single solutions to an evolving population of possibilities improved through time by mechanisms that mirror the biological process of natural selection. The best known family of such population metaheuristics, **genetic algorithms**, is introduced in this section.

Definition 15.10 Genetic algorithms evolve good heuristic optima by operations combining members of an improving **population** of individual solutions.

The best single solution encountered so far will always be part of the population (in the variant discussed here), but each **generation** will also include a spectrum of other solutions. Ideally, all will be feasible, and some may be nearly as good in the objective function as the best. Others may have quite poor solution values.

New solutions are created by combining pairs of individuals in the population. Local optima are less frequent because this combining process does not center entirely on the best current solution.

Crossover Operations in Genetic Algorithms

The standard genetic algorithm method for combining solutions of the population is known as crossover.

Definition 15.11 **Crossover** combines a pair of “parent” solutions to produce a pair of “children” by breaking both parent vectors at the same point and reassembling the first part of one parent solution with the second part of the other, and vice versa.

We can illustrate with two binary solution vectors $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$:

$$\begin{aligned}\mathbf{x}^{(1)} &= (1, 0, 1, 1, 0, \mid 0, 1, 0, 0) \\ \mathbf{x}^{(2)} &= (0, 1, 1, 0, 1, \mid 1, 0, 0, 1)\end{aligned}$$

Crossover after component $j = 5$ leads to children

$$\begin{aligned}\mathbf{x}^{(3)} &= (1, 0, 1, 1, 0, \mid 1, 0, 0, 1) \\ \mathbf{x}^{(4)} &= (0, 1, 1, 0, 1, \mid 0, 1, 0, 0)\end{aligned}$$

One child $\mathbf{x}^{(3)}$ combines the initial part of $\mathbf{x}^{(1)}$ with the final part of $\mathbf{x}^{(2)}$. Child $\mathbf{x}^{(4)}$ does just the opposite. Both become members of the new population, and the search continues.

There is no guarantee that crossover's rather arbitrary manipulation of parent solutions will yield improvement. Still, it does lead to fundamentally new solutions that preserve significant parts of their parents. Experience shows that this is often enough to produce very good results.

Managing Genetic Algorithms with Elites, Immigrants, Mutations, and Crossovers

Many variations on the basic genetic algorithm strategy have been employed successfully in particular applications, including many alternatives to standard crossover operations [15.11]. Principal differences in the various implementations concern how to select pairs of current solutions to produce new ones via crossover, how to decide which new and/or old solutions will survive in the next population, and how to maintain diversity in the population as the search advances from generation to generation. The only requirement is that better solutions have greater chance to breed.

In this brief introduction we consider only a single elitest method of population management. Each new generation will be composed of a combination of elite, immigrant, mutated, and crossover solutions.

Definition 15.12 | The **elitest** strategy for implementation of genetic algorithms forms each new generation as a mixture of **elite** (best) solutions held over from the previous generation, **immigrant** solutions added arbitrarily to increase diversity, random **mutation** of other solutions and children of crossover operations on nonoverlapping pairs of solutions in the previous population.

Maintenance of the elite solutions from the preceding generation assures that the best solutions known so far will remain in the population and have more opportunities to produce offspring. Addition of new immigrant solutions and random mutations of existing ones will help to maintain diversity as solutions are combined. The bulk of the new solutions will be the product of crossovers, with elites in the preceding population allowed to serve as parents. Algorithm 15E details a full procedure.

ALGORITHM 15E: GENETIC ALGORITHM SEARCH

Step 0: Initialization. Choose a population size p , initial starting feasible solutions $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}$, a generation limit t_{\max} , and population subdivisions p_e for elites, p_i for immigrants, and p_c for crossovers. Also set generation index $t \leftarrow 0$.

Step 1: Stopping. If $t = t_{\max}$, stop and report the best solution of the current population as an approximate optimum.

Step 2: Elite. Initialize the population of generation $t + 1$ with copies of the p_e best solutions in the current generation.

Step 3: Immigrants/mutations. Arbitrarily choose p_i new immigrant feasible solutions, or mutations of existing ones and include them in the $t + 1$ population.

Step 4: Crossovers. Choose $p_c/2$ nonoverlapping pairs of solutions from the generation t population, and execute crossover on each pair at an independently chosen random cut point to complete the generation $t + 1$ population.

Step 5: Increment. Increment $t \leftarrow t + 1$, and return to Step 1.

Solution Encoding for Genetic Algorithm Search

Just as design of an ordinary improving search requires careful construction of a move set (principle 15.5), implementations of genetic algorithms require judicious choice of a scheme for encoding solutions in a vector. To see the difficulty, return to our NCB drilling application.

If solutions are encoded merely by displaying the drilling sequence, two that might come together as crossover parents would be

$$(3, 1, 2, 4, 7, 8, 5, 9, 10, 6) \quad \text{and} \quad (7, 2, 3, 1, 6, 5, 8, 4, 9, 10)$$

Crossing over the solutions after component $j = 6$ would yield children

$$(3, 1, 2, 4, 7, 8, 8, 4, 9, 10) \quad \text{and} \quad (7, 2, 3, 1, 6, 5, 5, 9, 10, 6)$$

Neither is a feasible drilling sequence because some holes are visited more than once and some never. A poor choice of solution encoding has made it almost impossible for crossover to produce useful new solutions.

Principle 15.13 | Effective genetic algorithm search requires a choice for encoding problem solutions that often, if not always, preserves solution feasibility after crossover.

We can obtain a better encoding in the NCB application by a technique known as **random keys**.

Definition 15.14 | Random-key methods for genetic algorithms encode solutions indirectly as a sequence of random numbers. Then solutions are recovered indirectly by making the first encoded index that of the lowest random number, the second the next lowest, and so on.

To illustrate, consider the following encoding of a drilling sequence in the NCB application

$$(0.32, 0.56, 0.91, 0.44, 0.21, 0.68, 0.51, 0.07, 0.12, 0.39)$$

The drilling sequence implied is the one obtained by aligning hole 1 with the lowest random component, hole 2 at the next lowest, and so on. That is, the given random vector encodes the drilling sequence

$$(4, 8, 10, 6, 3, 9, 7, 1, 2, 5)$$

But notice that crossover on two vectors of random numbers produces two others. Thus every crossover operation of these random key encodings will yield two new feasible solutions. Also, it is very easy to generate arbitrary new solutions for the initial population and immigration simply by generating random vectors of random numbers.

Genetic Algorithm Search of NCB Application

Figure 15.5 shows objective function values in a 30-generation Algorithm 15E search of our NCB application. The population had size 20, with the initial population generated randomly. Following principle [15.11](#), each new generation contained the

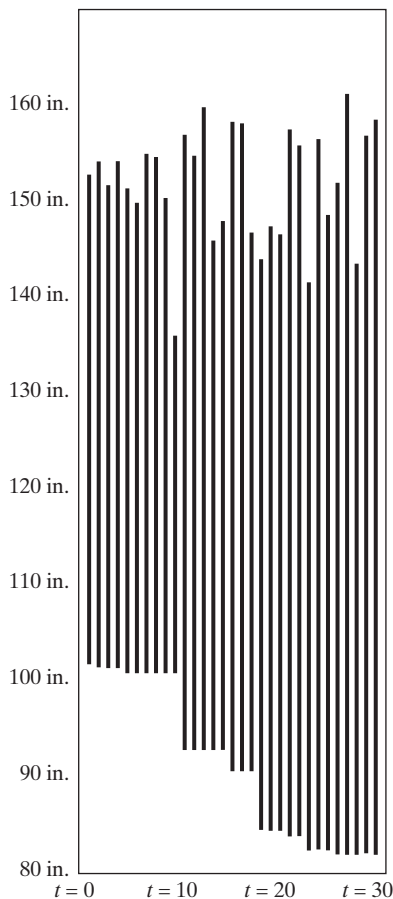


FIGURE 15.5 Genetic Search Population Ranges for NCB Application

6 best (elite) solutions of the preceding generation, 4 randomly generated immigrant solutions, and 10 solutions obtained from crossover of 5 pairs of parents in the preceding generation. All solutions were encoded by random keys.

Bars in Figure 15.5 extend from the lowest to the highest route length of solutions in each population. Notice that the lowest ones converge systematically to 81.8, which we know is the optimal value of this example. Still, the population always contains a variety of solutions, some with rather poor objective values. This diversity makes it possible for the search to range into distinctly new parts of the feasible space as it seeks improved solutions.

EXERCISES

15-1 Consider solving (approximately) the following knapsack problem by constructive search Algorithm 15A.

$$\begin{aligned} \max \quad & 11x_1 + 1x_2 + 9x_3 + 17x_4 \\ \text{s.t.} \quad & 9x_1 + 2x_2 + 7x_3 + 13x_4 \leq 17 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

- ✓ (a) Determine a global optimum by inspection.
- ✓ (b) Explain why it is reasonable to fix variables in decreasing order of ratio

$$\frac{\text{objective coefficient}}{\text{constraint coefficient}}$$

- ✓ (c) Apply constructive search Algorithm 15A to construct an approximate solution choosing variables to fix in this ratio sequence.

15-2 Do Exercise 15-1 for the knapsack model (this time with increasing ratio order)

$$\begin{aligned} \min \quad & 55x_1 + 150x_2 + 54x_3 + 180x_4 \\ \text{s.t.} \quad & 25x_1 + 30x_2 + 18x_3 + 45x_4 \geq 40 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

15-3 Now consider knapsack instance

$$\begin{aligned} \max \quad & 30x_1 + 20x_2 + 20x_3 \\ \text{s.t.} \quad & 21x_1 + 20x_2 + 20x_3 \leq 40 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

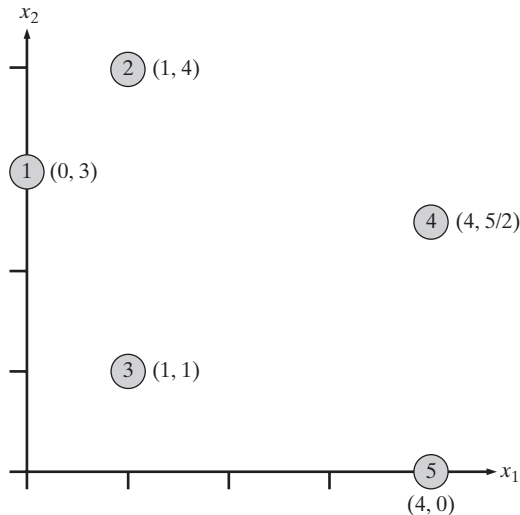
- (a) Determine a global optimum by inspection
- (b) Now apply Constructive Search Algorithm 15A with the greedy ratio of Exercise 15-1(b) to compute an approximate optimum.
- (c) Compare the results of (a) and (b), and discuss how the local nature of greedy ratio search led part (b) to a poorer solution.

15-4 Do Exercise 15-3 for the knapsack instance

$$\begin{aligned} \min \quad & 10x_1 + 10x_2 + 3x_3 \\ \text{s.t.} \quad & 10x_1 + 10x_2 + 6x_3 \geq 20 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

15-5 Recall the Traveling Salesman Problem (TSP) of Section 11.5, which seeks a minimum total length cycle (or tour) of a given graph $G(V, E)$ that includes all nodes of V . One constructive algorithm for the problem is the Nearest Neighbor method, which begins at one of the given points, then successively extends the current partial tour to the nearest-in-distance unvisited point of V , closing the tour when all points have been included.

- (a) Justify the greedy standard of the algorithm to proceed to the nearest unvisited point.
- (b) Now consider the following (TSP) instance (from Exercise 14-9) over the 5 points shown, assuming edges connect all pairs of nodes and distances are Euclidean.



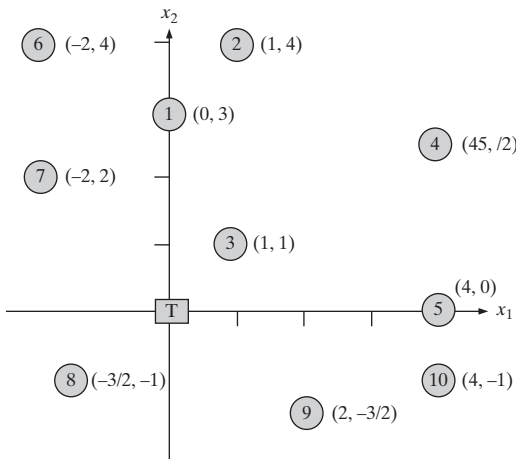
Use inspection to justify assuming an optimal tour is 1–2–4–5–3–1.

- (c) Next, solve the instance approximately by the Nearest Neighbor method, starting from node 1.
- (d) Discuss the advantages and disadvantages of employing a simple constructive method like Nearest Neighbor versus the quality of the solution produced.

15-6 Return to the (TSP) instance of Exercise 15-5(b). This time construct an approximately optimal tour by the Greedy Insertion strategy used on the KI Truck Routing application in Figure 15.1. Specifically begin with a 2-point tour 1–5–1. Then at each iteration insert to unvisited point closest to the average of coordinates for nodes included so far.

- (a) Justify the greedy standard of the algorithm to insert the nearest unvisited point.
- (b) Compare your results to the optimum of Exercise 15-5(b) and the heuristic solution 15-5(c).
- (c) Discuss the advantages and disadvantages of employing a Greedy Insertion constructive method versus the tour extension strategy of Nearest Neighbor.

15-7 Consider a simplified Vehicle Routing Problem (VRP) over the customer sites in the following plot (refer to Section 11.5).



Two 5-customer routes are to be designed, both originating and returning to terminal site

$x_1 = x_2 = 0$. All point-to-point distances are assumed Euclidean, and the goal is to minimize the total travel length of the two routes.

- (a) Parallel the constructive solution of routes in the KI Trucking application of Section 15.1 to construct two approximately optimal routes. Begin with seed routes to sites 6 and 10. Then iteratively insert all other points in order of their distance from centers of gravity computed as the average coordinates of sites already inserted in each of the two emerging routes.
- (b) Comment on the quality of the two ultimate routes and whether similar constructive techniques could be used in more realistic (VRP) applications.

15-8 Classic Bin Packing (BP) considers the task of packing a collection of items $j = 1, \dots, n$ of varying sizes a_j into a minimum number N of bins of capacity b . The first-fit algorithm to solve instances approximately begins with no bins open. Then it takes items in arbitrary sequence, considering possible bins for each in the order they were opened. The item is placed in the first bin that can accommodate it, or if none does, a new bin is opened and the item placed there.

- ✓ (a) Taking items in subscript sequence, apply this first-fit heuristic to an instance with $n = 12$ items of sizes $a_j = 12, 14, 9, 19, 2, 4, 13, 8, 8, 10, 13,$ and 7 and bins of capacity $b = 30$.
- (b) Justify the greedy standard of taking items in first-fit sequence.
- (c) Explain why $\sum_j a_j/b$ is a lower bound on the optimal number of bins needed in any solution, and compare to your result of part (a).
- (d) Explain why there can never be more than one bin \leq half full at any point in the execution of the algorithm, so that the final number \bar{N} of bins it uses must satisfy $(\bar{N} - 1)(b/2) < \sum_j a_j$.
- (e) Combine (c) and (d) to establish that the number of bins used by the first-fit heuristic solution can be no worse than twice optimal.

15-9 Consider solving (approximately) the ILP

$$\begin{aligned} \max \quad & 5x_1 + 7x_2 - 2x_3 \\ \text{s.t.} \quad & x_2 + x_3 \leq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

by a version of discrete improving search Algorithm 15B that employs move set $\mathcal{M} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ and always advances to the feasible neighbor with best objective value.

- ✔ (a) Identify a global optimal solution by inspection.
- ✔ (b) List all points in the neighborhood of feasible solution $\mathbf{x}^{(0)} = (0, 0, 1)$.
- ✔ (c) Compute a local optimal solution by applying Algorithm 15B starting from $\mathbf{x}^{(0)} = (0, 0, 1)$.
- ✔ (d) Repeat part (c), this time starting at $\mathbf{x}^{(0)} = (1, 0, 0)$.
- (e) Compare results in parts (c) and (d), and comment on the effect of starting solutions.
- ✔ (f) Repeat part (c), this time using the single-complement move set that allows any one $x_j = 1$ to be switched to $= 0$, or vice versa.
- (g) Compare results in parts (c) and (f), and comment on the effect of move set.

15-10 Repeat Exercise 15-9 for the ILP

$$\begin{aligned} \min \quad & 2x_1 - 11x_2 + 14x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 1 \\ & x_1, x_2, x_3 = 0 \text{ or } 1 \end{aligned}$$

15-11 Consider solving (approximately) the ILP

$$\begin{aligned} \max \quad & 12x_1 + 7x_2 + 9x_3 + 8x_4 \\ \text{s.t.} \quad & 3x_1 + x_2 + x_3 + x_4 \leq 3 \\ & x_3 + x_4 \leq 1 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

by a version of discrete improving search Algorithm 15B that always advances to the feasible neighbor with best objective value and uses the single-complement neighborhood permitting any one $x_j = 1$ to be switched to $= 0$, or vice versa.

- ✔ (a) Identify a global optimal solution by inspection.

- ✔ (b) Use Algorithm 15B to compute a local optimum starting from $\mathbf{x}^{(0)} = (0, 0, 0, 0)$.
- ✔ (c) Apply the multistart extension of improving search to compute a local optimum by trying starts at $\mathbf{x} = (0, 1, 0, 0)$, and $(0, 0, 0, 1)$.

15-12 Do Exercise 15-11 for the ILP

$$\begin{aligned} \min \quad & 20x_1 + 40x_2 + 20x_3 + 15x_4 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \\ & x_1 + x_4 \geq 1 \\ & x_1, \dots, x_4 = 0 \text{ or } 1 \end{aligned}$$

Start Algorithm 15B at $\mathbf{x} = (1, 1, 1, 1)$, and multistart at $\mathbf{x} = (1, 0, 1, 1)$, and $(1, 1, 1, 0)$.

15-13 Return to the improving search problem of Exercise 15-11.

- (a) Show that $\mathbf{x} = (1, 0, 0, 0)$ is a local optimum.
- (b) Show that if a nonimproving move is allowed at $\mathbf{x} = (1, 0, 0, 0)$, the next iteration will return the search to this same point.

15-14 Do Exercise 15-13 for the model of Exercise 15-12.

✔ **15-15** Return to the improving search problem of Exercise 15-11, starting from $\mathbf{x}^{(0)} = (1, 0, 0, 0)$. Compute an approximate optimum by Tabu search Algorithm 15C, forbidding complementation of a variable for one step after its value changes, and limiting the search to $t_{\max} = 5$ moves.

15-16 Do Exercise 15-15 for the model of Exercise 15-12. Forbid complementation of a variable for two steps after its value changes.

✔ **15-17** Return to the improving search problem of Exercise 15-11, starting from $\mathbf{x}^{(0)} = (0, 0, 0, 1)$. Compute an approximate optimum by Simulated Annealing Algorithm 15D, using a temperature of $q = 20$, limiting the search to $t_{\max} = 4$ moves, and resolving probabilistic decisions with (uniform $[0,1]$) random numbers 0.65, 0.10, 0.40, 0.53, 0.33, 0.98, 0.88, 0.37.

15-18 Do Exercise 15-17 for the model of Exercise 15-12. Use random numbers 0.60, 0.87, 0.77, 0.43, 0.13, 0.19, 0.23, 0.71, 0.78, 0.83, 0.29. Start at $\mathbf{x}^{(0)} = (1, 0, 0, 0)$.

15-19 Return to the (TSP) and the instance of Exercise 15-5(b). Given a feasible tour, the pairwise interchange move set considers all possible swaps of city positions in the current tour. For example, one swap of a current tour 1-2-5-4-3-1 would be 1-2-3-5-3-1.

- Begin with tour 1-2-5-4-3-1 in the instance of Exercise 15-5(b) and apply Improving Search Algorithm 15B with this pairwise interchange to compute an approximate optimum.
- Now apply multistart to the same instance with pairwise interchange neighbor starting from both the beginning solution of (a) and also 1-3-2-5-4-1. What approximate optimum is the result?
- Replicate part (a) applying Tabu Search Algorithm 15C, forbidding swaps involving the first of the two cities in a nonimproving interchange for 2 steps.
- Replicate part (a) applying Simulated Annealing Algorithm 15D, a temperature of $q = 15$, limiting the search to $t_{max} = 4$ moves, and using random numbers 0.05, 0.92, 0.77, 0.40, 0.81, and 0.53 to decide acceptance of nonimproving moves.

15-20 Inform College (IC) is planning a major government issues conference with panels on topics $i = 1, \dots, 30$. Panels will be scheduled in one of $t = 1, \dots, 6$ time blocks, with 5 running simultaneously in each block. To make the conference as convenient as possible, IC has surveyed prospective attendees and computed values $q_{i,i} \triangleq$ the number of attendees who would particularly like to be able to attend both sessions i and i' . Now IC would like to construct a session schedule that minimizes the total number of attendees inconvenienced by pairs of sessions they would like to attend running simultaneously.

- Formulate IC's session scheduling task as a transportation problem analog of Quadratic Assignment model [11.16](#), using decision variables $x_{i,t} \triangleq 1$ if panel i is scheduled in time block t and $= 0$ otherwise.
- Refer to Section 11.4 and Chapter 14 to justify solving the model heuristically using methods of this chapter because tractable exact solution is unlikely.

- Suppose now that the instance is to be solved approximately by Improving Search Algorithm 15B, beginning from a feasible schedule and using a pairwise swap move set to evaluate moves interchanging the current time assignment of a single pair of panels. Calculate the number of such moves at any current assignment $\bar{x}_{i,j}$. Also determine whether all the neighbors produced by such moves will be feasible.
- Repeat part (c) using a single change move set that changes only the assigned time of a single panel.

15-21 Silo State's Industrial Engineering faculty is moving to new offices. Professors $p = 1, \dots, 20$ will be assigned offices among the $r = 1, \dots, 25$ rooms, with unused rooms being left for graduate assistants. Walking distances $d_{r,r'}$ have been computed between all pairs of rooms (r, r') . The department head also has collected values $c_{p,p'} \triangleq$ the number of times per month each pair of professors (p, p') collaborates in their teaching and research. Now, she would like to pick an assignment of professors to rooms that minimizes the total of collaboration traffic $c_{p,p'} \cdot d_{r,r'}$ over all chosen assignments.

- Formulate the head's task as a Quadratic Assignment model [11.16](#), using decision variables $x_{p,r} \triangleq 1$ if professor p is assigned to room r and $= 0$ otherwise, together with the model parameters defined above.
- Refer to Sections 11.4 and Chapter 14 to justify solving the model heuristically using methods of this chapter because tractable exact solution is unlikely.

15-22 Return to the model of Exercise 15-21, and consider solving it approximately with Improving Search Algorithm 15B over each of the following move sets:

- $$M_1 \triangleq \{\text{reassignments of a single professor to any office}\}$$
- $$M_2 \triangleq \{\text{reassignments of a single professor to a vacant office}\}$$
- $$M_3 \triangleq \{\text{swaps of the assignments of two professors}\}$$

The search will begin with each professor randomly assigned to a different office.

- Show that all these move sets yield a polynomial-size neighborhood in terms of numbers of professors and rooms.
- Compare the 3 move sets with regard to whether every neighbor of any current feasible solution must also be feasible.
- For those move sets that can produce infeasible neighbors, show how the objective function of your model can be modified with “big M ” penalties to make such infeasible solutions unattractive.

✓ **15-23** Return to the improving search problem of Exercise 15-11.

- Show that the solutions $\mathbf{x}^{(1)} = (0, 0, 1, 0)$ and $\mathbf{x}^{(2)} = (0, 0, 0, 1)$ are eligible to belong to a genetic algorithm population for the problem.
- Construct all possible crossover results (all cut points) for the $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ of part (a).
- Determine whether all your resulting solutions in part (b) are feasible, and if not, explain what difficulty this presents for effective application of genetic algorithm search.

15-24 Do Exercise 15-23 on the model of Exercise 15-12 using $\mathbf{x}^{(1)} = (0, 1, 1, 1)$ and $\mathbf{x}^{(2)} = (1, 0, 1, 1)$.

✓ **15-25** Return again to the model of Exercise 15-11, and consider employing genetic Algorithm 15E with initial population $\{(0, 0, 1, 0), (0, 0, 0, 1), (0, 1, 1, 0), (1, 0, 0, 0)\}$, $p_e = p_i = 1$, and $p_c = 2$. Construct and evaluate each member of the next generation population, with crossover after component 2 of the best and worst current solutions. Use a large negative M as the objective value of any infeasible solutions produced by crossover.

15-26 Do Exercise 15-25 on the model of Exercise 15-12 with initial population $\{(0, 1, 1, 1), (1, 0, 1, 1), (0, 1, 0, 1), (1, 0, 0, 0)\}$.

15-27 Return to the model of Exercise 15-20 and consider applying Genetic Algorithm 15E.

- First consider encoding solutions by taking sessions in i order and recording the

time block t to which each is assigned. Sketch what would happen if two solutions like this were combined in crossover. What kinds of infeasibility could result? Explain. Then discuss how it might be managed by penalizing infeasibility with suitable “big M ” terms in the objective function.

- Now consider encoding solutions by first listing the 5 panel numbers for the first time slot, then the 5 for the second, and so on. Sketch what would happen if two solutions were combined in crossover with cut points limited to the 5 boundaries between session lists for particular time slots, that is, after entry 5, 10, 15, 20, and 25. What kinds of infeasibility could result? Explain. Then discuss how it might be managed by penalizing infeasibility with suitable “big M ” terms in the objective function.

- Finally, consider the method of Random Keys defined in [15.13]. Specifically, the solution would be encoded indirectly with 30 random numbers corresponding to the 30 panels. To recover the implied solution and evaluate the objective function, the 5 panels with lowest random numbers would be assigned to time slot 1, the next 5 to time slot 2, and so on. Explain why infeasibility could not result from crossovers between 2 such solutions, and how that could make Algorithm 15E more effective.

15-28 Return to the model of Exercise 15-21 and consider applying Genetic Algorithm 15E.

- First consider encoding solutions by taking professors in p order (adding $p = 21, \dots, 25$ for rooms assigned to graduate assistants), then recording the room r to which each is assigned. Sketch what would happen if two solutions such as this were combined in crossover. What kinds of infeasibility could result? Explain. Then discuss how it might be managed by penalizing infeasibility with suitable “big M ” terms in the objective function.

- Now consider encoding solutions by taking rooms in r order, recording the professor (or graduate assistants) p assigned

to the room. Sketch what would happen if two solutions were combined in crossover with cut points limited to the 5 boundaries between session lists for particular time slots, that is, after entry 5, 10, 15, 20, and 25. What kinds of infeasibility could result? Explain. Then discuss how it might be managed by penalizing infeasibility with suitable “big M” terms in the objective function.

- (c) Finally, consider the method of Random Keys defined in [15.13]. Specifically, the

solution would be encoded indirectly with 25 random numbers corresponding to rooms as in part (b). To recover the implied solution and evaluate the objective function, the room with the lowest random number would be assigned to professor 1, the second to professor 2, and so on with the last 5 assigned to graduate students. Explain why infeasibility could not result from crossovers between 2 such solutions, and how that could make Algorithm 15E more effective.

REFERENCES

- Aarts, Emile and Jan Korst (1989), *Simulated Annealing and Boltzmann Machines*, John Wiley, Chichester, England.
- Glover, Fred and Manuel Laguna (1997), *Tabu Search*, Kluwer, Boston, Massachusetts.
- Goldberg, David E. (1989), *Genetic Algorithms in Search and Machine Learning*, Addison-Wesley, Reading, Massachusetts.
- Parker, R. Gary and Ronald L. Rardin (1988), *Discrete Optimization*, Academic Press, San Diego, California.
- Talbi, El-Ghazali (2009), *Metaheuristics from Design to Implementation*, John Wiley, Hoboken, New Jersey.
- Wolsey, Laurence (1998), *Integer Programming*, John Wiley, New York, New York.

This page intentionally left blank

Unconstrained Nonlinear Programming

A major theme of this book has been the power and elegance of linear programming models, which are models with continuous decision variables, linear constraints, and a linear objective function. **Nonlinear programming (NLP)** encompasses all the rest of single-objective optimization over continuous decision variables.

Being defined only by what it is not—linear—leaves nonlinear programming with a host of quite different forms and algorithms. Some models have constraints. Others have only an objective function. Calculus yields readily exploitable derivatives in many models. Derivatives do not even exist in others. In some cases, both objective function and constraints are nonlinear. In others it is only the objective function. Even single-variable optimization is a nontrivial topic when the objective is nonlinear.

This chapter begins our treatment of nonlinear programming with the **unconstrained** case where no constraints apply. Chapter 17 follows with the more complicated models having constraints that cannot be ignored. There are important unconstrained applications, but most real models have at least a few constraints. We treat unconstrained cases first because many of the underlying notions of nonlinear programming are easier to understand without the encumbrance of constraints, and because most methods for constrained optimization use unconstrained algorithms as building blocks. Familiarity with definitions of Section 2.4 and improving search concepts of Chapter 3 is assumed throughout.

16.1 UNCONSTRAINED NONLINEAR PROGRAMMING MODELS

One of the major differences between linear and nonlinear programs is that **unconstrained** NLPs—ones with no constraints—can still be meaningful.

Principle 16.1 | Unconstrained optimization over a linear objective function is always unbounded (except in the trivial case where the objective is constant), but unconstrained nonlinear programs can have finite optimal solutions.

We begin our discussion of unconstrained NLPs with some typical applications.

APPLICATION 16.1: USPS SINGLE VARIABLE

Even models with a single decision variable can be challenging when the objective function is nonlinear. For a real single-variable nonlinear program we turn to the U.S. Postal Service (USPS).¹ Service “territories” for the USPS typically consist of a city and its suburbs. Mail delivery is provided by a number of postal carriers driving, or sometimes walking, specified “delivery regions.” Carriers are based at “delivery units” distributed throughout the territory, beginning and ending their routes there each workday. Often, a delivery unit is also a local post office that sells stamps, and so on, to the general public.

Determining the most efficient number of delivery units for a territory involves a trade-off between the fixed overhead costs of operating delivery units and the travel savings from carriers being based nearer their delivery regions. More delivery units increase overhead, but they reduce the number of carriers required by saving travel as the units are dispersed closer to customers.

Increasing automation of mail handling has significantly changed the relative economics of such decisions. To adjust, the USPS has developed and applied a rough decision model computing the approximate number of delivery units appropriate for any given territory. Input parameters are

- $a \triangleq$ land area of the territory
- $m \triangleq$ number of customers in the territory
- $t \triangleq$ average time for a carrier to service any customer site
- $d \triangleq$ length of the carrier work day
- $c \triangleq$ annual cost per carrier
- $u \triangleq$ annual overhead cost of operating a delivery unit

We want to determine the decision variable

$$x \triangleq \text{number of delivery units}$$

To develop a model, we employ approximations derived from an assumption that customers are spread evenly over the territory. Under that assumption it can be shown that

$$\begin{aligned} \text{average travel time per carrier to/from regions} &\approx k_1 \sqrt{\frac{a}{x}} \\ \text{travel time between route stops for all routes} &\approx k_2 \sqrt{am} \end{aligned}$$

where k_1 and k_2 are constants of proportionality. Then the total number of routes is

$$\frac{(\text{total time at stops}) + (\text{total time between stops})}{\text{effective work time per carrier}} = \frac{tm + k_2 \sqrt{am}}{d - k_1 \sqrt{a/x}}$$

and total cost is

$$\text{overhead} + \text{operations} = ux + c \left(\frac{tm + k_2 \sqrt{am}}{d - k_1 \sqrt{a/x}} \right) \quad (16.1)$$

¹Based on D. B. Rosenfield, I. Engelstein, and D. Feigenbaum (1992), “An Application of Sizing Service Territories,” *European Journal of Operational Research*, 63, 164–177.

USPS Single-Variable Application Model

To have a specific example with which to deal, pick

$$\begin{aligned} a &= 400, & m &= 200,000, & d &= 8, & t &= 0.05 \\ c &= 0.10, & u &= 0.75, & k_1 &= 0.2, & k_2 &= 0.1 \end{aligned}$$

in expression (16.1). Then our single-variable USPS nonlinear program is

$$\begin{aligned} \min f(x) &\triangleq (0.75)x + (0.10) \left[\frac{(0.05)(200,000) + (0.1)\sqrt{400(200,000)}}{(8) - (0.2)\sqrt{400/x}} \right] \quad (16.2) \\ &\approx 0.75x + \frac{1089.4}{8 - 0.2\sqrt{400/x}} \end{aligned}$$

Figure 16.1 shows that a unique optimal solution occurs at $x^* \approx 15.3$.

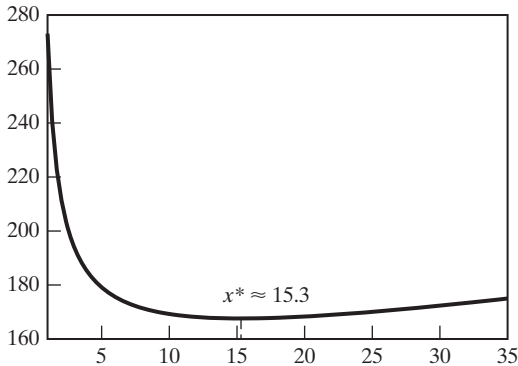


FIGURE 16.1 USPS Application Objective Function

Neglecting Constraints to Use Unconstrained Methods

Strictly speaking, model (16.2) is incomplete. A meaningful number of delivery units, x , should satisfy the constraint

$$x \geq 0$$

and perhaps also

$$x \text{ integer}$$

Still, we know (Chapter 6) that adding a constraint can change an optimal solution only if it is violated. With $x^* = 15.2 > 0$, the neglected nonnegativity constraint would have no impact even if we modeled it explicitly. Integrality is violated at x^* , but the rough planning nature of the model suggests that we would be quite justified in rounding to $x = 15$.

Many, perhaps most, problems modeled as unconstrained nonlinear programs actually have a few constraints that are neglected in this way.

Principle 16.2 | The relative tractability of unconstrained models often justifies neglecting simple constraints that should apply until an optimum has been computed, checking only afterward that the solution is feasible.

Of course, if the unconstrained optimum violates an important constraint, we must resort to constrained methods.

Curve Fitting and Regression Problems

Perhaps the most common of unconstrained NLPs involve **curve fitting** or **regression**. We seek to choose coefficients for a functional form to make it fit closely some observed data.

Principle 16.3 | Decision variables in a regression problem are the coefficients of the fitted functional form, and the objective function measures the accuracy of the fit.

APPLICATION 16.2: CUSTOM COMPUTER CURVE FITTING

For a simple application of curve fitting, we consider the problem of (fictitious) Custom Computer Company, which builds specialized computer workstations for engineers. Although there may be many commonalities, stations produced for each order are specially modified to meet customer specifications.

Table 16.1 shows the number of units and unit cost (in thousands of dollars) of $m = 12$ recent orders. Figure 16.2 plots this experience. Obviously, unit cost declines dramatically with the size of an order. Custom wants to fit an estimating function like the one depicted in Figure 16.2 to facilitate preparation of bids on future work.

TABLE 16.1 Cost Data for Custom Computer Application

i	Number, p_i	Cost, q_i	i	Number, p_i	Cost, q_i	i	Number, p_i	Cost, q_i
1	19	7.9	5	5	19.5	9	14	9.2
2	2	25.0	6	6	13.0	10	17	6.3
3	9	13.1	7	3	17.8	11	1	42.0
4	4	17.4	8	11	8.0	12	20	6.6

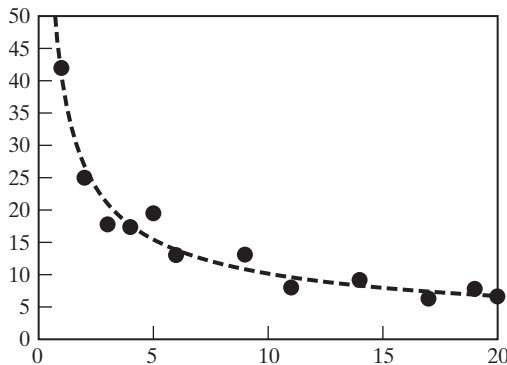


FIGURE 16.2 Fitted Curve for Custom Computer Application

Linear versus Nonlinear Regression

Our first task in dealing with Custom Computer's problem is choosing a regression form to fit. That is, we want to choose a function $r(p)$ with unknown coefficients to approximate

$$r(p_i) \approx q_i \quad \text{for all } i = 1, \dots, m$$

Nonlinear optimization will choose the best coefficient values.

The fact that functional coefficients constitute the decision variables (principle 16.3) in curve fitting leads to considerable confusion. For example, regression analysts often denote a form something like

$$r(x) \triangleq a + bx \tag{16.3}$$

with x being data and $\{a, b\}$ the coefficients to determine. This choice of notation is exactly the reverse of the one familiar in optimization.

We will follow the mathematical programming tradition of reserving x for decision variables. Thus form (16.3) might be expressed as

$$r(p) \triangleq x_1 + x_2 p \tag{16.4}$$

with p the data and $\{x_1, x_2\}$ the undetermined coefficients.

A similar confusion arises in distinguishing **linear** versus **nonlinear regression**.

Definition 16.4 A regression problem is termed linear if the functional form being fitted is linear in the unknown coefficients (decision variables) and nonlinear otherwise.

For example, the choice of

$$r(p) \triangleq x_1 + \frac{x_2}{p} \tag{16.5}$$

in our Custom Computer case would be a linear regression. The corresponding curve in Figure 16.2 would not be a straight line, but expression (16.5) is linear in the unknown coefficients x_1 and x_2 .

The distinction between linear and nonlinear regression is important because there is often a closed-form solution for fitting linear forms, but nonlinear ones usually require search. Computations throughout this chapter will illustrate for nonlinear regression form

$$r(p) \triangleq x_1 p^{x_2} \tag{16.6}$$

on our Custom Computer application. The curve depicted in Figure 16.2 is

$$r(p) \triangleq 40.69p^{-0.6024}$$

which provides an optimal fit.

EXAMPLE 16.1: DISTINGUISHING LINEAR AND NONLINEAR REGRESSION

Taking variables x_j to be the unknown coefficients and all other symbols as given data, determine whether fitting each of the following forms is linear or nonlinear regression.

- (a) $r(p) \triangleq x_1 + x_2 \sin(p)$
- (b) $r(p) \triangleq x_1 + \sin(x_2)p$
- (c) $r(p_1, p_2) \triangleq x_1 p_1^2 + x_2 e^{p_2}$

Solution: We apply definition 16.4.

- (a) This regression form is linear because it is linear in the decision variables x_1 and x_2 for given data p .
- (b) This regression form is nonlinear. Decision variable x_2 appears in the nonlinear expression $\sin(x_2)$.
- (c) This regression form is linear. It fits a nonlinear function of two inputs p_1 and p_2 , but decision variables x_1 and x_2 occur linearly.

Regression Objective Functions

To complete formulation of curve fitting as a nonlinear optimization, we require an objective function measuring fit. The error or **residual** associated with any data point is the difference between the fitted function and the actual value observed. For example, in our Custom Computer case, residuals under functional form (16.6) are

$$q_i - r(p_i) = q_i - x_1(p_i)^{x_2} \quad \text{for all } i = 1, \dots, m$$

Regression objective functions minimize some nondecreasing function of the magnitudes of residuals. Many possibilities have been employed, but the most common is the sum of residual squares or **least squares**. This objective possesses a number of desirable statistical properties, and it also has the intuitive appeal that small deviations cost little but large ones are heavily penalized.

Custom Computer Curve Fitting Application Model

For our Custom Computer application, the least squares objective produces unconstrained NLP model

$$\min f(x_1, x_2) \triangleq \sum_{i=1}^m [q_i - x_1(p_i)^{x_2}]^2 \tag{16.7}$$

Figure 16.3 displays the objective graphically, with global minimum at

$$x_1^* \approx 40.69, \quad x_2^* \approx -0.6024$$

yielding the best fit.

EXAMPLE 16.2: FORMULATING NONLINEAR REGRESSION MODELS

Three observations from a function believed to have the form

$$z = \alpha^u \beta^v$$

are $(u_1, v_1, z_1) = (1, 8, 3)$, $(u_2, v_2, z_2) = (4, 15, 2)$, and $(u_3, v_3, z_3) = (2, 29, 71)$. Formulate an unconstrained nonlinear program to choose the α and β yielding a least squares fit.

Solution: Residuals are $(z_i - \alpha^{u_i}\beta^{v_i})$. Thus a least squares fit will be obtained at α and β solving

$$\min f(\alpha, \beta) \triangleq (3 - \alpha^1\beta^8)^2 + (2 - \alpha^4\beta^{15})^2 + (71 - \alpha^2\beta^{29})^2$$

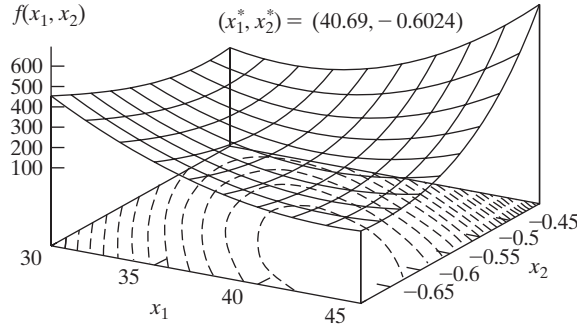


FIGURE 16.3 Objective Function in Custom Computer Application

Maximum Likelihood Estimation Problems

Another common application of unconstrained NLP arises in fitting continuous probability distributions to observed data. A **probability density function**, $d(p)$, characterizes any such distribution by showing how the probability is spread over different values of a random variable, P . For example, the $d(p)$ depicted in Figure 16.4 indicates relatively higher probability of values of P near 0.7 than near 0.2.

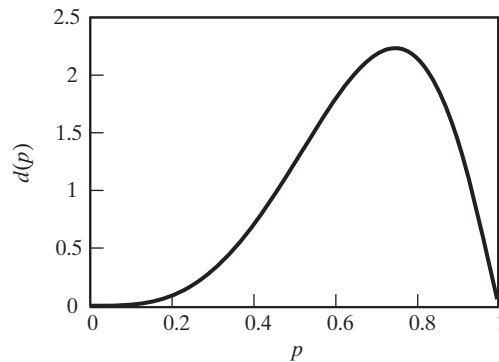


FIGURE 16.4 Density Function for PERT Maximum Likelihood Application

When say m independent random variables P_1, P_2, \dots, P_m have the same probability density $d(p)$, the **joint probability density function** or **likelihood** is

$$d(p_1, p_2, \dots, p_m) = d(p_1)d(p_2) \dots d(p_m) \quad (16.8)$$

That is, the density for any particular independent combination of values is the product of the densities for the values separately.

Estimation involves choosing values for unknown coefficients of a density functional form $d(p)$. The best estimates by many criteria are **maximum likelihood**, ones that maximize the joint density of a known sample of random variable values.

Principle 16.5 Decision variables in maximum likelihood estimation are the coefficients of the fitted probability density, and the objective function is the corresponding joint density or likelihood evaluated at the observed data.

APPLICATION 16.3: PERT MAXIMUM LIKELIHOOD

Maximum likelihood estimates for parameters of many standard probability densities can be obtained in closed form. However, some require a numerical optimization to maximize the likelihood function.

A specific example occurs in fitting the **beta distribution** often used in **project evaluation and review technique (PERT)** project management. PERT is an extension of the **CPM** project scheduling method introduced in Section 9.7. As with CPM, a project is divided into a series of work activities with specified time durations. The new element with PERT is that durations are taken as random variables (i.e., they are assumed to be known only in probability distribution at the time that planning takes place).

Beta random variables assign probability density over the interval $0 \leq p \leq 1$, so that in PERT they reflect the fraction p that some activity's duration forms of an allowed maximum. The beta probability density function is

$$d(p) \triangleq \frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (p)^{x_1-1}(1 - p)^{x_2-1} \tag{16.9}$$

where $x_1 > 0$ and $x_2 > 0$ are parameters controlling its shape.² For instance, $x_1 = 4.50$, and $x_2 = 2.20$ yields density shown in Figure 16.4. In expression (16.9), $\Gamma(x)$ is the standard Γ -function equal to the area under the curve

$$\gamma(x) \triangleq (h)^{x-1}(e)^{-h}$$

over $0 \leq h \leq +\infty$. $\Gamma(x)$ has no closed form.

Table 16.2 shows the data we will assume for $m = 10$ previous times a project activity was undertaken. Values p_i represent actual duration as a fraction of the maximum ever expected.

TABLE 16.2 Realized Data for PERT Maximum Likelihood Application

	Value		Value		Value		Value		Value
p_1	0.65	p_3	0.52	p_5	0.74	p_7	0.79	p_9	0.92
p_2	0.57	p_4	0.72	p_6	0.30	p_8	0.89	p_{10}	0.42

²Beta parameters are more commonly called α and β , but we will employ x_1 and x_2 to be consistent with the convention that x_j refers to a decision variable.

PERT Maximum Likelihood Application Model

The beta probability density for observation $p_1 = 0.65$ is

$$d(p_1) = d(0.65) = \frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (0.65)^{x_1-1} (1 - 0.65)^{x_2-1}$$

and for the first two observations together it is [applying expression (16.8)]

$$d(p_1)d(p_2) = \left[\frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (0.65)^{x_1-1} (1 - 0.65)^{x_2-1} \right] \\ \cdot \left[\frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (0.57)^{x_1-1} (1 - 0.57)^{x_2-1} \right]$$

Continuing in this way our nonlinear program to maximize the likelihood of the full m -value sample over coefficients x_1 and x_2 is

$$\max f(x_1, x_2) \triangleq \prod_{i=1}^m \left[\frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (p_i)^{x_1-1} (1 - p_i)^{x_2-1} \right] \quad (16.10)$$

Figure 16.5 plots this objective for various x_1 and x_2 . The unique global maximum is at $x_1^* \approx 4.50$ and $x_2^* \approx 2.20$ of the density in Figure 16.4.

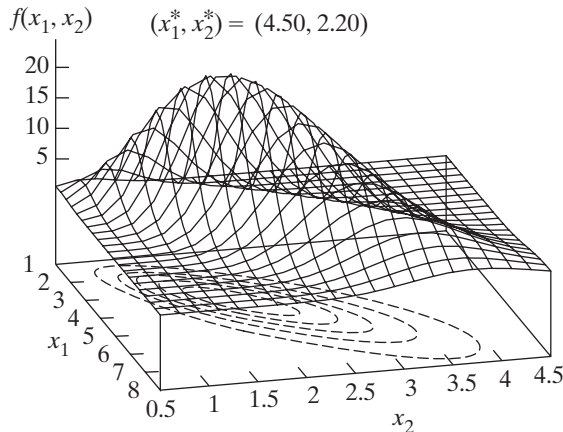


FIGURE 16.5 PERT Maximum Likelihood Application Objective Function

EXAMPLE 16.3: FORMULATING MAXIMUM LIKELIHOOD MODELS

Exponential probability distributions have density function

$$d(p) \triangleq \alpha e^{-\alpha p}$$

Formulate an unconstrained nonlinear program to determine the maximum likelihood value of parameter α consistent with realizations $p_1 = 4$, $p_2 = 9$, and $p_3 = 8$.

Solution: The likelihood is the product (expression (16.8)) of the densities for the three realizations. Thus the model required is

$$\max f(\alpha) \triangleq (\alpha e^{-4\alpha})(\alpha e^{-9\alpha})(\alpha e^{-8\alpha})$$

Smooth versus Nonsmooth Functions and Derivatives

It is useful to classify nonlinear programs according to whether their objective functions are smooth or nonsmooth.

Definition 16.6 A function $f(\mathbf{x})$ is said to be **smooth** if it is continuous and differentiable at all relevant \mathbf{x} . Otherwise, it is **nonsmooth**.

Figure 16.6 illustrates this for some functions of a single x . (Refer, if needed, to Primer 2 in Section 3.3 for a quick review of differential calculus.)

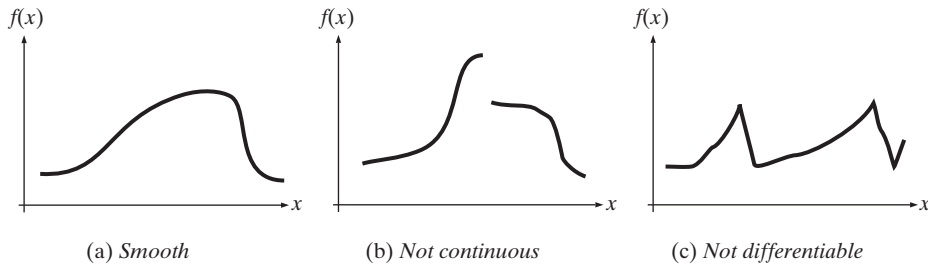


FIGURE 16.6 Examples of Smooth and Nonsmooth Functions

The smooth/nonsmooth distinction is useful because the more erratic nature of nonsmooth functions usually implies a more difficult search.

Principle 16.7 Nonlinear programs over smooth functions are generally more tractable than those over nonsmooth ones.

EXAMPLE 16.4: RECOGNIZING SMOOTH FUNCTIONS

Determine whether each of the following single variable functions is smooth or nonsmooth over the specified domain.

- (a) $f(x) \triangleq x^3$ for $x \in (-\infty, +\infty)$
- (b) $f(x) \triangleq |x - 1|$ for $x \in (-\infty, +\infty)$
- (c) $f(x) \triangleq \frac{1}{x}$ for $x > 0$

Solution: We apply definition 16.6.

- (a) This function is differentiable at every point in the domain, so the function is smooth.
- (b) This function is not differentiable at point $x = 1$, which is within the range. Thus the function is nonsmooth.
- (c) This function is discontinuous at $x = 0$. However, that point is not in the domain specified. Thus the function is smooth for $x > 0$.

Usable Derivatives

All three objective functions for the examples of this section (Figures 16.1, 16.3, and 16.5) are smooth. Still, the existence of (partial) derivatives at every relevant \mathbf{x} does not necessarily imply that derivatives are readily available to aid search algorithms.

For the USPS application

$$f(x) \triangleq 0.75x + \frac{1089.4}{8 - 0.2\sqrt{400/x}}$$

and

$$\frac{df}{dx} = 0.75 - \frac{1089.4}{(8 - 0.2\sqrt{400/x})^2} \left(\frac{0.1}{\sqrt{400/x}} \right) \left(\frac{400}{x^2} \right) \quad (16.11)$$

For the Custom Computer case

$$f(x_1, x_2) \triangleq \sum_{i=1}^m [q_i - x_1(p_i)^{x_2}]^2$$

with

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= -2 \sum_{i=1}^m [q_i - x_1(p_i)^{x_2}](p_i)^{x_2} \\ \frac{\partial f}{\partial x_2} &= -2 \sum_{i=1}^m [q_i - x_1(p_i)^{x_2}][x_1(p_i)^{x_2}] \ln(p_i) \end{aligned} \quad (16.12)$$

Neither (16.11) nor (16.12) is a particularly simple expression, but both show derivatives that can be computed with reasonable effort to speed a search.

In elementary calculus, unconstrained optima are computed by solving the system of equations resulting from setting derivatives = 0. With complex derivative expressions such as (16.11) and (16.12), solving such a system of equations is often as difficult as solving the underlying nonlinear program. Nevertheless, practically available derivatives can be a significant aid to a numerical search for optimal variable values.

Principle 16.8 | Nonlinear programs over smooth functions with convenient-to-compute derivatives are usually more tractable than those without, because derivatives can be exploited to produce a much more efficient search.

Contrast with the PERT maximum likelihood objective

$$f(x_1, x_2) \triangleq \prod_{i=1}^m \left[\frac{\Gamma(x_1 + x_2)}{\Gamma(x_1)\Gamma(x_2)} (p_i)^{x_1-1} (1 - p_i)^{x_2-1} \right]$$

The Γ -function itself has no closed form, so derivatives are certainly not readily available, even though they do exist in theory. To compute an optimum for this case we require a search method that does not depend on derivatives.

EXAMPLE 16.5: ASSESSING PRACTICALITY OF DERIVATIVES

Return to the least squares objective function of Example 16.2:

$$\min f(\alpha, \beta) \triangleq (3 - \alpha^1\beta^8)^2 + (2 - \alpha^4\beta^{15})^2 + (71 - \alpha^2\beta^{29})^2$$

- (a) Express partial derivatives with respect to α and β .
- (b) Discuss the usefulness of those partial derivatives in computing an optimal α and β .

Solution:

(a) Partial derivatives are

$$\begin{aligned} \frac{\partial f}{\partial \alpha} &= 2(3 - \alpha^1\beta^8)(-\beta^8) + 2(2 - \alpha^4\beta^{15})(-4\alpha^3\beta^{15}) \\ &\quad + 2(71 - \alpha^2\beta^{29})(-2\alpha\beta^{29}) \\ \frac{\partial f}{\partial \beta} &= 2(3 - \alpha^1\beta^8)(-8\alpha^1\beta^7) + 2(2 - \alpha^4\beta^{15})(-5\alpha^4\beta^{14}) \\ &\quad + 2(71 - \alpha^2\beta^{29})(-29\alpha^2\beta^{28}) \end{aligned}$$

(b) Although derivative expressions of part (a) are somewhat complicated, they can be evaluated efficiently. Simply setting them = 0 is impractical because it would leave a pair of difficult nonlinear equations to solve. However, the derivatives can be employed to speed an improving search (principle [16.8](#)).

16.2 ONE-DIMENSIONAL SEARCH

The easiest case on unconstrained nonlinear programming is single-variable or **1-dimensional search**. One-dimensional NLPs occur both directly, as in our USPS application of Section 16.1, and as **line search** subroutines choosing step sizes to apply to move directions of more general algorithms.

Unimodal Objective Functions

Figure 16.7 illustrates for

$$f(x) \triangleq (x - 4)(x - 6)^3(x - 1)^2$$

how 1-dimensional optimization can be quite challenging.

- Points $x^{(1)}$ and $x^{(3)}$ are both **local minima** because small changes from either x do not decrease the objective function (definition [3.5](#)). Only $x^{(3)}$ is an overall, **global minimum** (definition [3.7](#)) for the displayed interval.
- Point $x^{(2)}$ is a local maximum because small changes from $x^{(2)}$ do not increase the objective function. Being the only local maximum, it is also a global maximum for the displayed interval.
- Point $x^{(4)}$ is neither a maximum nor a minimum, despite the fact that the slope of $f(x)$ (derivative df/dx) at the point = 0.

Fortunately, most of the 1-dimensional searches we encounter in application are somewhat better behaved.

One example is unimodal objectives.

Definition 16.9 An objective function $f(x)$ is **unimodal** if the straight line direction $\Delta \mathbf{x}$ to any \mathbf{x} with better objective value is an improving direction. Over unimodal objective functions, every unconstrained local optimum must be a global optimum.

The “one-hump” or “single-mode” character that gives unimodal functions their name is particularly easy to grasp in 1-dimensional cases. For example, the minimizing USPS objective of Figure 16.1 is unimodal because the objective is decreasing (improving) at every x to the left of $x^* \approx 15.2$ and increasing (degrading) at every x to the right. Since a tiny move toward x^* always helps, the local minimum there must be global.

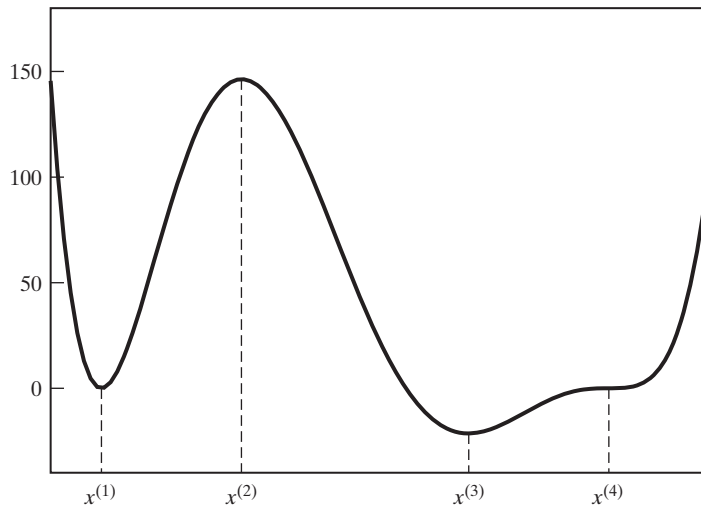


FIGURE 16.7 Single-Variable Nonlinear Function

Notice that an objective being unimodal depends on whether we are maximizing or minimizing. The USPS objective of Figure 16.1 is not unimodal for a maximize problem because, for example, a move toward $x^{(2)} = 30$ does not immediately improve the objective function at $x^{(1)} = 10$, even though $f(30) > f(10)$.

Golden Section Search

Although derivatives can sometimes be of assistance, many 1-dimensional optimizations employ simpler methods not requiring derivatives. Among the most clever is **golden section search**, which deals with an unimodal objective by rapidly narrowing an interval guaranteed to contain an optimum.

Figure 16.8 illustrates the idea for a minimize problem. We iteratively consider the functional value at four carefully spaced points. Leftmost $x^{(lo)}$ is always a lower bound on the optimal x^* , and $x^{(hi)}$ is an upper bound, so that an optimum is certain to lie within the interval $[x^{(lo)}, x^{(hi)}]$. Points $x^{(1)}$ and $x^{(2)}$ fall in between.

Each iteration begins by determining whether the objective is better at $x^{(1)}$ or $x^{(2)}$. If $x^{(1)}$ proves superior [part (a)], we may conclude the optimum lies within the

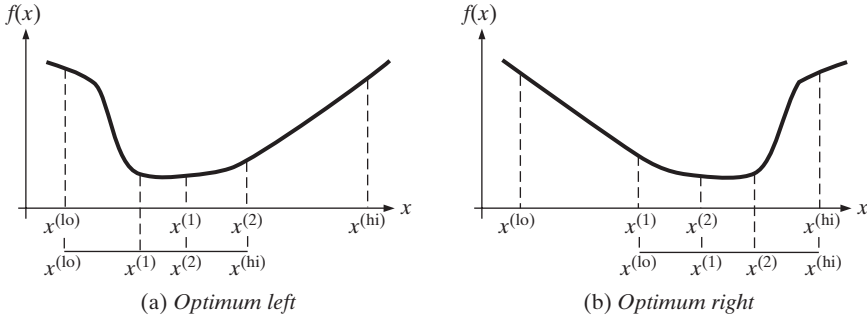


FIGURE 16.8 Interval Reduction in a Minimizing Golden Section Search

smaller interval $[x^{(lo)}, x^{(2)}]$. If $x^{(2)}$ has a better objective value [part (b)], we restrict further attention to the interval $[x^{(1)}, x^{(hi)}]$.

Each narrowing of the interval certain to contain an optimum leaves us with 2 endpoints and 1 interior point at which we already know the objective value. The efficiency of golden section search comes from how we choose 1 new interior point to evaluate. Any new point would allow the search to continue, but we want consistent progress regardless of whether the next interval is $[x^{(lo)}, x^{(2)}]$ or $[x^{(1)}, x^{(hi)}]$.

Golden section search proceeds by keeping both these possible intervals equal in length.

Principle 16.10 The two middle points of golden section search are spaced

$$x^{(1)} = x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)})$$

$$x^{(2)} = x^{(lo)} + \alpha(x^{(hi)} - x^{(lo)})$$

where $\alpha \approx 0.618$ is a fraction known as the **golden ratio**.

Whichever of $[x^{(lo)}, x^{(2)}]$ or $[x^{(1)}, x^{(hi)}]$ provides the next interval, its length will be α times the current.

The golden ratio value of

$$\alpha = \frac{-1 + \sqrt{5}}{2} \approx 0.618 \tag{16.13}$$

arises from the need to maintain the spacing of principle 16.10 as the algorithm proceeds. For example, suppose that the chosen next interval is $[x^{(lo)}, x^{(2)}]$. As indicated in Figure 16.8(a), we want current $x^{(1)}$ to play the role of $x^{(2)}$ in the next interval. Applying formulas 16.10 yields

$$\text{current } x^{(1)} = x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)})$$

and

$$\begin{aligned} \text{next } x^{(2)} &= x^{(lo)} + \alpha(x^{(2)} - x^{(lo)}) \\ &= x^{(lo)} + \alpha(x^{(lo)} + \alpha(x^{(hi)} - x^{(lo)}) - x^{(lo)}) \end{aligned}$$

Equating and regrouping produces

$$0 = \alpha^2(x^{(hi)} - x^{(lo)}) + \alpha(x^{(hi)} - x^{(lo)}) - (x^{(hi)} - x^{(lo)})$$

which further simplifies with $x^{(hi)} \neq x^{(lo)}$ to

$$0 = \alpha^2 + \alpha - 1$$

The unique positive root of this quadratic equation is $\alpha =$ the golden ratio of expression (16.13).

Golden Section Solution of USPS Application

Algorithm 16A formalizes these ideas of golden section search. Table 16.3 details its application to our minimizing unimodal USPS model (16.2).

ALGORITHM 16A: GOLDEN SECTION SEARCH

Step 0: Initialization. Choose lower bound $x^{(lo)}$ and upper bound $x^{(hi)}$ on an optimal solution x^* along with stopping tolerance $\varepsilon > 0$, compute

$$\begin{aligned} x^{(1)} &\leftarrow x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)}) \\ x^{(2)} &\leftarrow x^{(lo)} + \alpha(x^{(hi)} - x^{(lo)}) \end{aligned}$$

for golden ratio α of (16.13), evaluate objective function $f(x)$ at all four points, and initialize iteration counter $t \leftarrow 0$.

Step 1: Stopping. If $(x^{(hi)} - x^{(lo)}) \leq \varepsilon$, stop and report as an approximate optimal solution

$$x^* \leftarrow \frac{1}{2} (x^{(lo)} + x^{(hi)})$$

the midpoint of the remaining interval. Otherwise, proceed to Step 2 if $f(x^{(1)})$ is superior to $f(x^{(2)})$ (less for a minimize model, greater for a maximize), and to Step 3 if it is not.

Step 2: Left. Narrow the search to the left part of the interval by updating

$$\begin{aligned} x^{(hi)} &\leftarrow x^{(2)} \\ x^{(2)} &\leftarrow x^{(1)} \\ x^{(1)} &\leftarrow x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)}) \end{aligned}$$

and evaluate the objective at new point $x^{(1)}$. Then advance $t \leftarrow t + 1$, and return to Step 1.

Step 3: Right. Narrow the search to the right part of the interval by updating

$$\begin{aligned} x^{(lo)} &\leftarrow x^{(1)} \\ x^{(1)} &\leftarrow x^{(2)} \\ x^{(2)} &\leftarrow x^{(lo)} + \alpha(x^{(hi)} - x^{(lo)}) \end{aligned}$$

and evaluate the objective at new point $x^{(2)}$. Then advance $t \leftarrow t + 1$, and return to Step 1.

Computation in Table 16.3 starts arbitrarily with interval

$$[x^{(lo)}, x^{(hi)}] = [8, 32]$$

TABLE 16.3 Golden Section Search of USPS Application

t	$x^{(lo)}$	$x^{(1)}$	$x^{(2)}$	$x^{(hi)}$	$f(x^{(lo)})$	$f(x^{(1)})$	$f(x^{(2)})$	$f(x^{(hi)})$	$x^{(hi)} - x^{(lo)}$
0	8.00	17.17	22.83	32.00	171.42	167.74	169.22	173.38	24.00
1	8.00	13.67	17.17	22.83	171.42	167.73	167.74	169.22	14.83
2	8.00	11.50	13.67	17.17	171.42	168.36	167.73	167.74	9.17
3	11.50	13.67	15.00	17.17	168.36	167.73	167.62	167.74	5.67
4	13.67	15.00	15.83	17.17	167.73	167.62	167.63	167.74	3.50
5	13.67	14.49	15.00	15.83	167.73	167.64	167.62	167.63	2.16
6	14.49	15.00	15.32	15.83	167.64	167.62	167.61	167.63	1.34
7	15.00	15.32	15.51	15.83	167.62	167.61	167.62	167.63	0.83
8	15.00	15.20	15.32	15.51	167.62	167.61	167.61	167.62	0.51
9	15.20	15.32	15.39	15.51	167.61	167.61	167.61	167.62	0.32

It is only necessary that the interval contain the optimum. Intermediate points $x^{(1)}$ and $x^{(2)}$ are then computed by principle 16.10 as

$$\begin{aligned}
 x^{(1)} &\leftarrow x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)}) \\
 &= 32 - 0.618(32 - 8) \\
 &\approx 17.17 \\
 x^{(2)} &\leftarrow x^{(lo)} + \alpha(x^{(hi)} - x^{(lo)}) \\
 &= 8 + 0.618(32 - 8) \\
 &\approx 22.83
 \end{aligned}$$

At iteration $t = 0$, objective value

$$f(x^{(1)}) = 167.74 < f(x^{(2)}) = 169.22$$

implying that the optimum lies in the left part of the current interval. Following Algorithm 16A, we update

$$\begin{aligned}
 x^{(hi)} &\leftarrow x^{(2)} = 22.83 \\
 x^{(2)} &\leftarrow x^{(1)} = 17.17
 \end{aligned}$$

and compute new

$$\begin{aligned}
 x^{(1)} &\leftarrow x^{(hi)} - \alpha(x^{(hi)} - x^{(lo)}) \\
 &= 22.83 - 0.618(22.83 - 8.00) \\
 &\approx 13.67
 \end{aligned}$$

The process now repeats for iteration $t = 1$.

Computation continues until interval $[x^{(lo)}, x^{(hi)}]$ has sufficiently small length. In Table 16.3 stopping was set to occur when

$$x^{(hi)} - x^{(lo)} < \epsilon = 0.5$$

which happened at $t = 9$. Then our estimate of the optimal solution is midpoint

$$x^* \leftarrow \frac{1}{2}(x^{(lo)} + x^{(hi)}) = \frac{1}{2}(15.20 + 15.51) = 15.36$$

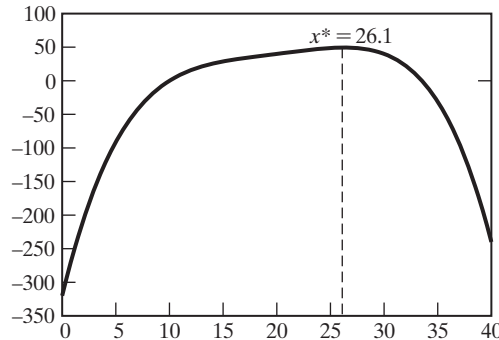
If greater accuracy were desired, we would need to continue through more iterations.

EXAMPLE 16.6: APPLYING GOLDEN SECTION SEARCH

Beginning with interval $[0, 40]$, apply golden section search to the unconstrained unimodal nonlinear program

$$\max f(x) \triangleq 2x - \frac{(x - 20)^4}{500}$$

plotted below.



Continue until the interval containing an optimum has length at most 10.

Solution: The algorithm proceeds exactly as in Table 16.3 except that this model maximizes. The following table provides details:

t	$x^{(lo)}$	$x^{(1)}$	$x^{(2)}$	$x^{(hi)}$	$f(x^{(lo)})$	$f(x^{(1)})$	$f(x^{(2)})$	$f(x^{(hi)})$	$x^{(hi)} - x^{(lo)}$
0	0.00	15.28	24.72	40.00	-320.00	29.57	48.45	-240.00	40.00
1	15.28	24.72	30.54	40.00	29.57	48.45	36.27	-240.00	24.72
2	15.28	21.12	24.72	30.56	29.57	42.23	48.45	36.27	15.28
3	21.12	24.72	26.95	30.56	42.23	48.45	49.23	36.27	9.44

Termination occurs at $t = 3$ with $x^{(hi)} - x^{(lo)} = 9.44 < \varepsilon = 10$.

Bracketing and 3-Point Patterns

Golden section search begins solving a 1-variable model with an interval $[x^{(lo)}, x^{(hi)}]$ known to contain an optimum. But how do we determine such initial intervals; that is, how do we **bracket** an optimal solution before the main search begins?

Sometimes the initial bracket is given because the model includes implicit upper and lower bounds on the decision variable. Much more commonly, one endpoint is known and the other must be determined. For example, in line searches, where the single variable is the step size λ to apply to a chosen move direction, λ must be positive. Thus we begin with $x^{(lo)} = 0$.

To locate the corresponding $x^{(hi)}$ bracketing the optimum of a unimodal objective function requires a search for a **3-point pattern**.

Definition 16.11 In 1-dimensional optimization, a 3-point pattern is a collection of 3 decision variable values $x^{(lo)} < x^{(mid)} < x^{(hi)}$ with the objective value at $x^{(mid)}$ superior to that of the other two (greater for a maximize, lesser for a minimize).

Figure 16.9 illustrates for our USPS model (16.2). Points

$$x^{(lo)} = 8, \quad x^{(mid)} = 16, \quad x^{(hi)} = 32$$

surround the minimum in a 3-point pattern with

$$f(x^{(lo)}) = f(8) \approx 171.42 > f(x^{(mid)}) = f(16) \approx 167.63$$

$$f(x^{(hi)}) = f(32) \approx 173.38 > f(x^{(mid)}) = f(16) \approx 167.63$$

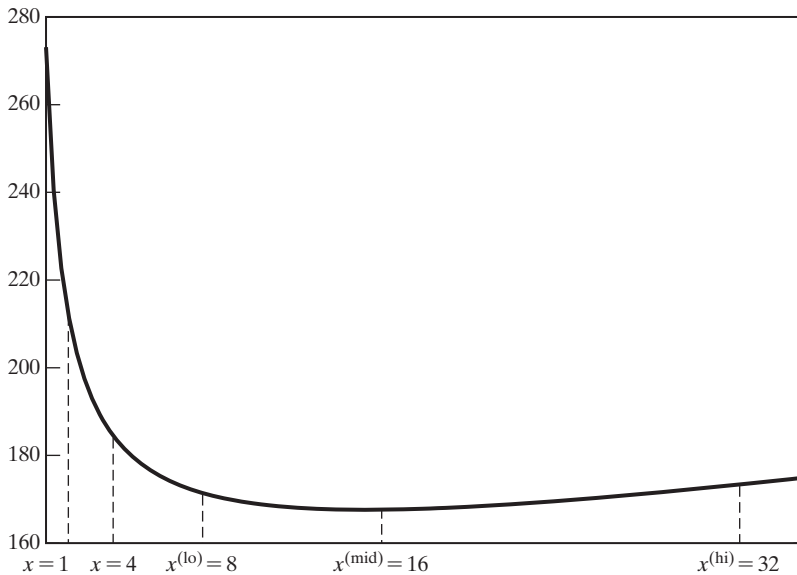


FIGURE 16.9 Bracketing the USPS Optimum with a 3-Point Pattern

A 3-point pattern provides the bracket we seek if the objective function is unimodal. Midpoint value $f(x^{(mid)})$ superior to $f(x^{(lo)})$ means that the function improves to the right of $x^{(lo)}$. Similarly, with $f(x^{(mid)})$ better than $f(x^{(hi)})$, the function improves to the left of $x^{(hi)}$. An optimum must lie in between.

Principle 16.12 If $\{x^{(lo)}, x^{(mid)}, x^{(hi)}\}$ is a 3-point pattern for unimodal objective function $f(x)$, there is an optimal x^* in the interval $[x^{(lo)}, x^{(hi)}]$.

Finding a 3-Point Pattern

Algorithm 16B details the most common scheme for quickly finding a 3-point pattern when we are given only an initial lower endpoint $x^{(lo)}$. Values of x are modified by exponentially changing step δ until the last three form a 3-point pattern.

ALGORITHM 16B: THREE-POINT PATTERN

Step 0: Initialization. Choose lower bound $x^{(lo)}$ on optimal solution x^* and initial step $\delta > 0$.

Step 1: Right or Left. If $f(x^{(lo)} + \delta)$ is superior to $f(x^{(lo)})$ (less for a minimize, greater for a maximize), set

$$x^{(mid)} \leftarrow x^{(lo)} + \delta$$

and go to Step 2 to search right. Otherwise, an optimum lies to the left; set

$$x^{(hi)} \leftarrow x^{(lo)} + \delta$$

and go to Step 3.

Step 2: Expand. Increase $\delta \leftarrow 2\delta$. If now $f(x^{(mid)})$ is superior to $f(x^{(mid)} + \delta)$, set

$$x^{(hi)} \leftarrow x^{(mid)} + \delta$$

and stop; $\{x^{(lo)}, x^{(mid)}, x^{(hi)}\}$ forms a 3-point pattern. Otherwise, update

$$x^{(lo)} \leftarrow x^{(mid)}$$

$$x^{(mid)} \leftarrow x^{(mid)} + \delta$$

and repeat Step 2.

Step 3: Reduce. Decrease $\delta \leftarrow \frac{1}{2}\delta$. If $f(x^{(lo)} + \delta)$ is now superior to $f(x^{(lo)})$, set

$$x^{(mid)} \leftarrow x^{(lo)} + \delta$$

and stop; $\{x^{(lo)}, x^{(mid)}, x^{(hi)}\}$ forms a 3-point pattern. Otherwise, update

$$x^{(hi)} \leftarrow x^{(lo)} + \delta$$

and repeat Step 3.

Values in Figure 16.9 illustrate the idea. Computation starts with $x^{(lo)} = \delta = 1$. Since

$$f(x^{(lo)} + \delta) = f(1 + 1) = f(2) \approx 212.16 < f(x^{(lo)}) = f(1) \approx 273.11$$

we must expand to the right to find a 3-point pattern bracketing the optimum. Setting $x^{(mid)} = 2$, we double δ and consider

$$x^{(mid)} + \delta = 2 + 2 = 4$$

The function improves again, so

$$x^{(lo)} \leftarrow x^{(mid)} = 2$$

$$x^{(mid)} \leftarrow x^{(mid)} + \delta = 4$$

and the process continues.

Eventually, we have $x^{(lo)} = 8$, $x^{(mid)} = 16$, and $\delta = 16$. Then

$$f(x^{(mid)}) = f(16) \approx 167.63 < f(x^{(mid)} + \delta) = f(32) \approx 173.63$$

and the algorithm stops after completing the 3-point pattern with

$$x^{(hi)} = x^{(mid)} + \delta = 32$$

EXAMPLE 16.7: FINDING A 3-POINT PATTERN

Return to the model

$$\max f(x) \triangleq 2x - \frac{(x - 20)^4}{500}$$

of Example 16.6, and apply Algorithm 16B to compute 3-point patterns with initial $x^{(lo)} = 0$ and (a) $\delta = 10$; (b) $\delta = 50$.

Solution:

(a) Initial $f(x^{(lo)}) = f(0) = -320$, and $f(x^{(lo)} + \delta) = f(10) = 0$ improves. Thus $x^{(mid)} \leftarrow 10$. Doubling δ and trying $f(x^{(mid)} + \delta) = f(30) = 40$ produces further improvement. Thus $x^{(lo)} \leftarrow x^{(mid)} = 10, x^{(mid)} \leftarrow 30$. Doubling δ again yields $f(x^{(mid)} + \delta) = f(30 + 40) = -12, 360$. Thus we stop with $x^{(hi)} \leftarrow 70$.

(b) Initial $f(x^{(lo)}) = f(0) = -320$, and $f(x^{(lo)} + \delta) = f(50) = -1520$ is worse. Setting $x^{(hi)} \leftarrow 50$, we could stop if our only purpose is to bracket the maximum. To complete a 3-point pattern, however, we must reduce δ . Halving to $\delta = 25$ produces $f(x^{(lo)} + \delta) = f(25) = 48.75$, which does improve on $f(x^{(lo)})$. Choosing $x^{(mid)} \leftarrow 25$ completes the 3-point pattern.

Quadratic Fit Search

Golden search Algorithm 16A is reliable, but its slow and steady narrowing of the optimum-containing interval can require considerable computation before an optimum is identified with sufficient accuracy. **Quadratic fit search** closes in much more rapidly by taking full advantage of a current 3-point pattern.

Given a 3-point pattern, we can fit a quadratic function through corresponding functional values that has a unique maximum or minimum, $x^{(qu)}$, whichever we are seeking for the given objective $f(x)$. Quadratic fit uses this approximation to improve the current 3-point pattern by replacing one of its points with approximate optimum $x^{(qu)}$.

Figure 16.10 illustrates for USPS model (16.2) with initial 3-point pattern

$$x^{(lo)} = 8, \quad x^{(mid)} = 20, \quad x^{(hi)} = 32$$

The main curve plots actual objective function $f(x)$. A second, dashed line shows the unique quadratic function fitting through the 3 pattern points.

That quadratic approximation has a minimum at

$$x^{(qu)} \approx 18.56 \quad \text{with} \quad f(x^{(qu)}) \approx 167.98 \tag{16.14}$$

Together with the current $x^{(lo)}$ and $x^{(mid)}$ it now forms a new 3-point pattern

$$x^{(lo)} = 8, \quad x^{(mid)} = 18.56, \quad x^{(hi)} = 20$$

with a smaller interval [8,20]. Repeating in this way isolates an optimum for $f(x)$ in an ever-narrowing range.

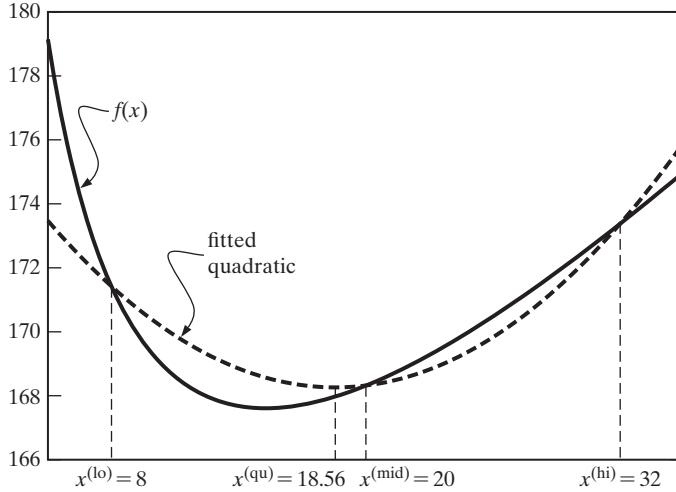


FIGURE 16.10 Quadratic Fit Search of USPS Application

The computation to determine $x^{(qu)}$ requires only some tedious algebra.

Principle 16.13 | The unique optimum of a quadratic function agreeing with $f(x)$ at 3-point pattern $\{x^{(lo)}, x^{(mid)}, x^{(hi)}\}$ occurs at

$$x^{(qu)} \triangleq \frac{\frac{1}{2} f^{(lo)} [s^{(mid)} - s^{(hi)}] + f^{(mid)} [s^{(hi)} - s^{(lo)}] + f^{(hi)} [s^{(lo)} - s^{(mid)}]}{f^{(lo)} [x^{(mid)} - x^{(hi)}] + f^{(mid)} [x^{(hi)} - x^{(lo)}] + f^{(hi)} [x^{(lo)} - x^{(mid)}}$$

Where $f^{(lo)} \triangleq f(x^{(lo)})$, $f^{(mid)} \triangleq f(x^{(mid)})$, $f^{(hi)} \triangleq f(x^{(hi)})$, $s^{(lo)} \triangleq (x^{(lo)})^2$, $s^{(mid)} \triangleq (x^{(mid)})^2$, and $s^{(hi)} \triangleq (x^{(hi)})^2$.

For example, the approximate minimum of expression (16.14) is

$$x^{(qu)} = \frac{1}{2} \frac{171.42[(20)^2 - (32)^2] + 168.32[(32)^2 - (8)^2] + 173.38[(8)^2 - (20)^2]}{171.42[20 - 32] + 168.32[32 - 8] + 173.38[8 - 20]} \approx 18.56$$

Quadratic Fit Solution of USPS Application

Algorithm 16C details a quadratic fit procedure for 1-dimensional search, and Table 16.4 tracks progress for our USPS application. The reader can verify that each iteration produces a new 3-point pattern and that interval $[x^{(lo)}, x^{(hi)}]$ narrows constantly. Computation for Table 16.4 was stopped when that interval had length at most $\epsilon = 0.50$.

One new element arises when the computed $x^{(qu)}$ happens to nearly coincide with the current $x^{(mid)}$. If nothing were done, the algorithm would loop forever. Step 3 of Algorithm 16C addresses this difficulty by perturbing $x^{(qu)}$ by $\epsilon/2$ toward the most distant endpoint.

The one value changed in this way in Table 16.4 is marked with an asterisk (*) at $t = 7$. There formula [\[16.13\]](#) produced $x^{(qu)} = 15.34$, which was too close to

ALGORITHM 16C: QUADRATIC FIT SEARCH

Step 0: Initialization. Choose starting 3-point pattern $\{x^{(lo)}, x^{(mid)}, x^{(hi)}\}$ along with a stopping tolerance $\epsilon > 0$, and initialize iteration counter $t \leftarrow 0$.

Step 1: Stopping. If $(x^{(hi)} - x^{(lo)}) \leq \epsilon$, stop and report approximate optimal solution $x^{(mid)}$.

Step 2: Quadratic Fit. Compute quadratic fit optimum $x^{(qu)}$ according to formula [16.13]. Then if $x^{(qu)} \approx x^{(mid)}$, go to Step 3; if $x^{(qu)} < x^{(mid)}$, go to Step 4; and if $x^{(qu)} > x^{(mid)}$, go to Step 5.

Step 3: Coincide. New $x^{(qu)}$ coincides essentially with current $x^{(mid)}$. If $x^{(mid)}$ is farther from $x^{(lo)}$ than from $x^{(hi)}$, preturb left

$$x^{(qu)} \leftarrow x^{(mid)} - \frac{\epsilon}{2}$$

and proceed to Step 4. Otherwise, adjust right

$$x^{(qu)} \leftarrow x^{(mid)} + \frac{\epsilon}{2}$$

and proceed to Step 5.

Step 4: Left. If $f(x^{(mid)})$ is superior to $f(x^{(qu)})$ (less for a minimize, greater for a maximize), then update

$$x^{(lo)} \leftarrow x^{(qu)}$$

Otherwise, replace

$$x^{(hi)} \leftarrow x^{(mid)}$$

$$x^{(mid)} \leftarrow x^{(qu)}$$

Either way, advance $t \leftarrow t + 1$, and return to Step 1.

Step 5: Right. If $f(x^{(mid)})$ is superior to $f(x^{(qu)})$ (less for a minimize, greater for a maximize), then update

$$x^{(hi)} \leftarrow x^{(qu)}$$

Otherwise, replace

$$x^{(lo)} \leftarrow x^{(mid)}$$

$$x^{(mid)} \leftarrow x^{(qu)}$$

Either way advance $t \leftarrow t + 1$, and return to Step 1.

TABLE 16.4 Quadratic Fit Solution of USPS Application

t	$x^{(lo)}$	$x^{(mid)}$	$x^{(hi)}$	$f(x^{(lo)})$	$f(x^{(mid)})$	$f(x^{(hi)})$	$x^{(hi)} - x^{(lo)}$	$x^{(qu)}$	$f(x^{(qu)})$
0	8.00	20.00	32.00	171.42	168.32	173.38	24.00	18.56	167.98
1	8.00	18.56	20.00	171.42	167.98	168.32	12.00	16.75	167.70
2	8.00	16.75	18.56	171.42	167.70	167.98	10.56	16.23	167.65
3	8.00	16.23	16.75	171.42	167.65	167.70	8.75	15.78	167.62
4	8.00	15.78	16.23	171.42	167.62	167.65	8.23	15.59	167.62
5	8.00	15.59	15.78	171.42	167.62	167.62	7.78	15.45	167.62
6	8.00	15.45	15.59	171.42	167.62	167.62	7.59	15.38	167.61
7	8.00	15.38	15.45	171.42	167.61	167.62	7.45	*15.13	167.62
8	15.13	[15.38]	15.45	167.62	167.61	167.62	0.32	—	—

$x^{(\text{mid})} = 15.38$. With $x^{(\text{lo})} = 8.00$ farther from this value than $x^{(\text{hi})} = 15.45$, Step 3 perturbed the computed value to

$$x^{(\text{qu})} = x^{(\text{mid})} - \frac{\epsilon}{2} = 15.38 - 0.25 = 15.13$$

EXAMPLE 16.8: APPLYING QUADRATIC FIT SEARCH

Return to the unconstrained nonlinear program of Example 16.6:

$$\max f(x) \triangleq 2x - \frac{(x - 20)^4}{500}$$

Using initial 3-point pattern $x^{(\text{lo})} = 0$, $x^{(\text{mid})} = 32$, $x^{(\text{hi})} = 40$, apply quadratic fit Algorithm 16C to identify an optimal solution within an interval $[x^{(\text{lo})}, x^{(\text{hi})}]$ of length at most 10.

Solution: Computation parallels Table 16.4 except that this model maximizes. Details are contained in the following table:

t	$x^{(\text{lo})}$	$x^{(\text{mid})}$	$x^{(\text{hi})}$	$f(x^{(\text{lo})})$	$f(x^{(\text{mid})})$	$f(x^{(\text{hi})})$	$x^{(\text{hi})} - x^{(\text{lo})}$	$x^{(\text{qu})}$	$f(x^{(\text{qu})})$
0	0.00	32.00	40.00	-320.00	22.53	-240.00	40.00	20.92	41.84
1	0.00	20.92	32.00	-320.00	41.84	22.53	32.00	25.00	48.75
2	20.92	25.00	32.00	41.84	48.75	22.53	11.80	*30.00	40.03
3	20.92	<u>25.00</u>	30.00	41.84	48.75	40.03	9.08	—	—

16.3 DERIVATIVES, TAYLOR SERIES, AND CONDITIONS FOR LOCAL OPTIMA IN MULTIPLE DIMENSIONS

Unconstrained nonlinear optimization is certainly possible without derivatives. Still, where derivatives are readily available, they can tell us a great deal about a model and substantially accelerate search algorithm progress (principle [16.8](#)). This section develops some of the most important insights to be gained.

Improving Search Paradigm

In Sections 3.1 and 3.2 we introduced the principle of **improving search** (Algorithm 3A, Section 3.2) on which almost all nonlinear algorithms are based. We begin with a (vector) solution $\mathbf{x}^{(0)}$ satisfying all model constraints. In the unconstrained context of this chapter, any $\mathbf{x}^{(0)}$ will do. Iterations t advance current solution $\mathbf{x}^{(t)}$ to

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$$

where $\Delta \mathbf{x}$ is a **move direction** and λ a positive **step size**. Each $\Delta \mathbf{x}$ should be an **improving direction**; that is, it should produce immediate objective function improvement (definition [3.13](#)). (It should also retain feasibility when constraints are present.) The process continues until a point is reached where no directions lead to such immediate improvement. There we stop (principle [3.17](#)) with what is usually a **local optimum**—a point as good in objective value as any nearby (definition [3.5](#)). [See Figure 3.8(a) for an exception that is not a local optimum even though it admits no improving direction.]

Figure 16.11 illustrates for our (minimizing) Custom Computers model (16.7). At initial point $\mathbf{x}^{(0)} = (32, -0.4)$ that search took a step of $\lambda = \frac{1}{2}$ in direction $\Delta \mathbf{x} = (2, -0.2)$ to produce

$$\mathbf{x}^{(1)} = \begin{pmatrix} 32 \\ -0.4 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 2 \\ -0.2 \end{pmatrix} = \begin{pmatrix} 33 \\ -0.5 \end{pmatrix}$$

Dashed lines in the figure, which show **contours** of the objective function plotted in Figure 16.3, demonstrate that the move is improving. Even very small steps from $\mathbf{x}^{(0)}$ in direction $\Delta \mathbf{x}$ advance the search to lower contours of the objective function.

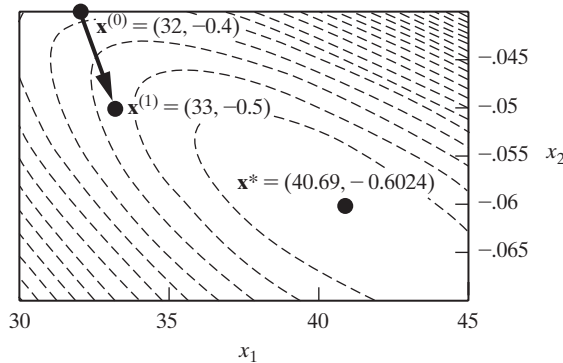


FIGURE 16.11 Improving Search of the Custom Computers Application

Local Information and Neighborhoods

What move direction should the search of Figure 16.11 adopt next? The best choice would make

$$\Delta \mathbf{x} = \mathbf{x}^* - \mathbf{x}^{(1)} = \begin{pmatrix} 40.69 \\ -0.6024 \end{pmatrix} - \begin{pmatrix} 33 \\ -0.5 \end{pmatrix} = \begin{pmatrix} 7.69 \\ -0.1024 \end{pmatrix}$$

which leads directly to the optimal solution.

Unfortunately, a search in progress does not have the global viewpoint available in Figure 16.11. The next move must be chosen using only experience with points already visited (here $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$) plus local information about the shape of the objective function in the immediate **neighborhood** (definition 3.4) of current $\mathbf{x}^{(1)}$.

First Derivatives and Gradients

We know from elementary calculus (see also Section 3.3 and Primer 2) that **first derivatives** or **gradients** provide information about how an objective function changes near a current solution $\mathbf{x}^{(t)}$.

Principle 16.14 | The first derivative $f'(x)$ of a single-variable objective function $f(x)$, or the gradient vector $\nabla f(\mathbf{x})$ of first partial derivatives $\partial f/\partial x_1, \dots, \partial f/\partial x_n$ with n variables, describes the slope or rate of change in f with small increments in current decision variable values.

For instance, at $\mathbf{x}^{(1)} = (33, -0.5)$ in minimizing Figure 16.11, we may apply partial derivative expressions (16.12) to compute

$$\frac{\partial f}{\partial x_1} \approx -23.07, \quad \frac{\partial f}{\partial x_2} \approx -174.23, \quad \text{so that} \quad \nabla f(\mathbf{x}^{(1)}) \approx (-23.07, -174.23)$$

Thus small increments from either $x_1 = 33$ or $x_2 = -0.5$ decrease $f(x_1, x_2)$, but the rate of change is much more rapid with increments in x_2 .

Second Derivatives and Hessian Matrices

When an objective function is twice differentiable, which is typical for the smooth objectives most often occurring in applications, **second derivatives** can tell us still more about the shape of the function in the neighborhood of current solution $\mathbf{x}^{(t)}$. Primer 7 reviews some of the fundamentals.

The second derivative of a single-variable objective f is a scalar function $f''(x)$. For an n -variable objective, there is a whole **Hessian matrix** of second partial derivatives with row i , column j entry $\partial^2 f / \partial x_i \partial x_j$. For example, our Custom Computer objective [model (16.7)]

$$f(x_1, x_2) \triangleq \sum_{i=1}^m (q_i - x_1 p_i^{x_2})^2$$

has first partial derivatives [expression (16.12)]

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= -2 \sum_{i=1}^m (q_i - x_1 p_i^{x_2}) p_i^{x_2} \\ \frac{\partial f}{\partial x_2} &= -2 \sum_{i=1}^m (q_i - x_1 p_i^{x_2}) (x_1 p_i^{x_2}) \ln(p_i) \end{aligned}$$

Thus second partials are

$$\begin{aligned} \frac{\partial^2 f}{\partial x_1^2} &= 2 \sum_{i=1}^m p_i^{2x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} &= \frac{\partial^2 f}{\partial x_2 \partial x_1} \\ &= -2 \sum_{i=1}^m [(q_i - x_1 p_i^{x_2}) (p_i^{x_2}) \ln(p_i) - (p_i^{x_2}) (x_1 p_i^{x_2}) \ln(p_i)] \\ \frac{\partial^2 f}{\partial x_2^2} &= -2 \sum_{i=1}^m \ln^2(p_i) [(q_i - x_1 p_i^{x_2}) (x_1 p_i^{x_2}) - (x_1 p_i^{x_2})^2] \end{aligned} \tag{16.15}$$

At the $\mathbf{x}^{(1)} = (33, -0.5)$ of Figure 16.11 constants p_i and q_i of Table 16.1 yield the Hessian matrix

$$\nabla^2 f(33, -0.5) \triangleq \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} \approx \begin{pmatrix} 5.77 & 179.65 \\ 179.65 & 11,003.12 \end{pmatrix}$$

PRIMER 7: SECOND DERIVATIVES AND HESSIAN MATRICES

Primer 2 (Section 3.3) provides a brief overview of first derivatives df/dx [or $f'(x)$] and first partial derivatives $\partial f/\partial x_j$, which measure the rate of change function f with respect to increases in its arguments. The vector of partial derivatives for n -variable function $f(\mathbf{x})$ is its **gradient** $\nabla f(\mathbf{x})$.

First derivatives or partial derivatives of function f are themselves functions of its arguments. If such derivative functions are also differentiable, f is said to be **twice differentiable**, and we may determine second derivatives. Second derivatives describe the rate of change in slopes (i.e., the curvature of f).

Second derivatives of a single-variable $f(x)$ are customarily denoted d^2f/dx^2 or $f''(x)$. For example, $f(x) \triangleq 3x^4$ has first derivative $f'(x) = 12x^3$ and second $f''(x) = 36x^2$. At $x = 2$, $df/dx = 12(2)^3 = 96$, while $d^2f/dx^2 = 36(2)^2 = 144$.

Twice differentiable functions $f(\mathbf{x}) \triangleq f(x_1, \dots, x_n)$ of n variables have **second partial derivatives** for each pair of variables x_i and x_j . Such second partial derivatives are customarily denoted $\partial^2 f/\partial x_i \partial x_j$ when $i \neq j$ and $\partial^2 f/\partial x_i^2$ if $i = j$. The order in which variables are listed indicates the sequence of differentiation. That is,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \triangleq \frac{\partial}{\partial x_j} \left(\frac{\partial f}{\partial x_i} \right)$$

To illustrate, consider $f(x_1, x_2) \triangleq 5x_1(x_2)^3$. First partial derivatives are $\partial f/\partial x_1 = 5(x_2)^3$ and $\partial f/\partial x_2 = 15x_1(x_2)^2$. Thus

$$\frac{\partial^2 f}{\partial x_1^2} = 0, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 15(x_2)^2, \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 15(x_2)^2, \quad \frac{\partial^2 f}{\partial x_2^2} = 30x_1x_2$$

Notice in this example that $\partial^2 f/\partial x_1 \partial x_2 = \partial^2 f/\partial x_2 \partial x_1$. It is always true that

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

when f and all its first partial derivatives are continuous functions.

It is often convenient to deal with second partial derivatives in a Hessian matrix denoted $\nabla^2 f(\mathbf{x})$ and defined

$$\nabla^2 f(x_1, \dots, x_n) \triangleq \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

For instance, at $x_1 = -3, x_2 = 2$, the $f(x_1, x_2)$ above has Hessian matrix

$$\nabla^2 f(-3, 2) = \begin{pmatrix} 0 & 15(2)^2 \\ 15(2)^2 & 30(-3)(2) \end{pmatrix} = \begin{pmatrix} 0 & 60 \\ 60 & -180 \end{pmatrix}$$

What second derivatives offer a search algorithm is information about the curvature of objective function f near current solution $\mathbf{x}^{(t)}$.

Principle 16.15 The second derivative $\mathbf{x}^{(t)}$ of a single-variable objective function $f(x)$, or the Hessian matrix $\nabla^2 f(x)$ of second partial derivatives of $\partial^2 f / \partial x_i \partial x_j$ for n variables describes the change in slope or curvature of f in the neighborhood of current decision variable values.

For example, the $\partial^2 f / \partial x_2^2 = 11,003.12$ confirms what we can see in Figure 16.11—that small changes in x_2 dramatically affect the slope of f near $\mathbf{x}^{(1)}$. The much smaller $\partial^2 f / \partial x_1^2 = 5.77$ indicates the function is flatter in the x_1 dimension.

Taylor Series Approximations with One Variable

A more concise description of what derivatives tell us about an objective function follows from classic **Taylor series**. For 1-dimensional function $f(x)$, Taylor's approximation represents the impact of a change λ from current $x^{(t)}$ as

$$f(x^{(t)} + \lambda) \approx f(x^{(t)}) + \frac{\lambda}{1!} f'(x^{(t)}) + \frac{\lambda^2}{2!} f''(x^{(t)}) + \frac{\lambda^3}{3!} f'''(x^{(t)}) + \dots \quad (16.16)$$

where $f'(x)$ is the first derivative of f , $f''(x)$ is the second derivative, and so on.

To illustrate, consider

$$f(x) \triangleq e^{3x-6} \quad (16.17)$$

for which $f'(x) = 3e^{3x-6}$, $f''(x) = 9e^{3x-6}$, and $f'''(x) = 27e^{3x-6}$. Near $x^{(t)} = 2$, derivatives approximate the impact of a change λ as

$$\begin{aligned} f(2 + \lambda) &\approx f(2) + \frac{\lambda}{1!} f'(2) + \frac{\lambda^2}{2!} f''(2) + \frac{\lambda^3}{3!} f'''(2) + \dots \\ &= 1 + 3\lambda + \frac{9}{2} \lambda^2 + \frac{27}{6} \lambda^3 + \dots \end{aligned}$$

Notice that as $|\lambda| \rightarrow 0$, higher powers of λ approach zero the most rapidly. That is why we may approximate a function with just the first few terms of expression (16.16) if our interest centers on the immediate neighborhood of current $x^{(t)}$. The results are the first-order or linear, and second-order or quadratic approximations to a function of a single variable.

Definition 16.16 The **first-order** or **linear**, and **second-order** or **quadratic Taylor series approximations** to single-variable function $f(x)$ near $x = x^{(t)}$ are, respectively,

$$f_1(x^{(t)} + \lambda) \triangleq f(x^{(t)}) + \lambda f'(x^{(t)})$$

and

$$f_2(x^{(t)} + \lambda) \triangleq f(x^{(t)}) + \lambda f'(x^{(t)}) + \frac{1}{2} \lambda^2 f''(x^{(t)})$$

where λ is the amount of change, f' is the first derivative of f , and f'' is the second.

Figure 16.12 illustrates for the $f(x) \triangleq e^{3x-6}$ of expression (16.17). Part (a) plots $f(x)$ and the first-order approximation for current $x^{(t)} = 2$

$$f_1(2 + \lambda) = f(2) + \lambda f'(2) = 1 + 3\lambda$$

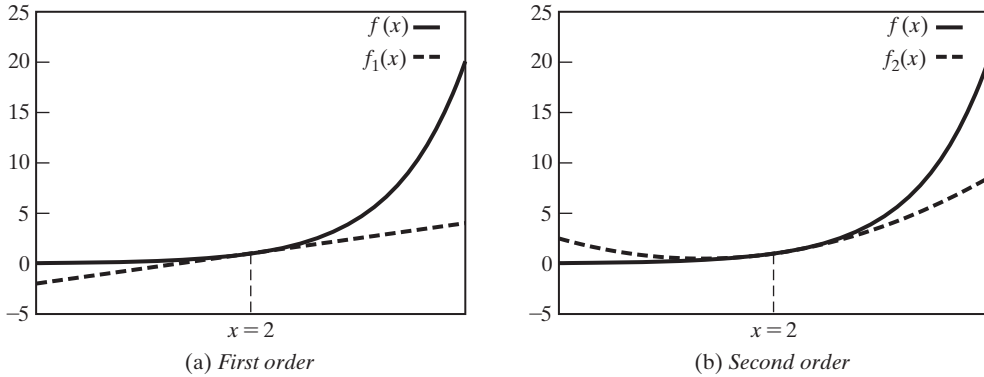


FIGURE 16.12 First- and Second-Order Taylor Series Approximations

and part (b) shows $f(x)$ versus second-order approximation

$$f_2(2 + \lambda) = f(2) + \lambda f'(2) + \frac{1}{2}\lambda^2 f''(2) = 1 + 3\lambda + \frac{9}{2}\lambda^2$$

Notice that the first-order approximation is a linear function of change λ . It assumes that the slope at $x^{(t)} = 2$ remains constant. Both approximations are fairly accurate near $\lambda = 0$ and deteriorate as λ becomes larger. Still, the second-order approximation in part (b) comes somewhat closer to the real f because it incorporates the curvature information in second derivative $f''(x^{(t)})$.

Taylor Series Approximations with Multiple Variables

We may extend Taylor series approximations to functions of more than one variable by using first and second partial derivatives.

Definition 16.17 The first-order or linear, and second-order or quadratic Taylor series approximations to n -variable function $f(\mathbf{x}) \triangleq f(x_1, \dots, x_n)$ at point $\mathbf{x}^{(t)}$ are, respectively,

$$\begin{aligned} f_1(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} \\ &\triangleq f(\mathbf{x}^{(t)}) + \lambda \sum_{j=1}^n \left(\frac{\partial f}{\partial x_j} \right) \Delta x_j \end{aligned}$$

and

$$\begin{aligned} f_2(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} + \frac{\lambda^2}{2} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} \\ &\triangleq f(\mathbf{x}^{(t)}) + \lambda \sum_{j=1}^n \left(\frac{\partial f}{\partial x_j} \right) \Delta x_j + \frac{\lambda^2}{2} \sum_{i=1}^n \sum_{j=1}^n \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right) \Delta x_i \Delta x_j \end{aligned}$$

where $\Delta \mathbf{x} \triangleq (\Delta x_1, \dots, \Delta x_n)$ is a direction of change, λ is the applied step size, $\nabla f(\mathbf{x}^{(t)})$ is the gradient of f at $\mathbf{x}^{(t)}$, and $\nabla^2 f(\mathbf{x}^{(t)})$ is the corresponding Hessian matrix.

As with 1-dimensional series (16.16), there are higher-order terms in the full Taylor series expansion of an n -variate function, but they become insignificant as $|\lambda| \rightarrow 0$.

To illustrate [16.17], consider

$$f(x_1, x_2) \triangleq x_1 \ln(x_2) + 2$$

at $\mathbf{x}^{(t)} = (-3, 1)$. There $f(-3, 1) = 2$, and gradient

$$\nabla f(-3, 1) \triangleq \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} \ln(x_2) \\ \frac{x_1}{x_2} \end{pmatrix} = \begin{pmatrix} 0 \\ -3 \end{pmatrix}$$

Thus the first-order approximation to $f(x_1, x_2)$ near $\mathbf{x}^{(t)} = (-3, 1)$ in direction $\Delta \mathbf{x} \triangleq (\Delta x_1, \Delta x_2)$ is

$$\begin{aligned} f_1(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} \\ &= 2 + \lambda(0, -3) \cdot (\Delta x_1, \Delta x_2) \\ &= 2 - 3\lambda \Delta x_2 \end{aligned}$$

To improve the approximation with second-order terms, we compute Hessian

$$\nabla^2 f(-3, 1) = \begin{pmatrix} 0 & \frac{1}{x_2} \\ \frac{1}{x_2} & -\frac{x_1}{(x_2)^2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix}$$

Then

$$\begin{aligned} f_2(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} + \frac{\lambda^2}{2} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} \\ &= 2 + \lambda(0, -3) \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} + \frac{\lambda^2}{2} (\Delta x_1, \Delta x_2) \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} \\ &= 2 - 3\lambda \Delta x_2 + \lambda^2 \Delta x_1 \Delta x_2 + \frac{3}{2} \lambda^2 (\Delta x_2)^2 \end{aligned}$$

Stationary Points and Local Optima

First and second derivatives tell us a great deal about whether a solution is a local optimum. Begin with stationary points.

Definition 16.18 | Solution \mathbf{x} is a **stationary point** of smooth function f if $\nabla f(\mathbf{x}) = \mathbf{0}$.

That is, stationary points are solutions where all first (partial) derivatives equal zero. Figure 16.13 illustrates for

$$f(x_1, x_2) \triangleq 40 + (x_1)^3(x_1 - 4) + 3(x_2 - 5)^2 \quad (16.18)$$

Partial derivatives are

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= (x_1)^2(4x_1 - 12) \\ \frac{\partial f}{\partial x_2} &= 6(x_2 - 5) \end{aligned} \quad (16.19)$$

It is easy to check that they become zero at two stationary points:

$$\mathbf{x}^{(1)} = (3, 5) \quad \text{and} \quad \mathbf{x}^{(2)} = (0, 5) \quad (16.20)$$

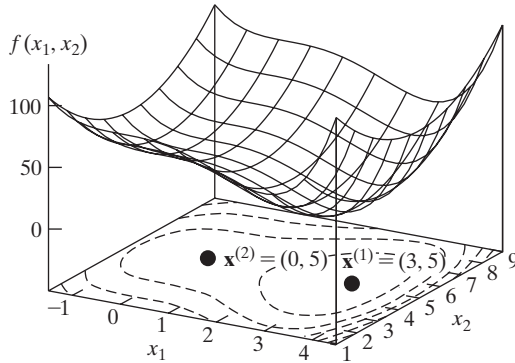


FIGURE 16.13 Stationary Points of a Minimize Objective

We can see in Figure 16.13 that one of these, $\mathbf{x}^{(1)}$, is a local (and here also global) minimum of f . This suggests our first (so-called **first-order necessary**) condition for an unconstrained local optima.

Principle 16.19 Every unconstrained local optimum of a smooth objective function must be a stationary point.

The reason that condition [16.19] must hold in every case is that a nonzero gradient $\nabla f(\mathbf{x}^{(t)})$ itself provides an improving direction at $\mathbf{x}^{(t)}$. Following principle [3.23], we may adopt $\Delta \mathbf{x} = \pm \nabla f(\mathbf{x}^{(t)})$ with + for maximize problems and - to minimize. Then the first-order Taylor series approximation [16.17] gives

$$\begin{aligned}
 f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\approx f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} \\
 &= f(\mathbf{x}^{(t)}) \pm \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \nabla f(\mathbf{x}^{(t)}) \\
 &= f(\mathbf{x}^{(t)}) \pm \lambda \sum_{j=1}^n \left(\frac{\partial f}{\partial x_j} \right)^2
 \end{aligned}
 \tag{16.21}$$

This is an improvement in the objective value unless all partial derivatives = 0, and we know that for λ sufficiently small the first-order part of the Taylor series expansion dominates all other terms.

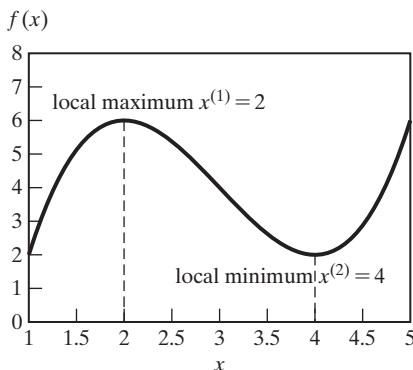
EXAMPLE 16.9: VERIFYING LOCAL OPTIMA AS STATIONARY POINTS

Consider the single-variable function

$$f(x) = x^3 - 9x^2 + 24x - 14$$

Plot the function for $1 \leq x \leq 5$ and verify that local maximum $x^{(1)} = 2$ and local minimum $x^{(2)} = 4$ are both stationary points.

Solution: A plot of the function is as follows:



Its first derivative is

$$f'(x) = 3x^2 - 18x + 24$$

Both $f'(x^{(1)}) = f'(2) = 0$ and $f'(x^{(2)}) = f'(4) = 0$, confirming that both are stationary points.

Saddle Points

Look again at Figure 16.13. Stationary point $\mathbf{x}^{(1)} = (3, 5)$ is a local minimum, but $\mathbf{x}^{(2)} = (0, 5)$ is not. Increasing x_1 reduces the objective at the latter. Point $\mathbf{x}^{(2)}$ is also not a local maximum. Increasing x_2 makes the objective value bigger. Figure 16.14 shows that the remaining possibility is a saddle point.

Definition 16.20 A **saddle point** is a stationary point that is neither a local maximum nor a local minimum.

Every stationary point is either a local maximum, a local minimum, or a saddle point.

Saddle points get their name from the saddlelike possibility of the 2-dimensional case in Figure 16.14(c). The same stationary point is a local maximum in one dimension and a local minimum in another, yet neither a local maximum nor a local minimum when both directions are considered together.

Hessian Matrices and Local Optima

To distinguish better among the 3 types of stationary points in Figure 16.14 we must look at second (partial) derivatives. At stationary points, which have $\nabla f(\mathbf{x}^{(t)}) = \mathbf{0}$, second-order Taylor approximation [16.17] simplifies as

$$\begin{aligned} f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\approx f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} + \frac{\lambda^2}{2} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} \\ &= f(\mathbf{x}^{(t)}) + 0 + \frac{\lambda^2}{\lambda} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} \end{aligned} \quad (16.22)$$

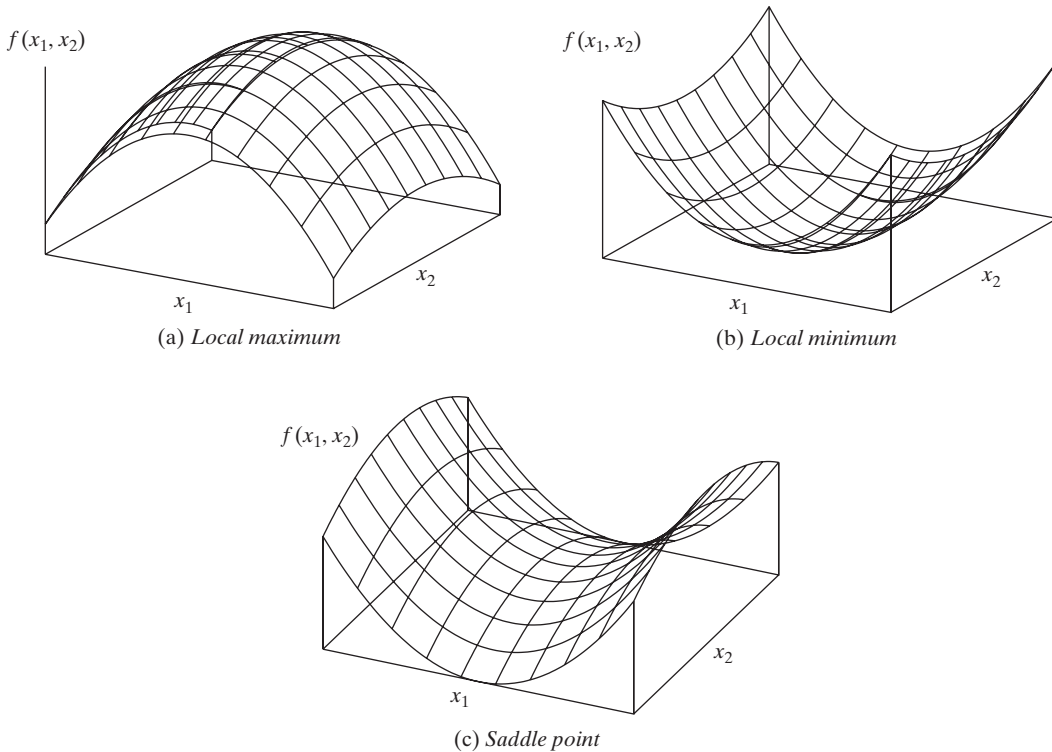


FIGURE 16.14 Three Forms of Stationary Points

Thus (nonzero) Hessian-based quadratic forms $\Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x}$ critically influence whether improving directions $\Delta \mathbf{x}$ exist at stationary points (i.e., whether such points have any chance of being local optima).

Consider, for example, a direction $\Delta \mathbf{x}$ with $\Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} < 0$ at stationary point $\mathbf{x}^{(t)}$. Quadratic approximation (16.22) implies that

$$\begin{aligned}
 f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) &\approx f(\mathbf{x}^{(t)}) + \frac{\lambda^2}{2} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} \\
 &< f(\mathbf{x}^{(t)})
 \end{aligned}$$

We may conclude that $\Delta \mathbf{x}$ is an improving direction for minimize problems at $\mathbf{x}^{(t)}$ because moves in direction $\Delta \mathbf{x}$ strictly reduce the objective value if λ is small enough for this quadratic Taylor approximation to dominate higher-order terms. With a descent direction at hand, stationary point $\mathbf{x}^{(t)}$ could not be a local minimum (principle [3.16](#)).

The ponderously named **positive** and **negative (semi)definite** properties of square matrices, which are reviewed briefly in Primer 8, address such sign issues in quadratic forms. Combining with Taylor expression (16.22), we may use these properties to distinguish among stationary points. Semidefinite forms provide **second-order necessary optimality conditions**.

PRIMER 8: POSITIVE AND NEGATIVE (SEMI) DEFINITE MATRICES

Single-variable **quadratic form** $dad = ad^2$ is positive for all $d \neq 0$ if constant $a > 0$ and negative for all $d \neq 0$ if $a < 0$. In a similar way, whether n -variable quadratic form $\mathbf{dMd} \triangleq \sum_{i=1}^n \sum_{j=1}^n m_{i,j} d_i d_j$ is positive or negative depends on properties of the matrix \mathbf{M} .

Square matrix \mathbf{M} is said to be **positive definite** if $\mathbf{dMd} > 0$ for all $\mathbf{d} \neq \mathbf{0}$, and **positive semidefinite** if $\mathbf{dMd} \geq 0$ for all \mathbf{d} . Similarly, \mathbf{M} is **negative definite** if $\mathbf{dMd} < 0$ for all $\mathbf{d} \neq \mathbf{0}$, and **negative semidefinite** if $\mathbf{dMd} \leq 0$ for all \mathbf{d} .

To illustrate, consider

$$\mathbf{A} \triangleq \begin{pmatrix} 3 & 0 \\ 0 & 8 \end{pmatrix}, \quad \mathbf{B} \triangleq \begin{pmatrix} -1 & 2 \\ 2 & -4 \end{pmatrix}, \quad \mathbf{C} \triangleq \begin{pmatrix} 3 & 0 \\ 0 & -8 \end{pmatrix}$$

Matrix \mathbf{A} is positive definite because

$$\mathbf{dAd} = 3(d_1)^2 + 8(d_2)^2$$

which is positive for every nonzero \mathbf{d} . Similarly, \mathbf{B} is negative semi-definite because

$$\mathbf{dBd} = -(d_1)^2 + 4d_1d_2 - 4(d_2)^2 = -(d_1 - 2d_2)^2 \leq 0$$

But with $\mathbf{dBd} = 0$ for $d_1 = 2d_2$, \mathbf{B} is not negative definite. Matrix \mathbf{C} is neither positive nor negative definite or semidefinite.

Obviously, \mathbf{M} being positive (or negative) definite implies that \mathbf{M} is positive (negative) semidefinite, so that example \mathbf{A} is positive semidefinite. Conversely, a positive (or negative) semidefinite matrix that is also symmetric (Primer 4) and nonsingular (Primer 5) is positive (negative) definite. Thus example \mathbf{A} being positive semidefinite, symmetric, and nonsingular proves that it is positive definite. Also, if \mathbf{M} is positive (semi)definite, then $-\mathbf{M}$ is negative (semi)definite, and vice versa.

One way to test whether a symmetric matrix \mathbf{M} satisfies any of these definitions is to check the determinants (Primer 5) of its **principal submatrices** [i.e., the submatrices made up of its first k rows and columns ($k = 1, \dots, n$)]. Symmetric matrix \mathbf{M} is positive definite if all such principal determinants are positive, and positive semidefinite if they are all nonnegative. Similarly, symmetric \mathbf{M} is negative definite if the principal determinants are nonzero and alternating in sign with the first negative; negative semidefinite allows zeros.

For example, one \mathbf{D} and its principal submatrix determinants are

$$\mathbf{D} \triangleq \begin{pmatrix} 5 & -2 & 0 \\ -2 & 3 & 0 \\ 0 & 0 & 8 \end{pmatrix}, \quad \det(5) = 5, \quad \det \begin{pmatrix} 5 & -2 \\ -2 & 3 \end{pmatrix} = 11, \quad \det \begin{pmatrix} 5 & -2 & 0 \\ -2 & 3 & 0 \\ 0 & 0 & 8 \end{pmatrix} = 88$$

Since all determinants are positive, \mathbf{D} is positive definite. On the other hand, principal determinants of example \mathbf{B} above are -1 and 0 . Alternating signs with the first nonpositive confirm that \mathbf{B} is negative semidefinite.

Principle 16.21 | The Hessian matrix of a smooth function f is negative semidefinite at every unconstrained local maximum and positive semidefinite at every unconstrained local minimum.

The stronger definite forms give **sufficient conditions**.

Principle 16.22 | A stationary point of a smooth function f is an unconstrained local maximum if the Hessian matrix at the point is negative definite. A stationary point is an unconstrained local minimum if the Hessian matrix is positive definite.

We may illustrate principles [16.21](#) and [16.22](#) by testing the two stationary points of the example in Figure 16.13 and function (16.18). Using first partial derivative expressions (16.19), the Hessian at $\mathbf{x}^{(1)}$ computes as

$$\nabla^2 f(\mathbf{x}^{(1)}) = \nabla^2 f(3, 5) = \begin{pmatrix} 12(x_1)^2 - 24x_1 & 0 \\ 0 & 6 \end{pmatrix} = \begin{pmatrix} 24 & 0 \\ 0 & 6 \end{pmatrix}$$

The matrix is positive definite because

$$\Delta \mathbf{x} \begin{pmatrix} 24 & 0 \\ 0 & 6 \end{pmatrix} \Delta \mathbf{x} = 24(\Delta x_1)^2 + 6(\Delta x_2)^2 > 0 \quad \text{for all } \Delta \mathbf{x} \neq \mathbf{0}$$

Confirming principal [16.21](#), local minimum $\mathbf{x}^{(1)}$ has a positive definite, and thus positive semidefinite Hessian. Conversely, we can establish that stationary point $\mathbf{x}^{(1)}$ is a local minimum by applying principle [16.22](#) with the Hessian positive definite.

The second stationary point $\mathbf{x}^{(2)} = (0, 5)$ shows that properties [16.21](#) and [16.22](#) are not always conclusive. There

$$\nabla^2 f(\mathbf{x}^{(2)}) = \nabla^2 f(0, 5) = \begin{pmatrix} 12(x_1)^2 - 24x_1 & 0 \\ 0 & 6 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 6 \end{pmatrix}$$

and quadratic form

$$\Delta \mathbf{x} \begin{pmatrix} 0 & 0 \\ 0 & 6 \end{pmatrix} \Delta \mathbf{x} = 6(\Delta x_2)^2 \geq 0$$

We can apply principle [16.21](#) to rule out the possibility of a local maximum because this Hessian is not negative semidefinite. Still, principle [16.22](#) cannot be applied to assure a local minimum with the Hessian only positive semidefinite. Without extending to third derivatives, we cannot distinguish between a local minimum and a saddle point.

EXAMPLE 16.10: VERIFYING LOCAL OPTIMA

Verify that function

$$f(x_1, x_2, x_3) \triangleq (x_1)^2 + x_1 x_2 + 5(x_2)^2 + 9(x_3 - 2)^2$$

has a local minimum at $\mathbf{x} = (0, 0, 2)$.

Solution: We apply sufficient conditions [16.22](#). First, \mathbf{x} must be a stationary point. All three partial derivatives

$$\frac{\partial f}{\partial x_1} = 2x_1 + x_2, \quad \frac{\partial f}{\partial x_2} = x_1 + 10x_2, \quad \frac{\partial f}{\partial x_3} = 18(x_3 - 2)$$

= 0 at $\mathbf{x} = (0, 0, 2)$ as required.

Next we consider the Hessian

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 10 & 0 \\ 0 & 0 & 18 \end{pmatrix}$$

We may verify that this matrix is positive definite, and thus \mathbf{x} a local minimum, by checking that all determinants of principal submatrices are positive (refer to Primer 8 if needed):

$$\det(2) = 2 > 0, \quad \det \begin{pmatrix} 2 & 1 \\ 1 & 10 \end{pmatrix} = 18 > 0, \quad \det \begin{pmatrix} 2 & 1 & 0 \\ 1 & 10 & 0 \\ 0 & 0 & 18 \end{pmatrix} = 324 > 0$$

EXAMPLE 16.11: VERIFYING SADDLE POINTS

Verify that function

$$f(x_1, x_2) \triangleq (x_1)^2 - 2x_1 - (x_2)^2$$

has a saddle point at $\mathbf{x} = (1, 0)$.

Solution: To fulfill definition [16.20](#), a saddle point must first be a stationary point. Checking yields

$$\frac{\partial f}{\partial x_1} = 2x_1 - 2 = 2(1) - 2 = 0 \quad \text{and} \quad \frac{\partial f}{\partial x_2} = -2x_2 = -2(0) = 0$$

Now computing the Hessian gives

$$\nabla^2 f(1, 0) = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

With first principal determinant = 2, and second = -4, this matrix is neither positive semidefinite nor negative semidefinite. Thus \mathbf{x} violates requirements [16.21](#) for both a local minimum and a local maximum. The remaining possibility is a saddle point.

16.4 CONVEX/CONCAVE FUNCTIONS AND GLOBAL OPTIMALITY

Improving search Algorithm 3A (Section 3.2), which provides the paradigm for nearly all unconstrained nonlinear programming algorithms, stops if it encounters a locally optimal solution (principle [3.6](#)). What then? We would clearly prefer an overall, or **global optimum**.

In this section we investigate objective functions having special **convex**, **concave**, and **unimodal** forms that allow us to prove that a local optimum must also be global (see also Section 3.4). With other objectives we must either accept the improving search stopping point or try for another by restarting the search from a different initial $\mathbf{x}^{(0)}$.

Convex and Concave Functions Defined

Convex and concave functions can be defined in terms of how $f(\mathbf{x})$ changes as we move from $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(2)}$ along straight line path $\Delta\mathbf{x} \triangleq (\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$. (See also Section 3.4)

Definition 16.23 | A function $f(\mathbf{x})$ is **convex** if

$$f(\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})) \leq f(\mathbf{x}^{(1)}) + \lambda(f(\mathbf{x}^{(2)}) - f(\mathbf{x}^{(1)}))$$

for every $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ in its domain and every step $\lambda \in [0, 1]$. Similarly, $f(\mathbf{x})$ is **concave** if

$$f(\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})) \geq f(\mathbf{x}^{(1)}) + \lambda(f(\mathbf{x}^{(2)}) - f(\mathbf{x}^{(1)}))$$

for all $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$ and $\lambda \in [0, 1]$.

Interpolation of f values along the line segment from $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(2)}$ should neither underestimate for a convex function nor overestimate for a concave one.

Figure 16.15 illustrates for functions of 2-vectors $\mathbf{x} \triangleq (x_1, x_2)$. The indicated moves start at $\mathbf{x}^{(1)}$ and advance toward $\mathbf{x}^{(2)}$ along direction $(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$. Each point

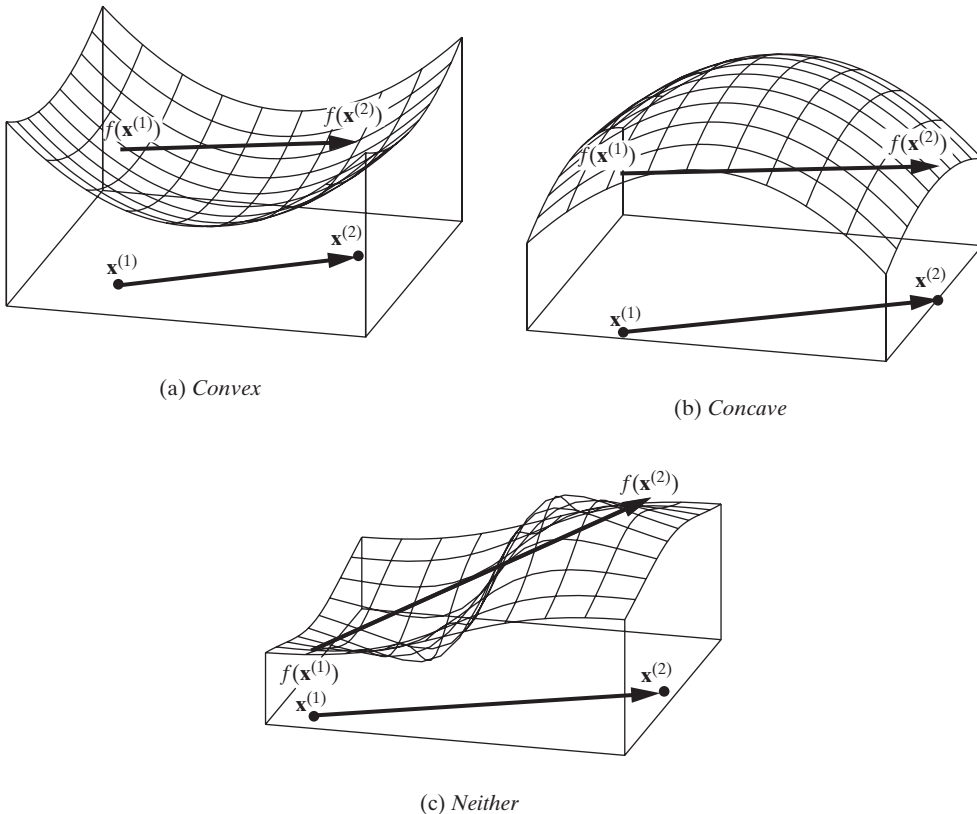


FIGURE 16.15 Convex and Concave Functions

\mathbf{x} in that trajectory has a representation

$$\mathbf{x} = \mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$$

for some $\lambda \in [0, 1]$ (property 3.31). For example, $\mathbf{x}^{(1)}$ corresponds to $\lambda = 0$, and $\mathbf{x}^{(2)}$ to $\lambda = 1$.

The issue in definition 16.23 is what happens when we interpolate an estimated value for f somewhere along the trajectory. For convex functions [Figure 16.15(a)] the corresponding interpolated values

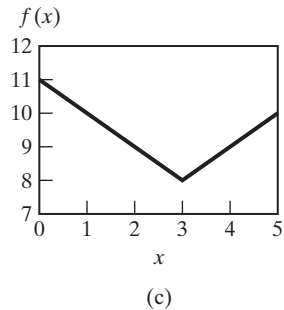
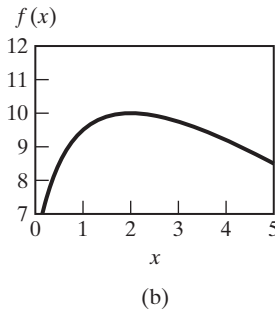
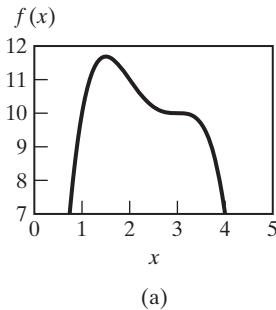
$$f(\mathbf{x}^{(1)}) + \lambda(f(\mathbf{x}^{(2)}) - f(\mathbf{x}^{(1)}))$$

should always equal or exceed the true $f(\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}))$. For concave functions [part (b)] it should fall equal or below.

The property must hold for every pair of points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. For example, some pairs would meet the test for convexity in Figure 16.15(c), and others would satisfy the definition of concave. Still, the function is neither convex nor concave because the indicated pair violates both definitions.

EXAMPLE 16.12: RECOGNIZING CONVEX AND CONCAVE FUNCTIONS

Determine graphically whether each of the following single-variable functions is convex, concave, or neither over $x \in [0, 5]$.



Solution: We apply definition 16.23.

(a) This function is neither convex nor concave. To demonstrate that it is not convex, take $x^{(1)} = 1, x^{(2)} = 2$, and $\lambda = \frac{1}{2}$.

$$\begin{aligned} f(x^{(1)} + \lambda(x^{(2)} - x^{(1)})) &= f(1 + \frac{1}{2}(2 - 1)) = f(1.5) \approx 11.7 \\ &\neq f(x^{(1)}) + \lambda(f(x^{(2)}) - f(x^{(1)})) = 10 + \frac{1}{2}(11 - 10) = 10.5 \end{aligned}$$

Similarly choosing $x^{(1)} = 3, x^{(2)} = 2$ and $\lambda = \frac{1}{2}$ establishes that the function is not concave because

$$\begin{aligned} f(x^{(1)} + \lambda(x^{(2)} - x^{(1)})) &= f(3 + \frac{1}{2}(2 - 3)) = f(2.5) \approx 10.2 \\ &\neq f(x^{(1)}) + \lambda(f(x^{(2)}) - f(x^{(1)})) = 10 + \frac{1}{2}(11 - 10) = 10.5 \end{aligned}$$

- (b) This function is apparently concave because definition 16.23 holds for all pairs of points displayed.
- (c) This function is apparently convex because definition 16.23 holds for all pairs of points displayed. Notice that convex (and concave) functions need not be differentiable.

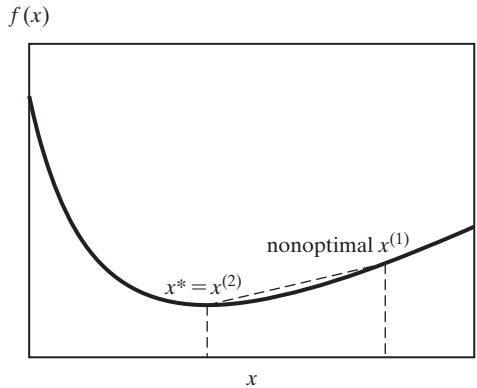
Sufficient Conditions for Unconstrained Global Optima

The importance of convex and concave objective functions lies with their unusual tractability for improving search.

Principle 16.24 If $f(\mathbf{x})$ is a convex function, every unconstrained local minimum of f is an unconstrained global minimum. If $f(\mathbf{x})$ is concave, every unconstrained local maximum is an unconstrained global maximum.

That is, a search must only achieve a local optimum to produce with no additional effort a global minimum of a convex objective function or a global maximum of a concave one.

To see why principle 16.24 must be true, consider a convex objective function $f(x)$, a global minimum x^* , and any $x^{(1)}$ that is not globally optimal:



Then

$$f(x^*) < f(x^{(1)}) \quad \text{or} \quad \lambda(f(x^*) - f(x^{(1)})) < 0 \tag{16.23}$$

for all $\lambda > 0$. Combining with the convexity definition 16.23 yields

$$f(x^{(1)} + \lambda(x^* - x^{(1)})) \leq f(x^{(1)}) + \lambda(f(x^*) - f(x^{(1)})) < f(x^{(1)}) \tag{16.24}$$

for all $\lambda \in (0, 1]$. That is, direction $\Delta x = x^* - x^{(1)}$ is an improving direction at every $x^{(1)}$ that is not globally optimal. A local optimum, which permits no improving directions, can exist only if it is also a global optimum.

Convex/Concave Functions and Stationary Points

Principle 16.24 shows that we need only compute a local minimum of a convex function or a local maximum of a concave one to obtain an unconstrained global optimum. In fact, the requirement is even weaker when the objective function is differentiable.

Principle 16.25 | Every stationary point of a smooth convex function is an unconstrained global minimum, and every stationary point of a smooth concave function is an unconstrained global maximum.

We require only an \mathbf{x} with $\nabla f(\mathbf{x}) = \mathbf{0}$.

For an idea of why principle 16.25 holds, let f be a smooth convex function, and $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Convexity definition 16.23 assures that

$$f(\mathbf{x}^* + \lambda(\mathbf{x} - \mathbf{x}^*)) \leq f(\mathbf{x}^*) + \lambda(f(\mathbf{x}) - f(\mathbf{x}^*))$$

for any \mathbf{x} and any $\lambda \in (0, 1]$. Furthermore, first-order Taylor approximation 16.16 gives

$$f(\mathbf{x}^* + \lambda(\mathbf{x} - \mathbf{x}^*)) \approx f(\mathbf{x}^*) + \lambda \nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)$$

Subtracting, simplifying, and dividing by $\lambda \rightarrow 0$ yields

$$f(\mathbf{x}) - f(\mathbf{x}^*) \geq \nabla f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)$$

It follows that \mathbf{x}^* is a global minimum when $\nabla f(\mathbf{x}^*) = \mathbf{0}$ because $f(\mathbf{x}) - f(\mathbf{x}^*) \geq 0$ for all \mathbf{x} .

EXAMPLE 16.13: VERIFYING GLOBAL OPTIMA WITH CONVEXITY

The function

$$f(x) \triangleq 20 - x^2 + 6x$$

is concave. Use this fact to establish that it has an unconstrained global maximum at $x = 3$.

Solution: Differentiating yields

$$f'(x) = -2x + 6$$

so that $f'(3) = 0$. Being a stationary point of a concave function, $x = 3$ must be an unconstrained global maximum (principle 16.25).

Tests for Convex and Concave Functions

Many familiar functions are either convex or concave, but it is often tedious to verify definitions 16.23. Fortunately, when the function's domain is all real n -vectors or all positive n -vectors, or any other open convex set (definition 3.27), some important properties are available to simplify the analysis.

Principle 16.26 | If $f(\mathbf{x})$ is convex, $-f(\mathbf{x})$ is concave, and vice versa.

Principle 16.27 An $f(\mathbf{x})$ with continuous second (partial) derivatives is convex if and only if Hessian $\nabla^2 f(\mathbf{x})$ is positive semidefinite at all \mathbf{x} in its (open convex set) domain. It is concave if and only if $\nabla^2 f(\mathbf{x})$ is negative semidefinite at all \mathbf{x} in the domain.

Principle 16.28 Linear functions are both convex and concave.

Principle 16.29 Any $f(\mathbf{x})$ formed as the nonnegative-weighted ($\alpha_i \geq 0$) sum

$$f(\mathbf{x}) \triangleq \sum_{i=1}^k \alpha_i g_i(\mathbf{x})$$

of convex functions $g_i(\mathbf{x}), i = 1, \dots, k$, is itself convex. The nonnegative-weighted sum of concave functions is concave.

Principle 16.30 Any $f(\mathbf{x})$ formed as the maximum

$$f(\mathbf{x}) \triangleq \max\{g_i(\mathbf{x}) : i = 1, \dots, k\}$$

of convex functions $g_i(\mathbf{x}), i = 1, \dots, k$, is itself convex. The minimum of concave functions is concave.

Principle 16.31 If $g(y)$ is a nondecreasing, single-variable convex function, and $h(\mathbf{x})$ is convex, $f(\mathbf{x}) \triangleq g(h(\mathbf{x}))$ is convex. If $g(y)$ is a nondecreasing, single-variable concave function, and $h(\mathbf{x})$ is concave, $f(\mathbf{x}) \triangleq g(h(\mathbf{x}))$ is concave.

Principle 16.32 If $g(\mathbf{x})$ is a concave function, $f(\mathbf{x}) \triangleq 1/g(\mathbf{x})$ is convex over \mathbf{x} with $g(\mathbf{x}) > 0$. If $g(\mathbf{x})$ is a convex function, $f(\mathbf{x}) \triangleq 1/g(\mathbf{x})$ is concave over \mathbf{x} with $g(\mathbf{x}) < 0$.

To see the power of principles [16.26](#) to [16.32](#), examine the curve-fitting objective in linear regression form (16.4):

$$\min f(x_1, x_2) \triangleq \sum_{i=1}^m [q_i - (x_1 + x_2 p_i)]^2$$

[The nonlinear case with $(q_i - x_1(p_i)^{x_2})$ is not convex.] Recall that the p_i and q_i are given constants.

To show this linear regression f is convex, notice first that it is the (unweighted) sum of functions

$$g_i(x_1, x_2) \triangleq [q_i - (x_1 + x_2 p_i)]^2$$

Under principle [16.29](#), f will be convex if each of the g_i is convex.

Now, dropping the i subscripts, we examine

$$\begin{aligned} g(x_1, x_2) &\triangleq [q - (x_1 + x_2 p)]^2 \\ &= [|q - (x_1 + x_2 p)|]^2 \\ &= [\max\{(q - x_1 - x_2 p), -(q - x_1 - x_2 p)\}]^2 \end{aligned}$$

(the last equality holds because $|z| = \max\{z, -z\}$). Expressions $(q - x_1 - x_2p)$ and $-(q - x_1 - x_2p)$ are both linear and thus convex by principle [16.28]. Therefore,

$$h(x_1, x_2) \triangleq |q - (x_1 + x_2p)| = \max\{q - x_1 - x_2p, -(q - x_1 - x_2p)\}$$

is also convex; it is the maximum of convex functions (principle [16.30]). Finally, consider $s(y) \triangleq y^2$. Second derivative $s''(y) = 2$ proves $s(y)$ is convex because $s''(y)$ is the 1 by 1 Hessian matrix and positive definite (principle [16.27]). Over domain $y \geq 0$, $s(y) \triangleq y^2$ is also nondecreasing. Thus we may apply composition principle [16.31] to conclude that

$$g(x_1, x_2) \triangleq (q - (x_1 + x_2p))^2 = s(h(x_1, x_2))$$

is convex. This completes the argument for convexity f .

EXAMPLE 16.14: VERIFYING CONVEXITY AND CONCAVITY

Apply principles [16.26] to [16.32] to establish that the first two of the following functions are convex and the last two are concave over the specified domains.

- (a) $f(x_1, x_2) \triangleq (x_1 + 1)^4 + x_1x_2 + (x_2 + 1)^4$ over all $x_1, x_2 > 0$
 (b) $f(x_1, x_2) \triangleq e^{-3x_1+x_2}$ over all x_1, x_2
 (c) $f(x_1, x_2, x_3) \triangleq -4(x_1)^2 + 5x_1x_2 - 2(x_2)^2 + 18x_3$ over all x_1, x_2
 (d) $f(x_1, x_2) \triangleq \frac{1}{-7x_1} - e^{-3x_1+x_2}$ over all $x_1, x_2 > 0$

Solution:

(a) Here the Hessian matrix is

$$\nabla^2 f(x_1, x_2) = \begin{pmatrix} 12(x_1 + 1)^2 & 1 \\ 1 & 12(x_2 + 1)^2 \end{pmatrix}$$

Determinants of its principal submatrices are $12(x_1 + 1)^2$ and $144(x_1 + 1)^2(x_2 + 1)^2 - 1$, which are both positive for all $x_1, x_2 > 0$. Thus the Hessian is positive definite, and f is convex by principle [16.27].

(b) Function $h(x_1, x_2) \triangleq -3x_1 + x_2$ is convex because it is linear (principle [16.28]). Also, $g(y) \triangleq e^y$ is nondecreasing and convex because $g''(y) = e^y > 0$. Thus composition principle [16.31] proves that $f(x_1, x_2) = g(h(x_1, x_2))$ is convex.

(c) For this function the Hessian matrix is

$$\nabla^2 f(x_1, x_2, x_3) = \begin{pmatrix} -8 & 5 & 0 \\ 5 & -4 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Principal submatrix determinants are -8 , $(32 - 25) = 7$, and 0 , which imply that the Hessian is negative semidefinite and f is concave (principle [16.27]).

(d) Over $x_1, x_2 > 0$, first term $g_1(x_1, x_2) \triangleq 1/(-7x_1)$ is the reciprocal of negative-valued, linear, and thus convex function $h(x_1, x_2) \triangleq -7x_1$. It follows that

this $g_1(x_1, x_2)$ is concave (principle [16.32](#)). Part (b) already established that $g_2(x_1, x_2) \triangleq e^{-3x_1+x_2}$ is convex, meaning (principle [16.26](#)) that its negative is concave. Thus f is the sum of concave functions and so concave (principle [16.29](#)).

Unimodal versus Convex/Concave Objectives

In Section 16.2 we introduced the notion of unimodal objective functions (definition [16.9](#)). Every unconstrained local optimum of a unimodal objective function is a global optimum because improving directions exist at every point that can be bettered.

Since both unimodal and convex/concave objective functions imply that unconstrained local optima are global it should be no surprise that there is a connection.

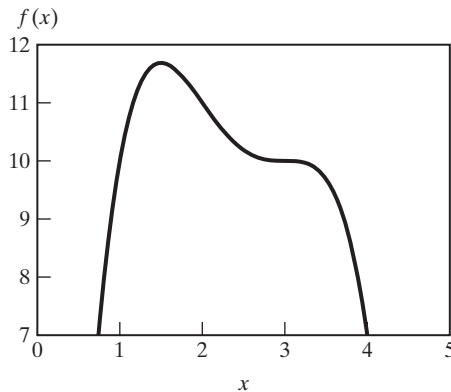
Principle 16.33 Both convex objective functions in minimize problems and concave objective functions in maximize problems are unimodal.

Expressions (16.23) and (16.24) have already shown why. Improving directions exist at all solutions not globally optimal in a convex minimization or concave maximization.

Unimodality is a weaker requirement than convexity or concavity.

Principle 16.34 A unimodal objective function need not be either convex or concave.

For example, the following is unimodal for a maximize problem, but we showed in Example 16.12(a) that it is not concave.



Other such examples are our Custom Computer objective in Figure 16.3 and PERT application in Figure 16.5.

Unfortunately, convenient combination rules such as [16.26] – [16.32] do not generally hold for arbitrary unimodal objectives. Thus in practice we must often establish the more restrictive convex or concave properties to be sure that an objective is unimodal. When the functions are not concave for a maximize or convex for a minimize, which often happens in applied models, we usually must accept the risk of local optima that are not global.

16.5 GRADIENT SEARCH

In Section 3.3, principle [3.23] established that a nonzero gradient $\nabla f(\mathbf{x}^{(t)})$ provides an improving direction at solution $\mathbf{x}^{(t)}$. First-order Taylor series computations of expression (16.21) guarantee improvement with sufficiently small steps in direction $\Delta \mathbf{x} = \nabla f(\mathbf{x}^{(t)})$ for a maximize problem or $\Delta \mathbf{x} = -\nabla f(\mathbf{x}^{(t)})$ for a minimize. In this section we develop the simple **gradient search** algorithm that adopts such gradient-based move directions.

Gradient Search Algorithm

Algorithm 16D provides details. The move direction for each iteration is derived from the gradient at the current point.

Definition 16.35 At any current point $\mathbf{x}^{(t)}$ with gradient $\nabla f(\mathbf{x}^{(t)}) \neq \mathbf{0}$, gradient search pursues move direction

$$\Delta \mathbf{x} \triangleq \pm \nabla f(\mathbf{x}^{(t)})$$

(+ for a maximize, – for a minimize).

ALGORITHM 16D: GRADIENT SEARCH

Step 0: Initialization. Choose any starting solution $\mathbf{x}^{(0)}$, pick stopping tolerance $\epsilon > 0$, and set solution index $t \leftarrow 0$.

Step 1: Gradient. Compute objective function gradient $\nabla f(\mathbf{x}^{(t)})$ at current point $\mathbf{x}^{(t)}$.

Step 2: Stationary Point. If gradient norm $\|\nabla f(\mathbf{x}^{(t)})\| < \epsilon$, stop. Point $\mathbf{x}^{(t)}$ is sufficiently close to a stationary point.

Step 3: Direction. Choose gradient move direction

$$\Delta \mathbf{x}^{(t+1)} \leftarrow \pm \nabla f(\mathbf{x}^{(t)})$$

(+ for maximize and – for minimize).

Step 4: Line Search. Solve (at least approximately) corresponding one-dimensional line search

$$\max \text{ or } \min f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)})$$

to compute λ_{t+1} .

Step 5: New Point. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda_{t+1} \Delta \mathbf{x}^{(t+1)}$$

Step 6: Advance. Increment $t \leftarrow t + 1$, and return to Step 1.

Gradient norm

$$||\nabla f(\mathbf{x}^{(t)})|| \triangleq \sqrt{\sum_j \left(\frac{\partial f}{\partial x_j}\right)^2}$$

provides a stopping rule at Step 1 of Algorithm 16D. If the gradient at $\mathbf{x}^{(t)}$ is very small in length (less than stopping tolerance ϵ), all its components must be nearly zero. Thus the search has essentially reached a stationary point (definition [16.18](#)).

Of course, we know from Section 16.3 that a stationary point may be a saddle point (definition [16.20](#)), not the local optimum we seek. Still, a pure gradient algorithm can guarantee no more. When $\nabla f(\mathbf{x}^{(t)}) = \mathbf{0}$, principle [16.35](#) provides no move direction to pursue.

Gradient Search of Custom Computer Application

Table 16.5 details application of Algorithm 16D to the Custom Computer regression model (16.7) (Section 16.1). Figure 16.16 plots the first few steps on a contour map.

TABLE 16.5 Gradient Search of Custom Computer Application

t	$\mathbf{x}^{(t)}$	$f(\mathbf{x}^{(t)})$	$\nabla f(\mathbf{x}^{(t)})$	$ \nabla f(\mathbf{x}^{(t)}) $	λ_{t+1}
0	(32.00, -0.4000)	174.746	(-6.24, 1053.37)	1053.39	0.00007
1	(32.00, -0.4687)	141.138	(-23.06, -0.14)	23.06	0.10558
2	(34.44, -0.4540)	112.599	(-4.57, 759.60)	759.61	0.00007
3	(34.44, -0.5078)	93.297	(-16.28, -0.10)	16.28	0.11303
4	(36.28, -0.4962)	78.123	(-3.34, 530.01)	530.02	0.00008
5	(36.28, -0.5365)	67.897	(-11.34, -0.07)	11.34	0.11970
6	(37.63, -0.5278)	60.133	(-2.33, 365.74)	365.75	0.00008
7	(37.63, -0.5571)	54.932	(-7.78, -0.05)	7.78	0.12905
8	(38.64, -0.5512)	51.006	(-1.48, 251.17)	251.17	0.00008
9	(38.64, -0.5722)	48.428	(-5.19, -0.03)	5.19	0.13926
10	(39.36, -0.5684)	46.548	(0.88, 168.22)	168.22	0.00009
11	(39.36, -0.5829)	45.348	(-3.34, -0.02)	3.34	0.14737
12	(39.85, -0.5806)	44.522	(-0.51, 108.66)	108.66	0.00009
13	(39.85, -0.5902)	44.007	(-2.10, -0.01)	2.10	0.15220
14	(40.17, -0.5888)	43.672	(-0.30, 68.07)	68.07	0.00009
15	(40.17, -0.5949)	43.466	(-1.29, 0.00)	1.29	0.16435
16	(40.38, -0.5941)	43.329	(-0.14, 42.34)	42.34	0.00009
17	(40.38, -0.5980)	43.248	(-0.76, 0.00)	0.76	0.15652
18	(40.50, -0.5975)	43.203	(-0.10, 24.60)	24.60	0.00009
19	(40.50, -0.5997)	43.175	(-0.46, 0.00)	0.46	0.14805
20	(40.57, -0.5994)	43.160	(-0.08, 14.77)	14.77	0.00009
21	(40.57, -0.6007)	43.150	(-0.29, 0.00)	0.29	0.15527
22	(40.62, -0.6005)	43.143	(-0.04, 9.37)	9.37	0.00009
23	(40.62, -0.6014)	43.139	(-0.18, 0.00)	0.18	0.16376
24	(40.65, -0.6013)	43.137	(-0.02, 5.78)	5.78	0.00009
25	(40.65, -0.6018)	43.135	(-0.11, 0.00)	0.11	0.15266
26	(40.66, -0.6017)	43.134	(-0.02, 3.37)	3.37	0.00009
27	(40.66, -0.6020)	43.134	(-0.06, 0.00)	0.06	Stop

The search begins at $\mathbf{x}^{(0)} = (32, -0.4)$. There the move direction (principle [16.35](#)) is

$$\Delta \mathbf{x} = -\nabla f(\mathbf{x}^{(0)}) = -(-6.24, 1053.37) = (6.24, -1053.37)$$

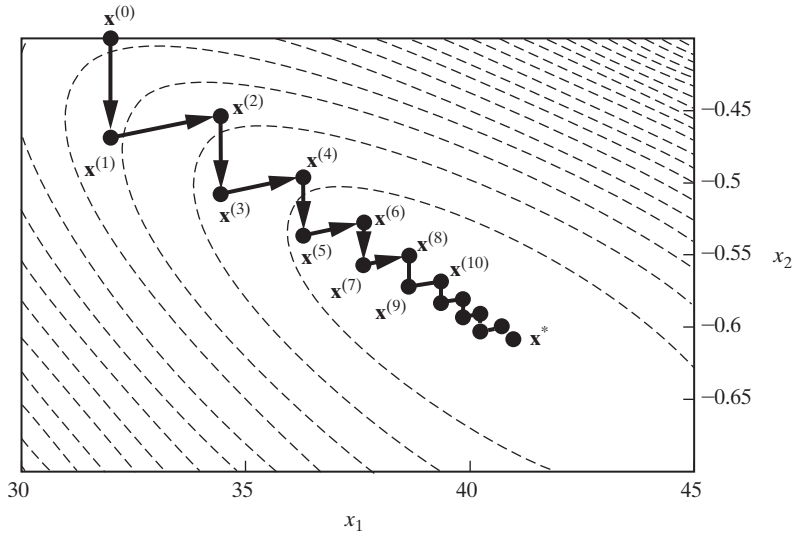
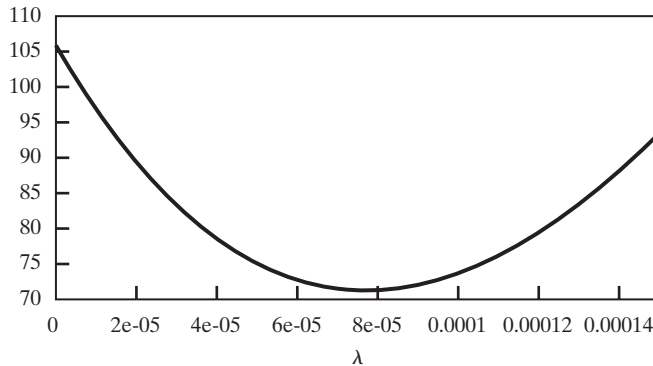


FIGURE 16.16 Gradient Search of Custom Computer Application

This move direction defines the first line search for step size λ . We identify the largest step for which $\Delta \mathbf{x}$ continues to improve by solving the 1-dimensional problem

$$\min f(\mathbf{x}^{(0)} + \lambda \Delta \mathbf{x}) \triangleq f(32 + 6.24\lambda, -0.4 - 1053.37\lambda)$$

Plotting shows that a minimum occurs at approximately $\lambda_1 = 0.00007$.



The result is new point

$$\mathbf{x}^{(1)} \leftarrow \mathbf{x}^{(0)} + \lambda_1 \Delta \mathbf{x} \approx (32, -0.4687)$$

Computations in Table 16.5 employ stopping tolerance $\epsilon = 0.1$. Thus the algorithm continues until norm $\|\nabla f(\mathbf{x}^{(t)})\| < 0.1$ at $t = 27$. The resulting (approximate) stationary point is $\mathbf{x}^{(27)} = (40.66, -0.6020)$ which we know from earlier analysis approximates a local (and here also global) minimum.

EXAMPLE 16.15: EXECUTING GRADIENT SEARCH

Consider the unconstrained nonlinear program

$$\max f(x_1, x_2) \triangleq \frac{x_1}{1 + e^{0.1x_1}} - (x_2 - 5)^2$$

(a) Compute the move direction that would be pursued by gradient search Algorithm 16D at $\mathbf{x}^{(0)} = (30, 2)$.

(b) State the line search problem implied by your direction of part (a).

Solution:

(a) At the specified $\mathbf{x}^{(0)}$, the gradient is

$$\nabla f(30, 2) = \begin{pmatrix} \frac{1 + e^{0.1x_1} - 0.1x_1e^{0.1x_1}}{(1 + e^{0.1x_1})^2} \\ -2(x_2 - 5) \end{pmatrix} = \begin{pmatrix} -0.088 \\ 6 \end{pmatrix}$$

Thus with a maximize problem we use direction

$$\Delta \mathbf{x} = +\nabla f(\mathbf{x}^{(0)}) = (-0.088, 6)$$

(b) The line search problem implied by the direction of part (a) is

$$\max f(30 - 0.088\lambda, 2 + 6\lambda) \triangleq \frac{(30 - 0.088\lambda)}{1 + e^{0.1(30 - 0.088\lambda)}} - [(2 + 6\lambda) - 5]^2$$

over $\lambda > 0$.

Steepest Ascent/Descent Property

Gradient search is sometimes called the **method of steepest ascent (steepest descent** for minimize problems) because the direction of principle [16.35](#) produces the most rapid rate of objective improvement near the current solution.

Principle 16.36 | At any $\mathbf{x}^{(t)}$ with $\nabla f(\mathbf{x}^{(t)}) \neq \mathbf{0}$, direction $\Delta \mathbf{x} \triangleq \nabla f(\mathbf{x}^{(t)})$ produces the locally steepest rate of objective function ascent, and $\Delta \mathbf{x} \triangleq -\nabla f(\mathbf{x}^{(t)})$ yields the locally steepest rate of descent.

The search of Figure 16.16 illustrates graphically. The rate of objective improvement at any point depends on the angle between the move direction and nearby objective function contours. Gradient-based directions, which move perpendicular to the contours (principle [3.20](#)), produce the most rapid local progress.

EXAMPLE 16.16: COMPUTING DIRECTIONS OF STEEPEST ASCENT/DESCENT

Return to the nonlinear program of Example 16.15 and compute the steepest descent direction of length 1 at point $\mathbf{x}^{(0)} = (30, 2)$.

Solution: In accord with principle [16.36](#), the steepest descent direction at (30, 2) will be the negative of the improving gradient direction computed in Example 16.15(a). Thus the steepest descent direction is $\Delta \mathbf{x} = (0.088, -6)$. Dividing by norm = 6.000645, gives length 1 direction (0.0147, 0.9999).

Zigzagging and Poor Convergence of Gradient Search

Gradient search is appealingly straightforward, but it is not very effective in most applications. To see why, look again at the search in Figure 16.16 and Table 16.5. The first move was in direction $\Delta \mathbf{x} = (6.24, -1053.37)$, the second pursued $\Delta \mathbf{x} = (23.06, 0.14)$, and the third adopted $\Delta \mathbf{x} = (4.57, -759.60)$. This third direction is almost exactly parallel to the first, and the fourth will parallel the second. Later iterations continued this **zigzagging** alternation of almost perpendicular move directions.

We would prefer to approach the optimum more directly, but these gradient-based directions still produced good progress in early iterations. The objective fell from 174.746 to 46.548 in 10 steps.

The difficulty arises later in the search. Near an optimal solution, the shape of the objective function changes rapidly with very small step sizes. Thus although gradient-based directions produce the locally steepest rate of improvement, they can be followed only a very short distance before the best direction changes dramatically. The resulting zigzagging consumed the last 17 iterations in Table 16.5 to reduce the objective from 46.548 to 43.134.

Unfortunately, this poor convergence is typical of gradient methods.

Principle 16.37 | Although gradient search may produce good initial progress, zigzagging as it approaches a stationary point makes the method too slow and unreliable to provide satisfactory results in many unconstrained nonlinear applications.

Zigzagging is not the only convergence problem with gradient search. With small solution changes having a big objective function impact, numerical errors also can hopelessly bog down the procedure far from an optimal solution. More sophistication is required to obtain a really satisfactory improving search algorithm.

16.6 NEWTON'S METHOD

Gradient search can be viewed as pursuing the move direction suggested by the first-order Taylor series approximation (definition [16.17](#))

$$f_1(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) \triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x}$$

Aligning $\Delta \mathbf{x}$ with gradient $\nabla f(\mathbf{x}^{(t)})$ produces the most rapid improvement in this first-order approximation to $f(\mathbf{x})$.

To improve on the slow, zigzagging progress characteristic of gradient search (principle [16.37](#)) requires more information. An obvious possibility is extending to the second-order Taylor approximation

$$f_2(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}) \triangleq f(\mathbf{x}^{(t)}) + \lambda \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} + \frac{\lambda^2}{2} \Delta \mathbf{x} \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x}$$

This section explores the famous **Newton's method**, which does exactly that.

Newton Step

Unlike the first-order Taylor approximation, which is linear in directional components Δx_j , the quadratic, second-order version may have a local maximum or minimum. To determine the $\lambda \Delta \mathbf{x}$ move that takes us to such a local optimum of the second-order approximation, we may fix $\lambda = 1$ and differentiate f_2 with respect to components of $\Delta \mathbf{x}$. With $\lambda = 1$ the scalar-notation form of f_2 is

$$f_2(\mathbf{x}^{(t)} + \Delta \mathbf{x}) \triangleq f(\mathbf{x}^{(t)}) + \sum_{i=1}^n \left(\frac{\partial f}{\partial x_i} \right) \Delta x_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right) \Delta x_i \Delta x_j$$

Then partial derivatives with respect to move components are

$$\frac{\partial f_2}{\partial \Delta x_i} = \left(\frac{\partial f}{\partial x_i} \right) + \sum_{j=1}^n \left(\frac{\partial^2 f}{\partial x_j \partial x_i} \right) \Delta x_j, \quad i = 1, \dots, n$$

or in matrix format,

$$\nabla f_2(\Delta \mathbf{x}) = \nabla f(\mathbf{x}^{(t)}) + \nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x}$$

Either way, setting $\nabla f_2(\Delta \mathbf{x}) = 0$ to find a stationary point produces the famous **Newton step**.

Definition 16.38 | Newton steps $\Delta \mathbf{x}$, which move to a stationary point (if there is one), of the second-order Taylor series approximation to $f(\mathbf{x})$ at current point $\mathbf{x}^{(t)}$ are obtained by solving the linear equation system

$$\nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} = -\nabla f(\mathbf{x}^{(t)})$$

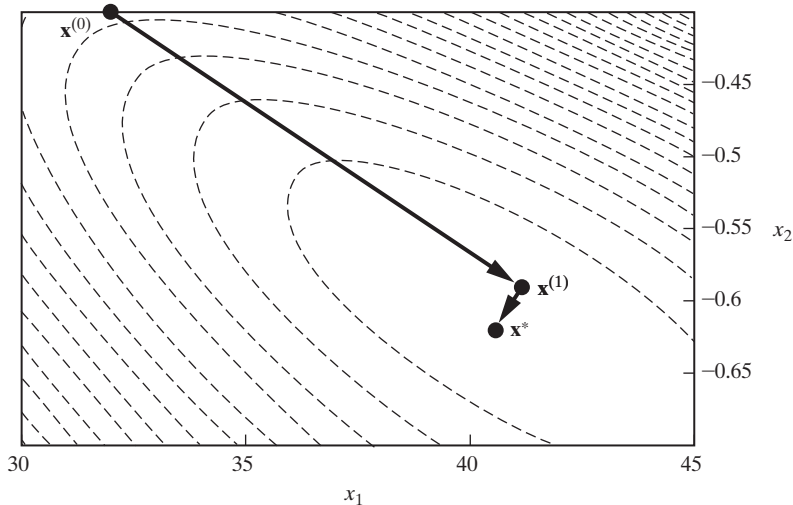


FIGURE 16.17 Newton's Method on the Custom Computer Application

Figure 16.17 illustrates for our Custom Computer curve-fitting model (16.7) (Section 16.1). First and second partial derivatives at initial point $\mathbf{x}^{(0)} = (32, -0.4)$ are

$$\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} -6.240 \\ 1053 \end{pmatrix} \quad \text{and} \quad \nabla^2 f(\mathbf{x}^{(0)}) = \begin{pmatrix} 7.13 & 293.99 \\ 293.99 & 18.817 \end{pmatrix}$$

Solving system

$$\begin{pmatrix} 7.13 & 293.99 \\ 293.99 & 18.817 \end{pmatrix} \Delta \mathbf{x} = - \begin{pmatrix} -6.240 \\ 1053 \end{pmatrix}$$

produces Newton step $\Delta \mathbf{x} = (8.956, -0.1959)$, which takes us to second-order Taylor approximation minimum

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta \mathbf{x} = (32, -0.4) + (8.956, -0.1959) = (40.96, -0.5959) \quad (16.25)$$

EXAMPLE 16.17: COMPUTING NEWTON STEPS

Compute the Newton step corresponding to current point $\mathbf{x}^{(0)} = (0, 1)$ in a search of unconstrained NLP

$$\min f(x_1, x_2) \triangleq (x_1 + 1)^4 + x_1 x_2 + (x_2 + 1)^4$$

Solution: To develop linear system [16.38](#), we compute partial derivatives

$$\nabla f(0, 1) = \begin{pmatrix} 4(x_1 + 1)^3 + x_2 \\ x_1 + 4(x_2 + 1)^3 \end{pmatrix} = \begin{pmatrix} 5 \\ 32 \end{pmatrix}$$

and

$$\nabla^2 f(0, 1) = \begin{pmatrix} 12(x_1 + 1)^2 & 1 \\ 1 & 12(x_2 + 1)^2 \end{pmatrix} = \begin{pmatrix} 12 & 1 \\ 1 & 48 \end{pmatrix}$$

Then Newton step $\Delta \mathbf{x}$ is the solution to the system

$$\begin{pmatrix} 12 & 1 \\ 1 & 48 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = - \begin{pmatrix} 5 \\ 32 \end{pmatrix}$$

which is approximately $\Delta x_1 = -0.3617$, $\Delta x_2 = -0.6591$.

Newton's Method

Newton's method proceeds by repeating the process above. That is, it uses first and second partial derivatives at the current point to compute a Newton step, updates the solution with that step, and repeats the process. Algorithm 16E provides details.

Newton's Method on the Custom Computer Application

Table 16.6 and Figure 16.17 apply Algorithm 16E to our Custom Computer application using stopping tolerance $\epsilon = 0.1$. Equation (16.25) already derived the first move to $\mathbf{x}^{(1)} = (40.96, -0.5959)$. The derivatives are recomputed, and new Newton step $\Delta \mathbf{x} = (-0.2733, -0.0061)$ brings us to

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \Delta \mathbf{x} = (40.96, -0.5959) + (-0.2733, -0.0061) = (40.68, -0.6020)$$

Notice that a step size of $\lambda = 1$ is assumed because the Newton step represents a full move rather than just a direction.

Algorithm 16E stops after one additional move to $\mathbf{x}^{(3)} = (40.68, -0.6024)$. The gradient norm 0.00272 at that point is less than stopping tolerance $\epsilon = 0.1$, which implies that we are sufficiently close to a stationary point of the full objective function.

ALGORITHM 16E: NEWTON'S METHOD

Step 0: Initialization. Choose any starting solution $\mathbf{x}^{(0)}$, pick stopping tolerance $\epsilon > 0$, and set solution index $t \leftarrow 0$.

Step 1: Derivatives. Compute objective function gradient $\nabla f(\mathbf{x}^{(t)})$ and Hessian matrix $\nabla^2 f(\mathbf{x}^{(t)})$ at current point $\mathbf{x}^{(t)}$.

Step 2: Stationary Point. If $\|\nabla f(\mathbf{x}^{(t)})\| < \epsilon$, stop. Point $\mathbf{x}^{(t)}$ is sufficiently close to a stationary point.

Step 3: Newton Move. Solve the linear system

$$\nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} = -\nabla f(\mathbf{x}^{(t)})$$

for Newton move $\Delta \mathbf{x}^{(t+1)}$.

Step 4: New Point. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta \mathbf{x}^{(t+1)}$$

Step 5: Advance. Increment $t \leftarrow t + 1$, and return to Step 1.

TABLE 16.6 Newton's Method on the Custom Computer Application

t	$\mathbf{x}^{(t)}$	$f(\mathbf{x}^{(t)})$	$\nabla f(\mathbf{x}^{(t)})$	$\nabla^2 f(\mathbf{x}^{(t)})$	$\ \nabla f(\mathbf{x}^{(t)})\ $	$\Delta \mathbf{x}^{(t+1)}$
0	(32.00, -0.4000)	174.746	(-6.240, 1053)	$\begin{pmatrix} 7.13 & 293.99 \\ 293.99 & 18.817 \end{pmatrix}$	1053.4	(8.956, -0.1959)
1	(40.96, -0.5959)	43.820	(2.347, 116.3)	$\begin{pmatrix} 4.86 & 18.817 \\ 166.21 & 166.21 \end{pmatrix}$	116.36	(-0.2733, -0.0061)
2	(40.68, -0.6020)	43.133	(0.0289, 2.989)	$\begin{pmatrix} 4.81 & 158.97 \\ 158.97 & 10.918 \end{pmatrix}$	2.9895	(0.0058, -0.0004)
3	(40.69, -0.6024)	43.133	(0.0000, 0.0027)	$\begin{pmatrix} 4.81 & 158.73 \\ 158.73 & 10.899 \end{pmatrix}$	0.00272	Stop

EXAMPLE 16.18: EXECUTING NEWTON'S METHOD

Return to the model of Example 16.17, and execute two iterations of Newton's method Algorithm 16E starting with $\mathbf{x}^{(0)} = (0, 1)$.

Solution: Example 16.17 already computed partial derivative expressions

$$\nabla f(x_1, x_2) = \begin{pmatrix} 4(x_1 + 1)^3 + x_2 \\ x_1 + 4(x_2 + 1)^3 \end{pmatrix}$$

and

$$\nabla^2 f(x_1, x_2) = \begin{pmatrix} 12(x_1 + 1)^2 & 1 \\ 1 & 12(x_2 + 1)^2 \end{pmatrix}$$

along with first Newton step $\Delta \mathbf{x}^{(1)} = (-0.3617, -0.6591)$. Thus the first iteration produces

$$\mathbf{x}^{(1)} = (0.1) + (-0.3617, -0.6591) = (-0.3617, 0.3409)$$

Notice that no step size λ is applied (or equivalently, $\lambda = 1$).

Substituting this $\mathbf{x}^{(1)}$ in gradient and Hessian expressions produces the next 16.38 linear system

$$\begin{pmatrix} 4.889 & 1 \\ 1 & 28.93 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = - \begin{pmatrix} 1.381 \\ 9.282 \end{pmatrix}$$

There the solution is $\Delta \mathbf{x}^{(2)} = (-0.2184, -0.3133)$, and we complete the second iteration with

$$\mathbf{x}^{(2)} = (-0.3167, -0.6591) + (-0.2184, -0.3133) = (-0.5801, 0.0276)$$

Rapid Convergence Rate of Newton's Method

Comparison of Tables 16.5 and 16.6 shows dramatically improved convergence with Newton's method versus gradient search. The gradient algorithm required 27 moves to reach an optimum. Newton's method took only three.

Although the mathematical theory to fully explain this gain is beyond the scope of this book, it is typical of comparative experience with the methods.

Principle 16.39 If Newton's method converges to a local optimum, it usually does so in many fewer steps than first-order procedures such as gradient search.

Computational Trade-offs between Gradient and Newton Search

Of course, the number of iterations is not the only consideration in comparing the efficiency of algorithms. We must also take into account the effort per iteration.

There Newton's method has both advantages and disadvantages. On the positive side is the absence of line searches (although some extensions add them to

Algorithm 16E). Once each direction $\Delta \mathbf{x}^{(t+1)}$ is computed, we may update immediately, with no need for a relatively costly search for the best step size λ .

The extra burdens of Newton's method come with its use of the second-order Taylor approximation. Each directional computation on an n -vector $\mathbf{x}^{(t)}$ requires evaluating n expressions for the various first partial derivatives, and (using symmetry) another $\frac{1}{2}n(n+1)$ expressions for the Hessian. This is roughly the same amount of work as evaluating the original objective $n + \frac{1}{2}n(n+1)$ times versus only n for gradient search. In addition, we must solve an n by n system of linear equations to find the next Newton move. This too represents a substantial computational burden at every iteration.

Principle 16.40 Computing both first and second partial derivatives plus solving a linear system of equations at each iteration makes Newton's method computationally burdensome as the dimension of the decision vector becomes large.

Starting Close with Newton's Method

Perhaps the greatest disadvantage of Newton's method is that it may not converge at all.

Principle 16.41 Newton's method is assured of converging to local optimum only if it starts relatively close to a local optimum.

There are two main reasons convergence can fail. First is the quadratic Taylor approximation itself. Far from an optimal solution, the second-order approximation can give such poor information that the computed Newton step does not even improve the objective function. For instance, suppose that the Newton search of Example 16.18 had begun at $\mathbf{x}^{(0)} = (-1, 1)$ instead of $(0, 1)$. Then the linear system of definition [16.38](#) would have been

$$\begin{pmatrix} 0 & 1 \\ 1 & 48 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix} = - \begin{pmatrix} 1 \\ 31 \end{pmatrix}$$

The implied Newton step $\Delta \mathbf{x}^{(1)} = (-1, 17)$ moves the minimizing objective value from $f(-1, 1) = 15$ to $f(-2, 18) = 130, 286$. Hardly an improvement!

Another potential difficulty arises with the linear system that must be solved at each Newton iteration. How do we know that it can be solved efficiently; that is (Primer 5), what assures that Hessian matrix $\nabla^2 f(\mathbf{x}^{(t)})$ in [16.38](#) is nonsingular?

Near a strict local optimum, second-order sufficient conditions [16.22](#) suggest that the Hessian may be positive or negative definite. Either implies nonsingularity (Primer 8). But if we are farther away from an optimum, there is no guarantee whatever.

16.7 QUASI-NEWTON METHODS AND BFGS SEARCH

In Section 16.5 we saw that gradient search requires only first partial derivatives but often gives poor numerical performance. Newton's method of Section 16.6 yields much improved convergence but requires second derivatives and solving a system

of linear equations at each iteration. It is natural to look for a blend of the two that preserves their advantages while ameliorating their worst defects. That is precisely the idea behind **quasi-Newton methods**, which provide the most effective known algorithms for many unconstrained nonlinear programs.

Deflection Matrices

The Newton step of definition [16.38](#) solves

$$\nabla^2 f(\mathbf{x}^{(t)}) \Delta \mathbf{x} = -\nabla f(\mathbf{x}^{(t)})$$

for move $\Delta \mathbf{x}$. Assuming that the Hessian is nonsingular, we may left-multiply by its matrix inverse to express the move as

$$\Delta \mathbf{x} = -\nabla^2 f(\mathbf{x}^{(t)})^{-1} \nabla f(\mathbf{x}^{(t)})$$

That is, directions are computed by applying a suitable deflection matrix $\mathbf{D}_t \triangleq \nabla^2 f(\mathbf{x}^{(t)})^{-1}$ to the current gradient.

Definition 16.42

directions

Deflection matrices \mathbf{D}_t produce modified gradient search

$$\Delta \mathbf{x}^{(t+1)} = -\mathbf{D}_t \nabla f(\mathbf{x}^{(t)})$$

By stretching a point, we can also think of gradient search as a deflection matrix method. For example, the maximize case of Algorithm 16D employs directions

$$\Delta \mathbf{x} = \nabla f(\mathbf{x}^{(t)}) = -(-\mathbf{I}) \nabla f(\mathbf{x}^{(t)})$$

which can be viewed as adopting negative identity deflection matrix $\mathbf{D}_t = -\mathbf{I}$. The corresponding minimize case uses $\mathbf{D}_t = +\mathbf{I}$.

Quasi-Newton Approach

Quasi-Newton methods work with a deflection matrix that approximates the Hessian inverse $\nabla^2 f^{-1}(\mathbf{x}^{(t)})$ of Newton's method. Unlike the full Newton's method, however, this \mathbf{D}_t is built up from prior search results using only first derivatives.

The key to this approach is identifying properties that a deflection matrix should possess if it is to do the job of an inverse Hessian. Principal among these is the idea that the Hessian $\nabla^2 f(\mathbf{x}^{(t)})$ reflects the rates of change in first derivatives $\nabla f(\mathbf{x}^{(t)})$. As we move from $\mathbf{x}^{(t)}$ to $\mathbf{x}^{(t+1)}$, it follows that

$$\nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)}) \approx \nabla^2 f(\mathbf{x}^{(t)}) (\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})$$

or

$$\nabla^2 f(\mathbf{x}^{(t)})^{-1} (\nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)})) \approx \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$$

The analogous requirement on \mathbf{D}_t is known as the **quasi-Newton condition**.

Principle 16.43 | Deflection matrices of quasi-Newton algorithms approximate the gradient change behavior of inverse Hessian matrices by satisfying the quasi-Newton condition

$$\mathbf{D}_{t+1}\mathbf{g} = \mathbf{d}$$

at every iteration, where $\mathbf{d} \triangleq \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$ and $\mathbf{g} \triangleq \nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)})$.

Another characteristic property of Hessian matrices is their symmetry. For most common functions both $\nabla^2 f(\mathbf{x}^{(t)})$ and $\nabla^2 f(\mathbf{x}^{(t)})^{-1}$ are symmetric matrices. If quasi-Newton deflection matrices are to have any hope of approximating such inverse Hessians, they must also preserve this property.

Principle 16.44 | Deflection matrices of quasi-Newton algorithms should parallel inverse Hessians by being symmetric.

Guaranteeing Directions Improve

One of the difficulties that we encountered with Newton’s method in Section 16.6 is that it gives unpredictable results far from an optimal solution. It may not even produce an improving step.

We would like our quasi-Newton algorithms to avoid this difficulty. Recall from earliest principles [3.21] and [3.22] that direction $\Delta\mathbf{x}$ improves for a maximize problem at $\mathbf{x}^{(t)}$ if $\nabla f(\mathbf{x}^{(t)}) \cdot \Delta\mathbf{x} > 0$ and for a minimize if $\nabla f(\mathbf{x}^{(t)}) \cdot \Delta\mathbf{x} < 0$. With directions from deflection matrix definition [16.42], these conditions become

$$\nabla f(\mathbf{x}^{(t)})(-\mathbf{D}_t \nabla f(\mathbf{x}^{(t)})) = \nabla f(\mathbf{x}^{(t)})(-\mathbf{D}_t) \nabla f(\mathbf{x}^{(t)}) > 0$$

and

$$\nabla f(\mathbf{x}^{(t)})(-\mathbf{D}_t \nabla f(\mathbf{x}^{(t)})) = \nabla f(\mathbf{x}^{(t)})(-\mathbf{D}_t) \nabla f(\mathbf{x}^{(t)}) < 0$$

Notice that the maximize case will be satisfied for any gradient if every \mathbf{D}_t is negative definite, so that $-\mathbf{D}_t$ is positive definite (Primer 8). Similarly, the minimize case requires \mathbf{D}_t positive definite. These concerns motivate another specification.

Principle 16.45 | Deflection matrices in quasi-Newton algorithms should assure improving directions by keeping \mathbf{D}_t negative definite for maximize problems and positive definite for minimizes.

BFGS Formula

It turns out that a variety of deflection matrix update formulas can meet quasi-Newton requirements [16.43] to [16.45]. Still, one has proved more effective than all the others. Developed through the combined work of C. Broyden, R. Fletcher, D. Goldfarb, and D. Shanno, it is known as the BFGS formula.

Definition 16.46 The **BFGS formula** updates deflection matrices by

$$\mathbf{D}_{t+1} \leftarrow \mathbf{D}_t + \left(\mathbf{1} + \frac{\mathbf{g}\mathbf{D}_t\mathbf{g}}{\mathbf{d} \cdot \mathbf{g}} \right) \frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} - \frac{\mathbf{D}_t\mathbf{g}\mathbf{d}^T + \mathbf{d}\mathbf{g}^T\mathbf{D}_t}{\mathbf{d} \cdot \mathbf{g}}$$

where $\mathbf{d} \triangleq \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$ and $\mathbf{g} \triangleq \nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)})$.

Although it appears rather imposing, BFGS update [16.46](#) actually changes deflection matrices rather modestly at each iteration. The update has the form

$$\mathbf{D}_t + \varphi \mathbf{C}_1 - [(\mathbf{D}_t\mathbf{C}_2) + (\mathbf{D}_t\mathbf{C}_2)^T]$$

where weight

$$\varphi \triangleq 1 + \frac{\mathbf{g}\mathbf{D}_t\mathbf{g}}{\mathbf{d} \cdot \mathbf{g}} \quad (16.26)$$

is applied to combine simple matrices

$$\mathbf{C}_1 \triangleq \frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} = \frac{1}{\sum_j d_j g_j} \begin{pmatrix} (d_1)^2 & d_1 d_2 & \dots & d_1 d_n \\ d_2 d_1 & (d_2)^2 & \dots & d_2 d_n \\ \vdots & \vdots & \ddots & \vdots \\ d_n d_1 & d_n d_2 & \dots & (d_n)^2 \end{pmatrix} \quad (16.27)$$

$$\mathbf{C}_2 \triangleq \frac{\mathbf{g}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} = \frac{1}{\sum_j d_j g_j} \begin{pmatrix} g_1 d_1 & g_1 d_2 & \dots & g_1 d_n \\ g_2 d_1 & g_2 d_2 & \dots & g_2 d_n \\ \vdots & \vdots & \ddots & \vdots \\ g_n d_1 & g_n d_2 & \dots & g_n d_n \end{pmatrix}$$

Notice that both \mathbf{C}_1 and \mathbf{C}_2 are **rank one** with every row a multiple of every other.

BFGS Search of Custom Computer Application

Algorithm 16F details a search algorithm based on BFGS update formula [16.46](#). Table 16.7 and Figure 16.18 then track its application to our Custom Computer curve-fitting model (16.7).

The initial iteration of Algorithm 16F for this minimize problem employs identity deflection matrix $\mathbf{D}_0 = \mathbf{I}$, which leaves the first direction

$$\Delta \mathbf{x}^{(1)} = -\mathbf{D}_0 \nabla f(\mathbf{x}^{(0)}) = -\mathbf{I} \nabla f(\mathbf{x}^{(0)}) = -\nabla f(\mathbf{x}^{(0)})$$

Thus our BFGS procedure follows the same first direction as gradient search Algorithm 16D.

Like gradient Algorithm 16D, and unlike Newton Algorithm 16E, quasi-Newton methods require line search. Table 16.7 shows that the first such search produces step $\lambda = 0.0001$. The result is

$$\begin{aligned} \mathbf{x}^{(1)} &\leftarrow \mathbf{x}^{(0)} + \lambda_1 \Delta \mathbf{x}_1 \\ &= (32.00, -0.4000) + (0.0001)(6.240, -1053) \\ &= (32.00, -0.4685) \end{aligned}$$

with gradient $\nabla f(\mathbf{x}^{(1)}) = (-23.02, 2.514)$. Thus

$$\begin{aligned} \mathbf{d} &\triangleq \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} = \begin{pmatrix} 32.0004 \\ -0.4685 \end{pmatrix} - \begin{pmatrix} 32 \\ -0.4 \end{pmatrix} = \begin{pmatrix} 0.0004 \\ -0.0685 \end{pmatrix} \\ \mathbf{g} &\triangleq \nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)}) = \begin{pmatrix} -23.02 \\ 2.514 \end{pmatrix} - \begin{pmatrix} -6.240 \\ 1053.4 \end{pmatrix} = \begin{pmatrix} -16.78 \\ -1050.9 \end{pmatrix} \end{aligned}$$

and

$$\mathbf{d} \cdot \mathbf{g} = (0.0004, -0.0685) \cdot (-16.78, -1050.9) = 71.9$$

ALGORITHM 16F: BFGS QUASI-NEWTON SEARCH

Step 0: Initialization. Choose any starting solution $\mathbf{x}^{(0)}$, compute gradient $\nabla f(\mathbf{x}^{(0)})$ and pick stopping tolerance $\epsilon > 0$. Also initialize deflection matrix

$$\mathbf{D}_0 = \mp \mathbf{I}$$

($-$ for a maximize, $+$ for a minimize) and set solution index $t \leftarrow 0$.

Step 1: Stationary Point. If norm $\|\nabla f(\mathbf{x}^{(t)})\| < \epsilon$, stop. Point $\mathbf{x}^{(t)}$ is sufficiently close to a stationary point.

Step 2: Direction. Use the current deflection matrix \mathbf{D}_t to compute the move direction

$$\Delta \mathbf{x}^{(t+1)} \leftarrow -\mathbf{D}_t \nabla f(\mathbf{x}^{(t)})$$

Step 3: Line Search. Solve (at least approximately) 1-dimensional line search max or min $f(\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)})$ to compute step size λ_{t+1} .

Step 4: New Point. Update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda_{t+1} \Delta \mathbf{x}^{(t+1)}$$

and compute new gradient $\nabla f(\mathbf{x}^{(t+1)})$.

Step 5: Deflection Matrix. Revise the deflection matrix as

$$\mathbf{D}_{t+1} \leftarrow \mathbf{D}_t + \left(1 + \frac{\mathbf{g} \mathbf{D}_t \mathbf{g}}{\mathbf{d} \cdot \mathbf{g}} \right) \frac{\mathbf{d} \mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} - \frac{\mathbf{D}_t \mathbf{g} \mathbf{d}^T + \mathbf{d} \mathbf{g} \mathbf{D}_t}{\mathbf{d} \cdot \mathbf{g}}$$

where $\mathbf{d} \triangleq (\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})$ and $\mathbf{g} \triangleq (\nabla f(\mathbf{x}^{(t+1)}) - \nabla f(\mathbf{x}^{(t)}))$.

Step 6: Advance. Increment $t \leftarrow t + 1$, and return to Step 1.

TABLE 16.7 BFGS Search of Custom Computer Application

t	$\mathbf{x}^{(t)}$	$f(\mathbf{x}^{(t)})$	$\nabla f(\mathbf{x}^{(t)})$	$\ \nabla f(\mathbf{x}^{(t)})\ $	\mathbf{D}_t	$\Delta \mathbf{x}^{(t+1)}$	λ_{t+1}
0	(32.00, -0.4000)	174.746	(-6.240, 1053)	1053.4	$\begin{pmatrix} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{pmatrix}$	(6.240, -1053)	0.0001
1	(32.00, -0.4685)	141.139	(-23.02, 2.514)	23.15	$\begin{pmatrix} 1.0002 & -0.0160 \\ -0.0160 & 0.0003 \end{pmatrix}$	(23.06, -0.3684)	0.3755
2	(40.66, -0.6068)	43.258	(-0.823, -51.54)	51.54	$\begin{pmatrix} 0.3759 & -0.0059 \\ -0.0059 & 0.0002 \end{pmatrix}$	(0.0079, 0.0032)	1.4361
3	(4.067, -0.6021)	43.133	(-0.041, 0.100)	0.11	$\begin{pmatrix} 0.3775 & -0.0055 \\ -0.0055 & 0.0002 \end{pmatrix}$	(0.0160, -0.0002)	1.0604
4	(40.69, -0.6024)	43.132	(0.000, -0.008)	0.01	Stop		

Next we compute main update matrices

$$\frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} = \frac{1}{71.9} \begin{pmatrix} 0.0004 & \\ & -0.4685 \end{pmatrix} (0.0004, -0.4685) = \begin{pmatrix} 0.0000000 & -0.0000004 \\ -0.0000004 & 0.0000652 \end{pmatrix}$$

$$\frac{\mathbf{D}_0 \mathbf{g} \mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} = \frac{1}{71.9} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.78 \\ -1050.9 \end{pmatrix} (0.0004, -0.4685) = \begin{pmatrix} -0.000095 & 0.015975 \\ -0.005927 & 1.00062 \end{pmatrix}$$

Then with

$$\mathbf{g} \mathbf{D}_0 \mathbf{g}^T = (-16.78, -1050.9) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.78 \\ -1050.9 \end{pmatrix} = 1,104,583$$

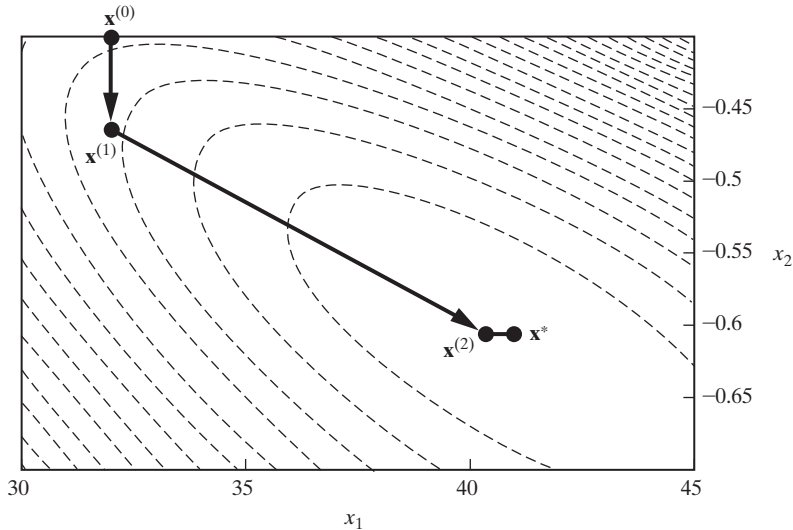


FIGURE 16.18 BFGS Search of Custom Computer Application

the new deflection matrix is

$$\begin{aligned} \mathbf{D}_1 &\leftarrow \mathbf{D}_0 + \left(1 + \frac{\mathbf{g}\mathbf{D}_0\mathbf{g}}{\mathbf{d}\cdot\mathbf{g}}\right) \frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d}\cdot\mathbf{g}} - \frac{\mathbf{D}_0\mathbf{g}\mathbf{d}^T + \mathbf{d}\mathbf{g}^T\mathbf{D}_0}{\mathbf{d}\cdot\mathbf{g}} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \left(1 + \frac{1,104,583}{71.9}\right) \begin{pmatrix} 0.0000000 & -0.0000004 \\ -0.0000004 & 0.0000652 \end{pmatrix} \\ &\quad - \left[\begin{pmatrix} -0.000095 & 0.015975 \\ -0.005927 & 1.00062 \end{pmatrix} + \begin{pmatrix} -0.000095 & -0.005927 \\ 0.015975 & 1.00062 \end{pmatrix} \right] \\ &= \begin{pmatrix} 1.0002 & -0.0160 \\ -0.0160 & 0.0003 \end{pmatrix} \end{aligned}$$

This revised deflection matrix produces the next move direction,

$$\Delta\mathbf{x}^{(1)} \leftarrow -\mathbf{D}_1 \nabla f(\mathbf{x}^{(1)}) = -\begin{pmatrix} 1.0002 & -0.0160 \\ -0.0160 & 0.0003 \end{pmatrix} \begin{pmatrix} -23.02 \\ 2.514 \end{pmatrix} = \begin{pmatrix} 23.06 \\ -0.3684 \end{pmatrix}$$

and the search continues.

Algorithm 16F stops when $\|\nabla f(\mathbf{x}^{(t)})\| < \varepsilon$, indicating that we have reached an approximately stationary point. Using $\varepsilon = 0.1$, this occurs at iteration $t = 4$ of Table 16.7.

EXAMPLE 16.19: EXECUTING BFGS SEARCH

Suppose that BFGS Algorithm 16F reaches iteration $t = 5$ with

$$\mathbf{x}^{(5)} = (10, 16), \nabla f(\mathbf{x}^{(5)}) = (-1, 1), \mathbf{D}_5 = \begin{pmatrix} -10 & 2 \\ 2 & -4 \end{pmatrix}$$

of a maximizing search, and then takes a step of $\lambda_6 = \frac{1}{2}$ in the BFGS direction to reach a new $\mathbf{x}^{(6)}$ with $\nabla f(\mathbf{x}^{(6)}) = (5, -3)$.

- (a) Determine the direction $\Delta\mathbf{x}^{(6)}$ that was employed and the new solution $\mathbf{x}^{(6)}$.
- (b) Compute the revised deflection matrix \mathbf{D}_6 needed for the next iteration.

Solution:

(a) Following gradient deflection computation 16.42

$$\Delta\mathbf{x}^{(6)} \leftarrow -\mathbf{D}_5 \nabla f(\mathbf{x}^{(5)}) = -\begin{pmatrix} -10 & 2 \\ 2 & -4 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -12 \\ 6 \end{pmatrix}$$

Thus the new solution

$$\mathbf{x}^{(6)} = \mathbf{x}^{(5)} + \lambda_6 \Delta\mathbf{x}^{(6)} = \begin{pmatrix} 10 \\ 16 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -12 \\ 6 \end{pmatrix} = \begin{pmatrix} 4 \\ 19 \end{pmatrix}$$

(b) We apply BFGS formula [16.46](#). First, the difference vectors are

$$\begin{aligned}\mathbf{d} &\triangleq \mathbf{x}^{(6)} - \mathbf{x}^{(5)} = \begin{pmatrix} 4 \\ 19 \end{pmatrix} - \begin{pmatrix} 10 \\ 16 \end{pmatrix} = \begin{pmatrix} -6 \\ 3 \end{pmatrix} \\ \mathbf{g} &\triangleq \nabla f(\mathbf{x}^{(6)}) - \nabla f(\mathbf{x}^{(5)}) = \begin{pmatrix} 5 \\ -3 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ -4 \end{pmatrix}\end{aligned}$$

with $\mathbf{d} \cdot \mathbf{g} = -48$. The update matrices are then

$$\begin{aligned}\frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} &= \frac{1}{-48} \begin{pmatrix} -6 & \\ & 3 \end{pmatrix} (-6.3) = \begin{pmatrix} -0.75 & 0.375 \\ 0.375 & -0.1875 \end{pmatrix} \\ \frac{\mathbf{D}_5 \mathbf{g}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} &= \frac{1}{-48} \begin{pmatrix} -10 & 2 \\ & -4 \end{pmatrix} \begin{pmatrix} 6 \\ -4 \end{pmatrix} (-6.3) = \begin{pmatrix} -8.5 & 4.25 \\ 3.5 & -1.75 \end{pmatrix}\end{aligned}$$

Also,

$$\mathbf{g}\mathbf{D}_5\mathbf{g} = (6, -4) \begin{pmatrix} -10 & 2 \\ 2 & -4 \end{pmatrix} \begin{pmatrix} 6 \\ -4 \end{pmatrix} = -520$$

Now substituting in formula [16.46](#) gives

$$\begin{aligned}\mathbf{D}_6 &\leftarrow \mathbf{D}_5 + \left(\mathbf{1} + \frac{\mathbf{g}\mathbf{D}_5\mathbf{g}}{\mathbf{d} \cdot \mathbf{g}} \right) \frac{\mathbf{d}\mathbf{d}^T}{\mathbf{d} \cdot \mathbf{g}} - \frac{\mathbf{D}_5\mathbf{g}\mathbf{d}^T + \mathbf{d}\mathbf{g}^T\mathbf{D}_5}{\mathbf{d} \cdot \mathbf{g}} \\ &= \begin{pmatrix} -10 & 2 \\ 2 & -4 \end{pmatrix} + \left(1 + \frac{-520}{-48} \right) \begin{pmatrix} -0.75 & 0.375 \\ 0.375 & -0.1875 \end{pmatrix} \\ &\quad - \left[\begin{pmatrix} -8.5 & 4.25 \\ 3.5 & -1.75 \end{pmatrix} + \begin{pmatrix} -8.5 & 3.5 \\ 4.25 & -1.75 \end{pmatrix} \right] \\ &= \begin{pmatrix} -1.8750 & -1.3125 \\ -1.3125 & -2.7188 \end{pmatrix}\end{aligned}$$

Verifying Quasi-Newton Requirements

Although proving most of them is beyond the scope of this book, BFGS update [16.46](#) can be shown to fulfill all of our quasi-Newton requirements.

Principle 16.47 BFGS update formula [16.46](#) produces deflection matrices satisfying quasi-Newton condition [16.43](#) at every iteration, as well as symmetry requirement [16.44](#) and improving direction specification [16.45](#).

To illustrate, focus on $t = 1$ in Table 16.7. There

$$\begin{aligned}\mathbf{d} = \mathbf{x}^{(2)} - \mathbf{x}^{(1)} &= \begin{pmatrix} 40.66 \\ -0.6068 \end{pmatrix} - \begin{pmatrix} 32.00 \\ -0.4685 \end{pmatrix} = \begin{pmatrix} 8.66 \\ -0.1383 \end{pmatrix} \\ \mathbf{g} = \nabla f(\mathbf{x}^{(2)}) - \nabla f(\mathbf{x}^{(1)}) &= \begin{pmatrix} -0.823 \\ -51.54 \end{pmatrix} - \begin{pmatrix} -23.02 \\ 2.514 \end{pmatrix} = \begin{pmatrix} 22.193 \\ -54.05 \end{pmatrix}\end{aligned}$$

Thus quasi-Newton principle [16.43](#) $\mathbf{D}_2\mathbf{g} = \mathbf{d}$ checks

$$\mathbf{D}_2\mathbf{g} = \begin{pmatrix} 0.37594 & -0.00585 \\ -0.00585 & 0.00016 \end{pmatrix} \begin{pmatrix} 22.193 \\ -54.05 \end{pmatrix} \approx \begin{pmatrix} 8.66 \\ -0.1383 \end{pmatrix} = \mathbf{d}$$

It is easy to see that all deflection matrices \mathbf{D}_t in Table 16.7 are also symmetric and positive definite (as required for a minimize problem). For instance, with \mathbf{D}_3 the principal minor determinants are

$$\det(0.3775) = 0.3775 > 0 \quad \text{and} \quad \det \begin{pmatrix} 0.3775 & -0.0055 \\ -0.0055 & 0.0002 \end{pmatrix} = 0.00004 > 0$$

EXAMPLE 16.20: VERIFYING QUASI-NEWTON REQUIREMENTS

Return to the maximize model of Example 16.19, and demonstrate that the computed \mathbf{D}_6 satisfies quasi-Newton algorithm principles [16.43](#) to [16.45](#).

Solution: Quasi-Newton principle [16.43](#) is $\mathbf{D}_6\mathbf{g} = \mathbf{d}$. Checking gives

$$\mathbf{D}_6\mathbf{g} = \begin{pmatrix} -1.8750 & -1.3125 \\ -1.3125 & -2.7188 \end{pmatrix} \begin{pmatrix} 6 \\ -4 \end{pmatrix} = \begin{pmatrix} -6 \\ 3 \end{pmatrix} = \mathbf{d}$$

as required. Also, matrix \mathbf{D}_6 is symmetric (principle [16.44](#)) because $d_{1,2}^{(6)} = d_{2,1}^{(6)} = -1.3125$

To guarantee an improving direction for a maximize problem, \mathbf{D}_6 should also be negative definite (principle [16.45](#)). This too is true because principal minor determinants

$$\det(-1.8750) = -1.8750 \quad \text{and} \quad \det \begin{pmatrix} -1.8750 & -1.3125 \\ -1.3125 & -2.7188 \end{pmatrix} = 3.375$$

alternate in sign and the first is negative.

Approximating the Hessian Inverse with BFGS

Quasi-Newton principles [16.43](#) to [16.45](#) were motivated to mimic Newton’s method’s use of Hessian inverse deflection matrices $\mathbf{D}_t = \nabla^2 f(\mathbf{x}^{(t)})^{-1}$. It should not surprise that BFGS and many other quasi-Newton deflection matrices tend to this Newton case.

Principle 16.48 | As BFGS Algorithm 16F nears a local optimum, deflection matrices \mathbf{D}_t approach the inverse Hessian matrix at that optimum.

Once again, we may illustrate with results for our Custom Computer application. The final deflection matrix of BFGS Table 16.7 is

$$\mathbf{D}_3 = \begin{pmatrix} 0.3775 & -0.0055 \\ -0.0055 & 0.0002 \end{pmatrix}$$

The corresponding Hessian matrix is

$$\nabla^2 f(\mathbf{x}^{(3)}) = \begin{pmatrix} 4.811 & 158.8 \\ 158.8 & 10,903 \end{pmatrix}$$

with inverse

$$\nabla^2 f(\mathbf{x}^{(3)})^{-1} = \begin{pmatrix} 0.4004 & -0.0058 \\ -0.0058 & 0.0002 \end{pmatrix}$$

In accord with principle [16.48](#), this \mathbf{D}_3 closely approximates $\nabla^2 f(\mathbf{x}^{(3)})^{-1}$.

16.8 OPTIMIZATION WITHOUT DERIVATIVES AND NELDER–MEAD

Sometimes nonlinear programs must be addressed over objective functions that are not differentiable, or at least do not have readily computable derivatives. In such cases, improving search must rely entirely on functional evaluations.

How do algorithms choose search directions without derivatives? Numerous schemes have been proposed. Some simply use the coordinate directions—searching each in turn. Others seek to align the search with the trend of recent progress. We develop here only the method due to Nelder and Mead, which constructs directions by maintaining an ensemble of current points.

Nelder–Mead Strategy

One of the most popular schemes for unconstrained search without derivatives is the Nelder–Mead procedure detailed in Algorithm 16G. Table 16.8 traces its application to our PERT maximum likelihood model (16.10).

In contrast to other improving search methods, which keep only one current point, Nelder–Mead Algorithm 16G maintains a set of $n + 1$.

Definition 16.49 In an optimization over n decision variables, the Nelder–Mead algorithm maintains an ensemble of $n + 1$ distinct solutions $\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n+1)}$, with $\mathbf{y}^{(1)}$ having the best objective function value, $\mathbf{y}^{(2)}$ the second best, and so on.

Each iteration of the search tries to replace the worst solution $\mathbf{y}^{(n+1)}$ with a better one.

The maximization of Table 16.8 illustrates for $n = 2$. Search begins with the ensemble

$$\begin{aligned} \mathbf{y}^{(1)} &= (5, 3), \mathbf{y}^{(2)} = (6, 3), \mathbf{y}^{(3)} = (6, 4) \\ f(\mathbf{y}^{(1)}) &= 12.425, f(\mathbf{y}^{(2)}) = 11.429, f(\mathbf{y}^{(3)}) = 2.663 \end{aligned}$$

Notice that the solutions are numbered from best to worst.

It is somewhat arbitrary that exactly $n + 1$ solutions are maintained in the ensemble. Still, too many would complicate computation, and too few would not adequately surround an emerging optimum. Over n decision variables, $n + 1$ solutions is just enough to define vertices of a polytope surrounding a point.

ALGORITHM 16G: NELDER–MEAD DERIVATIVE–FREE SEARCH

Step 0: Initialization. Choose $(n + 1)$ distinct solutions $\mathbf{x}^{(i)}$ as starting set $\{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n+1)}\}$, evaluate $f(\mathbf{y}^{(1)}), \dots, f(\mathbf{y}^{(n+1)})$, and initialize iteration index $t \leftarrow 0$.

Step 1: Centroid. Renumber as necessary to arrange the $\mathbf{y}^{(i)}$ in nonimproving sequence by solution value. Then compute best- n centroid

$$\mathbf{x}^{(t)} = \frac{1}{n} \sum_{i=1}^n \mathbf{y}^{(i)}$$

Step 2: Stopping. If all solution values $f(\mathbf{y}^{(1)}), \dots, f(\mathbf{y}^{(n)})$ are sufficiently close to centroid objective value $f(\mathbf{x}^{(t)})$, stop and report the best of $\mathbf{y}^{(1)}$ and $\mathbf{x}^{(t)}$.

Step 3: Direction. Use centroid $\mathbf{x}^{(t)}$ to compute away-from-worst move direction

$$\Delta \mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \mathbf{y}^{(n+1)}$$

Step 4: Reflection. Try $\lambda = 1$ by computing $f(\mathbf{x}^{(t)} + 1\Delta \mathbf{x}^{(t+1)})$. If this new value is at least as good as current best $f(\mathbf{y}^{(1)})$, go to Step 5 and expand. If it is no better than second-worst value $f(\mathbf{y}^{(n)})$, go to Step 6 and contract. Otherwise, accept $\lambda \leftarrow 1$, and proceed to Step 8.

Step 5: Expansion. Try $\lambda = 2$ by computing $f(\mathbf{x}^{(t)} + 2\Delta \mathbf{x}^{(t+1)})$. If this value is no worse than $f(\mathbf{x}^{(t)} + 1\Delta \mathbf{x}^{(t+1)})$ fix $\lambda \leftarrow 2$, and otherwise set $\lambda \leftarrow 1$. Then proceed to Step 8.

Step 6: Contraction. If reflection value $f(\mathbf{x}^{(t)} + 1\Delta \mathbf{x}^{(t+1)})$ is better than worst current $f(\mathbf{y}^{(n+1)})$, try $\lambda = \frac{1}{2}$ by computing $f(\mathbf{x}^{(t)} + \frac{1}{2}\Delta \mathbf{x}^{(t+1)})$. If not, try $\lambda = -\frac{1}{2}$ by evaluating $f(\mathbf{x}^{(t)} - \frac{1}{2}\Delta \mathbf{x}^{(t+1)})$. Either way, if the result improves on worst current $f(\mathbf{y}^{(n+1)})$, fix λ at the $\pm \frac{1}{2}$ tried and proceed to Step 8. Otherwise, go to Step 7 to shrink.

Step 7: Shrinking. Shrink the current solution set toward best $\mathbf{y}^{(1)}$ by

$$\mathbf{y}^{(i)} \leftarrow \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(i)}) \quad \text{for all } i = 2, \dots, n + 1$$

Then compute new $f(\mathbf{y}^{(2)}), \dots, f(\mathbf{y}^{(n+1)})$, advance $t \leftarrow t + 1$, and return to Step 1.

Step 8: Replacement. Replace worst $\mathbf{y}^{(n+1)}$ in the solution set by

$$\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)}$$

Then advance $t \leftarrow t + 1$ and return to Step 1.

TABLE 16.8 Nelder–Mead Search of PERT Application

t	$\mathbf{y}^{(1)}$	$\mathbf{y}^{(2)}$	$\mathbf{y}^{(3)}$	$\mathbf{y}^{(r)}$	$\Delta \mathbf{x}^{(r+1)}$	Reflect	Second
0	(5.000, 3.000) $f = 12.425$	(6.000, 3.000) $f = 11.429$	(6.000, 4.000) $f = 2.663$	(5.500, 3.000) $f = 13.084$	(-0.500, -1.000)	$\lambda = 1.0$ $f = 11.415$	$\lambda = 0.5$ $f = 24.915$
1	(5.250, 2.500) $f = 24.915$	(5.000, 3.000) $f = 12.425$	(6.000, 3.000) $f = 11.429$	(5.125, 2.750) $f = 13.074$	(-0.875, -0.250)	$\lambda = 1.0$ $f = 30.005$	$\lambda = 2.0$ $f = 19.654$
2	(4.250, 2.500) $f = 30.005$	(5.250, 2.500) $f = 24.915$	(5.000, 3.000) $f = 12.425$	(4.750, 2.500) $f = 30.482$	(-0.250, -0.500)	$\lambda = 1.0$ $f = 18.229$	$\lambda = 0.5$ $f = 17.354$
3	(4.250, 2.500) $f = 30.005$	(5.250, 2.500) $f = 24.915$	(4.625, 2.250) $f = 17.354$	(4.750, 2.500) $f = 30.482$	(-0.203, -0.031)	$\lambda = 1.0$ $f = 21.434$	$\lambda = 0.5$ $f = 11.231$
4	(4.438, 2.375) $f = 35.513$	(4.750, 2.500) $f = 30.482$	(4.250, 2.500) $f = 30.005$	(4.594, 2.438) $f = 16.119$	(0.344, -0.063)	$\lambda = 1.0$ $f = 31.642$	
5	(4.438, 2.375) $f = 35.513$	(4.938, 2.375) $f = 31.642$	(4.750, 2.500) $f = 30.482$	(4.688, 2.375) $f = 15.893$	(-0.063, -0.125)	$\lambda = 1.0$ $f = 17.354$	$\lambda = -0.5$ $f = 26.577$
6	(4.438, 2.375) $f = 35.513$	(4.594, 2.438) $f = 16.119$	(4.688, 2.375) $f = 15.893$	(4.516, 2.406) $f = 25.980$	(-0.172, 0.031)	$\lambda = 1.0$ $f = 31.062$	
7	(4.438, 2.375) $f = 35.513$	(4.344, 2.438) $f = 31.062$	(4.594, 2.438) $f = 16.119$	(4.391, 2.406) $f = 32.845$	(-0.203, -0.031)	$\lambda = 1.0$ $f = 17.253$	$\lambda = 0.5$ $f = 28.811$
8	(4.438, 2.375) $f = 35.513$	(4.344, 2.438) $f = 31.062$	(4.289, 2.391) $f = 28.811$	(4.391, 2.406) $f = 32.845$	(0.102, 0.016)	$\lambda = 1.0$ $f = 29.758$	$\lambda = 0.5$ $f = 31.979$
9	(4.438, 2.375) $f = 35.513$	(4.441, 2.414) $f = 31.979$	(4.344, 2.438) $f = 31.062$	(4.439, 2.395) $f = 33.857$	(0.096, -0.043)	$\lambda = 1.0$ $f = 26.094$	$\lambda = -0.5$ $f = 32.250$
10	(4.438, 2.375) $f = 35.513$	(4.392, 2.416) $f = 32.250$	(4.441, 2.414) $f = 31.979$	(4.415, 2.396) $f = 33.726$	(-0.027, -0.019)	$\lambda = 1.0$ $f = 34.140$	
11	(4.438, 2.375) $f = 35.513$	(4.388, 2.377) $f = 34.140$	(4.392, 2.416) $f = 32.250$	(4.413, 2.376) $f = 34.957$	(0.021, -0.040)	$\lambda = 1.0$ $f = 37.893$	$\lambda = 2.0$ $f = 41.050$
12	(4.455, 2.296) $f = 41.050$	(4.438, 2.375) $f = 35.513$	(4.388, 2.377) $f = 34.140$	(4.446, 2.335) $f = 38.569$	(0.058, -0.042)	$\lambda = 1.0$ $f = 42.813$	$\lambda = 2.0$ $f = 27.929$
13	(4.504, 2.294) $f = 42.813$	(4.455, 2.296) $f = 41.050$	(4.438, 2.375) $f = 35.513$	(4.479, 2.295) $f = 43.202$	(0.042, -0.080)	$\lambda = 1.0$ $f = 41.193$	
14	(4.504, 2.294) $f = 42.813$	(4.521, 2.215) $f = 41.193$	(4.455, 2.296) $f = 41.050$	(4.513, 2.254) $f = 42.388$	(0.058, -0.042)	$\lambda = 1.0$ $f = 28.723$	$\lambda = -0.5$ $f = 44.727$
15	(4.484, 2.275) $f = 44.727$	(4.504, 2.294) $f = 42.813$	(4.521, 2.215) $f = 41.193$	(4.494, 2.285) $f = 45.087$	(-0.027, 0.070)	$\lambda = 1.0$ $f = 37.793$	$\lambda = -0.5$ $f = 44.627$
16	(4.484, 2.275) $f = 44.727$	(4.508, 2.250) $f = 44.627$	(4.504, 2.294) $f = 42.813$	(4.496, 2.262) $f = 46.671$	(-0.009, -0.032)	$\lambda = 1.0$ $f = 46.356$	$\lambda = 2.0$ $f = 43.832$
17	(4.487, 2.231) $f = 46.356$	(4.484, 2.275) $f = 44.727$	(4.508, 2.250) $f = 44.627$	(4.485, 2.253) $f = 45.790$	(-0.022, 0.003)	$\lambda = 1.0$ $f = 42.708$	$\lambda = -0.5$ $f = 47.301$
18	(4.497, 2.251) $f = 47.301$	(4.487, 2.231) $f = 46.356$	(4.484, 2.275) $f = 44.727$	(4.492, 2.241) $f = 46.977$	(0.008, -0.034)	$\lambda = 1.0$ $f = 48.520$	$\lambda = 2.0$ $f = 44.839$
19	(4.500, 2.207) $f = 48.520$	(4.497, 2.251) $f = 47.301$	(4.487, 2.231) $f = 46.356$	(4.498, 2.229) $f = 48.192$	(0.011, -0.002)	$\lambda = 1.0$ $f = 44.882$	$\lambda = -0.5$ $f = 47.275$
20	(4.500, 2.207) $f = 48.520$	(4.497, 2.251) $f = 47.301$	(4.493, 2.230) $f = 47.275$	(4.498, 2.229) $f = 48.192$	Stop		

EXAMPLE 16.21: ARRANGING A NELDER–MEAD ENSEMBLE

Consider the unconstrained nonlinear program

$$\min f(x_1, x_2, x_3) \triangleq (x_1)^2 + (x_2 - 1)^2 + (x_3 + 4)^2$$

Choose an initial ensemble of solutions for Nelder–Mead search and arrange them with the notation of [16.49].

Solution: With $n = 3$ we need 4 distinct solutions that “bracket” a region likely to produce good solutions. One possibility is

$$\begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}, \begin{pmatrix} -3 \\ 3 \\ -3 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \\ 3 \end{pmatrix}, \text{ and } \begin{pmatrix} -3 \\ 0 \\ -3 \end{pmatrix}$$

Evaluating the objective gives

$$f(3, 3, 3) = 62, \quad f(-3, 3, -3) = 14, \quad f(3, 0, 3) = 59, \quad \text{and} \quad f(-3, 0, -3) = 11$$

Thus, for this minimizing model, we have initial ensemble

$$\mathbf{y}^{(1)} = \begin{pmatrix} -3 \\ 0 \\ -3 \end{pmatrix}, \quad \mathbf{y}^{(2)} = \begin{pmatrix} -3 \\ 3 \\ -3 \end{pmatrix}, \quad \mathbf{y}^{(3)} = \begin{pmatrix} 3 \\ 0 \\ 3 \end{pmatrix}, \quad \text{and} \quad \mathbf{y}^{(4)} = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}$$

Nelder–Mead Direction

The critical issue in any derivative-free optimization method is how to construct search directions without the aid of partial derivatives. Nelder–Mead Algorithm 16G adopts an away-from-worst approach.

Principle 16.50 At iteration t , the Nelder–Mead algorithm employs search direction

$$\Delta \mathbf{x} \triangleq \mathbf{x}^{(t)} - \mathbf{y}^{(n+1)}$$

which moves away from worst current solution $\mathbf{y}^{(n+1)}$ through the best- n centroid

$$\mathbf{x}^{(t)} \triangleq \frac{1}{n} \sum_{i=1}^n \mathbf{y}^{(i)}$$

The idea is to move away from the worst solution in the ensemble and toward the rest.

Figure 16.19 illustrates for $t = 0$ of the 2-variable search in Table 16.8. The centroid of the best two solutions is

$$\mathbf{x}^{(0)} \triangleq \frac{1}{2} \left[\begin{pmatrix} 5 \\ 3 \end{pmatrix} + \begin{pmatrix} 6 \\ 3 \end{pmatrix} \right] = \begin{pmatrix} 5.5 \\ 3 \end{pmatrix}$$

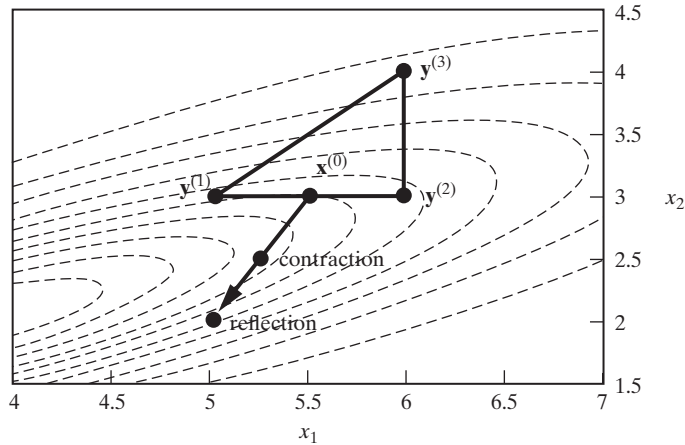


FIGURE 16.19 Nelder-Mead Direction for PERT Application

Thus the search direction of [16.50](#) is

$$\Delta \mathbf{x} \triangleq \mathbf{x}^{(0)} - \mathbf{y}^{(3)} = \begin{pmatrix} 5.5 \\ 3 \end{pmatrix} - \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} -0.5 \\ -1.0 \end{pmatrix}$$

EXAMPLE 16.22: COMPUTING NELDER–MEAD DIRECTIONS

Return to Example 16.21, and compute the Nelder–Mead direction corresponding to the initial ensemble constructed there.

Solution: The centroid of the best 3 solutions in that 3-variable model is

$$\begin{aligned} \mathbf{x}^{(0)} &= \frac{1}{3} (\mathbf{y}^{(1)} + \mathbf{y}^{(2)} + \mathbf{y}^{(3)}) \\ &= \frac{1}{3} [(-3, 0, -3) + (-3, 3, -3) + (3, 0, 3)] \\ &= (-1, 1, -1) \end{aligned}$$

Thus direction [16.50](#) becomes

$$\Delta \mathbf{x} = \mathbf{x}^{(0)} - \mathbf{y}^{(4)} = (-1, 1, -1) - (3, 3, 3) = (-4, -2, -4)$$

Nelder–Mead Limited Step Sizes

Centroid $\mathbf{x}^{(t)}$ plays the role of a current solution in Nelder–Mead Algorithm 16G. But what step size λ should be applied to direction [16.50](#)?

One strategy would undertake a full line search over

$$\mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$$

However, the required computation is usually not justified for such a crudely derived direction $\Delta \mathbf{x}$. Also, too large a step would destroy the “spread” pattern of the ensemble.

Algorithm 16G confronts these concerns by trying only one or two λ 's drawn from possibilities $\{+1, +2, +\frac{1}{2}, -\frac{1}{2}\}$.

Principle 16.51 The Nelder–Mead algorithm explores new points $\mathbf{x}^{(t)} + \lambda\Delta\mathbf{x}$ by first **reflecting** about centroid $\mathbf{x}^{(t)}$ with $\lambda = 1$. If replacing $\mathbf{y}^{(n+1)}$ with this new point would leave it neither best nor worst in the ensemble, it is adopted without further trials. If the reflection point is a new best, the algorithm **expands** to try $\lambda = 2$. If the point would be worst, the procedure **contracts** to try either $\lambda = +\frac{1}{2}$ or $\lambda = -\frac{1}{2}$.

Figure 16.19 illustrates for $t = 0$ in Table 16.8. Trying $\lambda = 1$ produces reflection point

$$\mathbf{x}^{(0)} + \Delta\mathbf{x} = (5.5, 3) + (-0.5, -1.0) = (5, 2)$$

with $f(5, 2) = 11.415$. The term reflection comes from the fact that this new point mirrors worst ensemble $\mathbf{y}^{(3)} = (6, 4)$ on the opposite side of centroid $\mathbf{x}^{(0)} = (5.5, 3)$.

Immediately replacing $\mathbf{y}^{(3)}$ with this reflection point would leave it worst in the new ensemble. Thus we try to do better by contracting with $\lambda = +\frac{1}{2}$. (Contraction would use $\lambda = -\frac{1}{2}$ if the reflection point had been no better than worst $\mathbf{y}^{(3)}$.)

The resulting $(5.25, 2.5)$ yields far superior $f(5.25, 2.5) = 24.915$. Substituting this contraction point for the worst in the current ensemble produces the improved set

$$\begin{aligned} \mathbf{y}^{(1)} &= (5.25, 2.5), \mathbf{y}^{(2)} = (5, 3), \mathbf{y}^{(3)} = (6, 3) \\ f(\mathbf{y}^{(1)}) &= 24.915, f(\mathbf{y}^{(2)}) = 12.425, f(\mathbf{y}^{(3)}) = 11.429 \end{aligned}$$

of $t = 1$. Notice that points have been renumbered to keep them in objective function value sequence.

Table 16.8 shows the direction 16.50 for this new ensemble is $\Delta\mathbf{x} = (-0.875, -0.250)$, and new centroid $\mathbf{x}^{(1)} = (5.125, 2.75)$. Now reflection with $\lambda = 1$ produces new best $(4.25, 2.5)$ with $f(4.25, 2.5) = 30.005$. This leads to trying expansion with $\lambda = 2$. However, the result is not as good, so the reflection point joins the ensemble.

EXAMPLE 16.23: EXECUTING NELDER–MEAD SEARCH

Suppose that a minimizing Nelder–Mead search has current ensemble objective values $f(\mathbf{y}^{(1)}) = 13, f(\mathbf{y}^{(2)}) = 21, f(\mathbf{y}^{(3)}) = 25$, and $f(\mathbf{y}^{(4)}) = 50$. For cases (a) to (d) in the following table, describe which step sizes λ would be tried at this iteration, and which (if any) should be chosen.

	$f(\mathbf{x}^{(t)} - \frac{1}{2}\mathbf{x}\Delta)$	$f(\mathbf{x}^{(t)} + \frac{1}{2}\mathbf{x}\Delta)$	$f(\mathbf{x}^{(t)} + \mathbf{1}\Delta\mathbf{x})$	$f(\mathbf{x}^{(t)} + \mathbf{2}\Delta\mathbf{x})$
(a)	12	15	18	30
(b)	40	22	12	10
(c)	43	51	60	25
(d)	60	49	44	70

Solution: We follow details of Algorithm 16G.

(a) Reflection ($\lambda = 1$) objective value 18 is neither better than best ensemble value $f(\mathbf{y}^{(1)}) = 13$ nor worse than second worst $f(\mathbf{y}^{(3)}) = 25$. Thus we replace worst current point $\mathbf{y}^{(4)}$ with the one for $\lambda = 1$.

(b) Reflection ($\lambda = 1$) objective value 12 is better than ensemble best $f(\mathbf{y}^{(1)}) = 3$. Thus we expand and try $\lambda = 2$. The resulting value of 10 improves, so we replace worst current point $\mathbf{y}^{(4)}$ with the one for $\lambda = 2$.

(c) Reflection ($\lambda = 1$) objective value 60 is worse than ensemble worst $f(\mathbf{y}^{(4)}) = 50$. Thus we contract and try $\lambda = -\frac{1}{2}$. The resulting value of 43 is better than worst, so we replace $\mathbf{y}^{(4)}$ with the one for $\lambda = -\frac{1}{2}$.

(d) Reflection point ($\lambda = 1$) objective value 44 is better than worst $f(\mathbf{y}^{(4)}) = 50$, but not second worst $f(\mathbf{y}^{(3)}) = 25$. Thus we contract and try $\lambda = \frac{1}{2}$. The resulting value of 49 is better than worst, so we replace $\mathbf{y}^{(4)}$ with the one for $\lambda = \frac{1}{2}$.

Nelder–Mead Shrinking

Sometimes neither the reflection point nor a contraction alternative yields a solution that would much improve the ensemble. This suggests the need for rescaling the entire array of ensemble points.

Principle 16.52 | When reflection and subsequent contraction fails to improve the Nelder–Mead algorithm’s current ensemble, the procedure **shrinks** the whole array toward best point $\mathbf{y}^{(1)}$ by

$$\mathbf{y}^{(i)} \leftarrow \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(i)}) \quad \text{for all } i = 2, \dots, n + 1$$

Figure 16.20 depicts such a shrinking step at $t = 3$ of the PERT maximum likelihood search in Table 16.8. Ensemble points $\mathbf{y}^{(2)}$ and $\mathbf{y}^{(3)}$ are moved halfway to best $\mathbf{y}^{(1)}$ as

$$\begin{aligned} \text{new } \mathbf{y}^{(2)} &\leftarrow \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(2)}) = \frac{1}{2} \left[\begin{pmatrix} 4.25 \\ 2.5 \end{pmatrix} + \begin{pmatrix} 5.25 \\ 2.5 \end{pmatrix} \right] = \begin{pmatrix} 4.75 \\ 2.5 \end{pmatrix} \\ \text{new } \mathbf{y}^{(3)} &\leftarrow \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(3)}) = \frac{1}{2} \left[\begin{pmatrix} 4.25 \\ 2.5 \end{pmatrix} + \begin{pmatrix} 4.625 \\ 2.25 \end{pmatrix} \right] = \begin{pmatrix} 4.438 \\ 2.375 \end{pmatrix} \end{aligned}$$

After renumbering to reflect new objective value sequence, the search proceeds with this more compact ensemble.

EXAMPLE 16.24: SHRINKING IN NELDER–MEAD SEARCH

Suppose that it becomes necessary to apply shrinking Step 7 of Algorithm 16G with current ensemble

$$\mathbf{y}^{(1)} = (2, 3), \quad \mathbf{y}^{(2)} = (-4, 5), \quad \text{and} \quad \mathbf{y}^{(3)} = (8, -1)$$

Compute the new ensemble.

Solution: We apply principle 16.52.

$$\text{new } \mathbf{y}^{(1)} = \mathbf{y}^{(1)} = (2, 3)$$

$$\text{new } \mathbf{y}^{(2)} = \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(2)}) = \frac{1}{2} [(2, 3) + (-4, 5)] = (-1, 4)$$

$$\text{new } \mathbf{y}^{(3)} = \frac{1}{2} (\mathbf{y}^{(1)} + \mathbf{y}^{(3)}) = \frac{1}{2} [(2, 3) + (8, -1)] = (5, 1)$$

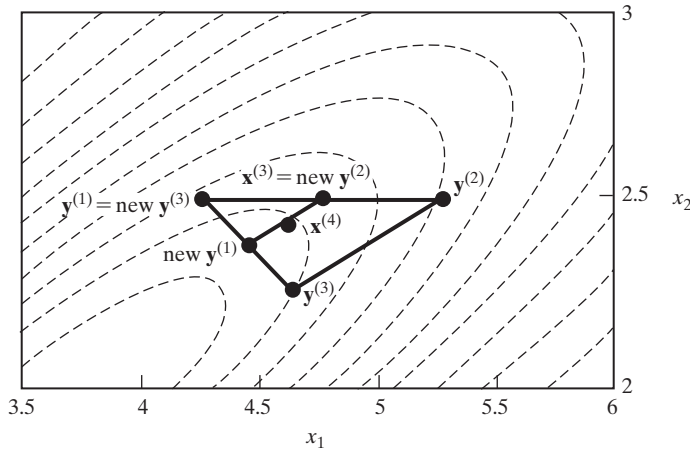


FIGURE 16.20 Nelder–Mead Shrinking in PERT Application

Nelder–Mead Search of PERT Application

Table 16.8 detailed all 20 iterations required in Algorithm 16G search of our maximum likelihood PERT model (16.10). When one of the trial λ values was adopted, it was shown boxed in the table. Otherwise, shrinking principle 16.52 was invoked.

Nelder–Mead search stops when objective function values for points in the ensemble become essentially equal. In Table 16.8 this condition was reached when

$$\sqrt{\frac{1}{n+1} \sum_{i=1}^{n+1} [f(\mathbf{y}^{(i)}) - f(\mathbf{x}^{(t)})]^2} < \epsilon = 0.5$$

Then the best of the ensemble and centroid points is selected as an approximate optimum $\mathbf{x}^* = (4.5, 2.207)$ with $f(\mathbf{x}^*) = 48,520$.

EXERCISES

16-1 A biomedical instrumentation company sells its main product at the rate of 5 units per day. The instrument is manufactured in lots run every few days. It costs the company \$2000 to setup for production of a lot and \$40 per unit per day to hold finished instruments in inventory between runs. The company would like to choose a lot size that minimizes average inventory and setup cost per day assuming that demand occurs smoothly at the given rate.

- ✓ (a) Formulate a 1-variable unconstrained NLP to choose an optimum lot size.
- ✓ (b) Plot the objective function of your model and compute an optimum lot size graphically.

16-2 As part of a study of 911 emergency calls, an analyst wishes to choose the value of parameter α in **exponential** probability density function $d(t) \triangleq \alpha e^{-\alpha t}$ that best fits call interarrival times 80, 10, 14, 26, 40, and 22 minutes.

- (a) Formulate a 1-variable unconstrained NLP to choose a maximum likelihood estimate of α .
- ✓ (b) Plot the objective function of your model and compute an optimal estimate graphically.

16-3 An oil drilling company wishes to locate a supply base somewhere in the jungle area where it is presently exploring for oil. The base will service drilling sites at map coordinates (0, -30), (50, -10), (70, 20), and (30, 50) with helicopter supply runs. The company wishes to choose a location that minimizes the sum of flying distances to the four sites.

- ✓ (a) Formulate an unconstrained NLP to choose an optimum base location.
- ✓ (b) Use class optimization software to compute at least a local optimum starting from coordinates (10, 10).

16-4 Repeat Exercise 16-3, this time minimizing the maximum distance to any drilling site.

16-5 An electronics assembly firm is planning its production staff needs to make a new modem. It

has measured one test worker assembling the unit and observed the following data:

Through unit	2	6	20	25	40
Average time	8.4	5.5	4.2	3.7	3.1

Experience shows that **learning curves**, which describe the ability of a worker to improve his or her productivity as more and more units are produced, often take the form

$$\text{average time} = a (\text{no. units})^b$$

where a and b are empirical constants.

- ✓ (a) Formulate an unconstrained NLP to fit this form to the given data so as to minimize the sum of squared errors.
- ✓ (b) Use class optimization software to compute at least a local optimum for your curve fitting model starting with $a = 15, b = -0.5$.

16-6 The following shows a series of measurements of the height (in inches) of a new genetically engineered tomato plant versus the number of weeks after the plant was replanted outdoors.

Week	1	2	4	6	8	10
Height	9	15	22	33	44	52

Researchers wish to fit this experience with an S-shaped **logistics curve**

$$\text{size} = \frac{k}{1 + e^{a+b(\text{weeks})}}$$

where $k, a,$ and b are empirical parameters.

- (a) Formulate an unconstrained NLP to fit this form to the given data so as to minimize the sum of squared errors.
- ✓ (b) Use class optimization software to compute at least a local optimum for your curve fitting model starting with $k = 50, a = 3, b = -0.3$.

16-7 The university motor pool³ provides a large number of cars n for faculty and staff traveling on university business. Motor pool cars have an

³Based on W. W. Williams and O. S. Fowler (1980), "Minimum Cost Fleet Sizing for a University Motor Pool," *Interfaces* 10:3, 21-27.

average annual cost of f dollars per car for fixed expenses such as depreciation, insurance, and licensing, plus a variable operating cost of v_m cents per mile driven. Those travelers who cannot be accommodated by the pool must drive their personal cars and be reimbursed at v_p cents per mile. Demand varies greatly among times of the year, but an extensive analysis of past travel records has fitted regression equations

$$m(n) \triangleq a_m + b_m n + c_m/n$$

$$p(n) \triangleq a_p + b_p n + c_p/n$$

to the annual numbers of miles that would be driven in motor pool and personal cars as a function of the size of the motor pool available. Formulate a 1-variable unconstrained NLP to compute a minimum total cost motor pool size. Ignore integrality.

16-8 Once a site for a new service facility has been chosen, the limits of its market area must be determined,⁴ along with the corresponding facility size. Assume (i) that the facility is to be located at the center of a circular market area and sized to cover uniform density d calls per unit area out to a radius of r ; (ii) the cost of operating the facility is $[f + c(\text{size})]$, where f is a fixed cost and c a variable cost per unit size; and (iii) transportation costs per call are proportional to (straight-line, one-way) distance from the facility to the customer at t per unit distance. Formulate a 1-variable unconstrained NLP to choose a market area radius that minimizes the facility's average total cost per call. Evaluate any calculus integrals in your objective.

16-9 Renewing highway pavement markings⁵ costs c dollars per mile but reduces social costs from delays, accidents, and other effects of declining marking performance over time. Suppose that new markings yield maximum performance p_{\max} and that performance t days after renewal can be expressed $p(t) \triangleq p_{\max} e^{-\alpha t}$, where α is a constant depending on the durability of the markings. Also assume that each unit of lost performance costs d dollars per day per mile. Formulate

a 1-variable unconstrained NLP to choose a time between pavement renewals to minimize average total daily cost per mile. Evaluate any calculus integrals in your objective.

16-10 The number of potential patrons p_i of a new movie theater complex has been estimated from census data for each of the surrounding counties $i = 1, \dots, 15$. However, the fraction of potential patrons from any i who will actually use the complex varies inversely with its (straight line) distance from the county centroid at coordinates (x_i, y_i) . Formulate a 2-variable NLP to choose a maximum total realized patronage location for the complex.

16-11 Denoting by n_t the number of universities using a textbook through semester t of its availability ($n_0 = 0$), the number of new adoptions in any single semester t can be estimated $(a + bn_{t-1})(m - n_{t-1})$, where a and b are parameters relating to the rapidity of success, and m is the maximum number of universities who will ever adopt.

- (a) Given values of n_t for $t = 1, \dots, 10$, formulate an unconstrained NLP to make a nonlinear least squares fit of the foregoing approximation to these data.
- ✓ (b) Show that an appropriate change of parameters can convert your model into a linear least squares problem.

16-12 Major aircraft parts undergo inspection and overhaul⁶ every t_1 flying hours, and replacement every t_2 . Experience shows the cost of overhauling a particular model of jet engine can be expressed as increasing nonlinear function $a(t_1)^b$ of the overhaul cycle, and operating costs per hour are an increasing nonlinear function $c(t_2)^d$ of the replacement time. Each replacement costs a fixed part cost f plus labor cost $g(t_1)^h$ to identify and fix associated problems. Formulate an unconstrained NLP over t_1 and t_2 to find an overhaul and replacement policy that minimizes total cost per flying hour.

⁴Based on D. Erlenkotter (1989), "The General Optimal Market Area Model," *Annals of Operations Research*, 18, 45–70.

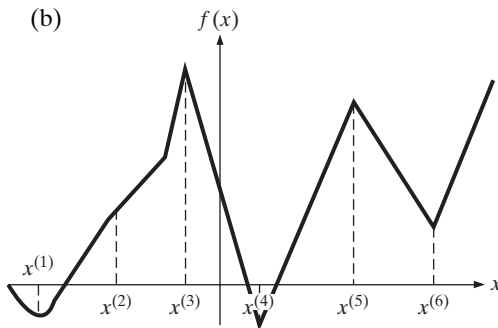
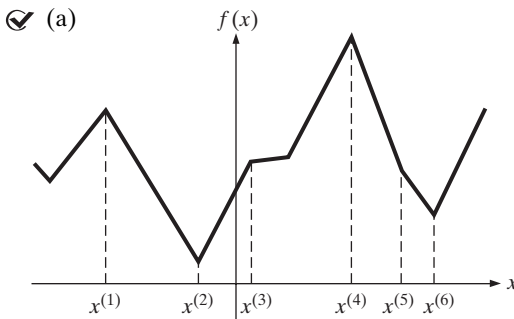
⁵Based on V. Kouskoulas (1988), "An Optimization Model for Pavement Marking Systems," *European Journal of Operational Research*, 33, 298–303.

⁶Based on T. C. E. Cheng (1992), "Optimal Replacement of Ageing Equipment Using Geometric Programming," *International Journal of Production Research*, 30, 2151–2158.

16-13 Determine whether each of the following functions is smooth on the specified domain.

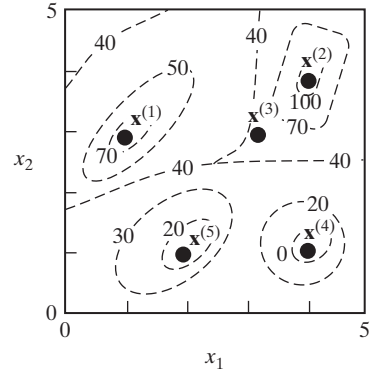
- ✓ (a) $f(x) \triangleq x^4 + 3x - 19$ for all x
- (b) $f(x) \triangleq \min \{2x - 1, 2 - x\}$ for all x
- ✓ (c) $f(x) \triangleq |x - 5|$ for $x > 0$
- (d) $f(x) \triangleq 3x + \ln(x)$ for $x > 0$
- ✓ (e) $f(x_1, x_2) \triangleq x_1 e^{x_2}$ for all x_1, x_2
- (f) $f(x_1, x_2) \triangleq |x_1 + 1| + |x_2 - 3|$ for $x_1, x_2 \geq 0$.

16-14 Each of the following plots shows a function $f(x)$. Determine graphically whether each indicated point is an unconstrained local maximum, an unconstrained global maximum, an unconstrained local minimum, an unconstrained global minimum, or none of the above over the domain depicted.

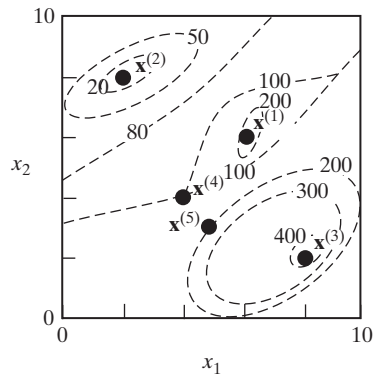


16-15 Each of the following plots shows contours of a smooth function $f(x_1, x_2)$. Determine graphically whether each indicated point is an unconstrained local maximum, an unconstrained global maximum, an unconstrained local minimum, an unconstrained global minimum, or none of the above over the domain depicted.

- ✓ (a) Points $\mathbf{x}^{(1)} = (1, 3)$, $\mathbf{x}^{(2)} = (4, 4)$,
 $\mathbf{x}^{(3)} = (3, 3)$, $\mathbf{x}^{(4)} = (4, 1)$,
 $\mathbf{x}^{(5)} = (2, 1)$



- (b) Points $\mathbf{x}^{(1)} = (6, 6)$, $\mathbf{x}^{(2)} = (2, 8)$,
 $\mathbf{x}^{(3)} = (8, 2)$, $\mathbf{x}^{(4)} = (4, 4)$, $\mathbf{x}^{(5)} = (4, 2)$



16-16 Use golden section Algorithm 16A to find an optimum of the NLP

$$\begin{aligned} \min \quad & 10x + \frac{70}{x} \\ \text{s.t.} \quad & 1 \leq x \leq 10 \end{aligned}$$

to within an error of ± 1 .

16-17 Use golden section Algorithm 16A to find an optimum of the NLP

$$\begin{aligned} \max \quad & 500 - x(x - 20)^3 \\ \text{s.t.} \quad & 0 \leq x \leq 12 \end{aligned}$$

to within an error of ± 1 .

16-18 Suppose that we were given only the lower limit of 1 in the NLP of Exercise 16-16. Apply 3-point pattern Algorithm 16B to compute a corresponding upper limit with which golden section search could begin using each of the following initial step sizes δ .

- ✓ (a) $\delta = 0.5$
- ✓ (b) $\delta = 16$

16-19 Do Exercise 16-18 for the NLP of Exercise 16-17 using $\delta = 2$ and $\delta = 5$.

✔ **16-20** Use quadratic fit Algorithm 16C to compute an optimum for the NLP of Exercise 16-16 within an error tolerance of 2. Start with the 3-point pattern $\{1, 2, 10\}$.

16-21 Use quadratic fit Algorithm 16C to compute an optimum for the NLP of Exercise 16-17 within an error tolerance of 4. Start with 3-point pattern $\{0, 3, 12\}$.

16-22 Consider the 1-variable function $f(x) \triangleq x^3 - 3x^2 + 11x$ at current point $x = 3$.

- ✔ (a) Derive the first-order Taylor approximation to $f(x + \lambda)$.
- ✔ (b) Derive the second-order Taylor approximation to $f(x + \lambda)$.
- ✔ (c) Plot the original function and both Taylor series approximations in the vicinity of x . How accurate do the approximations appear to be? Which is better?

16-23 Do Exercise 16-22 for function $f(x) \triangleq 18x - 20 \ln(x)$ at $x = 16$.

16-24 Consider the 2-variable function $f(x_1, x_2) \triangleq (x_1)^3 - 5x_1x_2 + 6(x_2)^2$ with current point $\mathbf{x} = (0, 2)$ and move direction $\Delta \mathbf{x} = (1, -1)$.

- ✔ (a) Derive the first-order Taylor approximation to $f(\mathbf{x} + \lambda \Delta \mathbf{x})$.
- ✔ (b) Derive the second-order Taylor approximation to $f(\mathbf{x} + \lambda \Delta \mathbf{x})$.
- ✔ (c) Plot the original function and both Taylor series approximations as functions of λ . How accurate do the approximations appear to be? Which is better?

16-25 Do Exercise 16-24 for function $f(x_1, x_2) \triangleq 13x_1 - 6x_1x_2 + 8/x_2$, $\mathbf{x} = (2.1)$ and $\Delta \mathbf{x} = (3.1)$.

16-26 For each of the following unconstrained NLPs, either verify that the given \mathbf{x} is a stationary point of the objective function or give a direction $\Delta \mathbf{x}$ that improves at \mathbf{x} .

- ✔ (a) $\min (x_1)^2 + x_1x_2 - 6x_1 - 8x_2$, $\mathbf{x} = (8, -10)$
- (b) $\max 10(x_1)^2 + 2 \ln(x_2)$, $\mathbf{x} = (1, 2)$
- ✔ (c) $\min 16x_1 - x_1x_2 + 2(x_2)^2$, $\mathbf{x} = (3, 0)$
- (d) $\max x_1x_2 - 8x_1 + 2x_2$, $\mathbf{x} = (-2, 8)$

16-27 For each of the following functions f , use conditions 16.19 to 16.22 to classify the specified

\mathbf{x} as definitely local maximum, possibly local maximum, definitely local minimum, possibly local minimum, and/or definitely neither.

- ✔ (a) $f(x_1, x_2) \triangleq 3(x_1)^2 - x_1x_2 + (x_2)^2 - 11x_1$, $\mathbf{x} = (2, 1)$
- (b) $f(x_1, x_2) \triangleq -(x_1)^2 - 6x_1x_2 - 9(x_2)^2$, $\mathbf{x} = (-3, 1)$
- ✔ (c) $f(x_1, x_2) \triangleq x_1 - x_1x_2 + (x_2)^2$, $\mathbf{x} = (2, 1)$
- (d) $f(x_1, x_2) \triangleq 12x_2 - (x_1)^2 + 3x_1x_2 - 3(x_2)^2$, $\mathbf{x} = (12, 8)$
- ✔ (e) $f(x_1, x_2) \triangleq x_1(x_2)^3$, $\mathbf{x} = (0, 0)$
- (f) $f(x_1, x_2) \triangleq 6x_1 + \ln(x_1) + (x_2)^2$, $\mathbf{x} = (1, 3)$
- ✔ (g) $f(x_1, x_2) \triangleq 2(x_1)^2 + 8x_1x_2 + 8(x_2)^2 - 12x_1 - 24x_2$, $\mathbf{x} = (1, 1)$
- (h) $f(x_1, x_2) \triangleq 4(x_1)^2 + 3/x_2 - 8x_1 + 3x_2$, $\mathbf{x} = (1, 1)$

16-28 Determine whether each of the following functions is convex, concave, both, or neither over the domain specified.

- ✔ (a) $f(x_1, x_2) \triangleq \ln(x_1) + 20 \ln(x_2)$ over $x_1, x_2 > 0$
- (b) $f(x) \triangleq x \sin(x)$ over $x \in [0, 2\pi]$
- ✔ (c) $f(x) \triangleq x(x - 2)^2$ over all $x \geq 0$
- (d) $f(x) \triangleq (x - 8)^4 + 132x$ over all x
- ✔ (e) $f(x_1, \dots, x_5) \triangleq 3x_1 + 11x_2 - x_3 - 8x_5$ over all (x_1, \dots, x_5)
- (f) $f(x_1, x_2) \triangleq 21x_1 + 63x_2$ over $x_1, x_2 \geq 0$
- ✔ (g) $f(x_1, x_2) \triangleq \max\{x_1, x_2\}$ over all x_1, x_2
- (h) $f(x_1, x_2) \triangleq \min\{13x_1 - 2x_2, -(x_2)^2\}$ over all x_1, x_2
- ✔ (i) $f(x_1, x_2) \triangleq 10/\sqrt{x_1} - 40\sqrt{x_2}$ over $x_1, x_2 \geq 0$
- (j) $f(x_1, x_2) \triangleq \ln[-3(x_1)^2 - 9(x_2)^2] - 10/x_2$ over $x_1, x_2 > 0$

16-29 Use convexity/concavity to establish that each of the following solutions \mathbf{x} is either an unconstrained global maximum or an unconstrained global minimum of the f indicated, and explain which.

- ✔ (a) $f(x_1, x_2) \triangleq (x_1 - 5)^2 + x_1x_2 + (x_2 - 7)^2$ at $\mathbf{x} = (2, 6)$
- (b) $f(x_1, x_2) \triangleq 500 - 8(x_1 + 1)^2 - 2(x_2 - 1)^2 + 4x_1x_2$ at $\mathbf{x} = (-1, 0)$

16-30 Consider the unconstrained NLP

$$\max x_1x_2 - 5(x_1 - 2)^4 - 3(x_2 - 5)^4$$

- ☐ (a) Use graphing software to produce a contour map of the objective function for $x_1 \in [1, 4], x_2 \in [2, 8]$.
- ✓ (b) Compute the move direction that would be pursued by gradient search Algorithm 16D at $\mathbf{x}^{(0)} = (1, 3)$.
- ✓ (c) State the line search problem implied by your direction.
- ✓ (d) Solve your line search problem graphically and compute the next search point $\mathbf{x}^{(1)}$.
- ☐ (e) Do two additional iterations of Algorithm 16D to compute $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$.
- ✓ (f) Plot progress of the search on the contour map of part (a).

16-31 Do Exercise 16-30 for the unconstrained NLP

$$\min \frac{1000}{x_1 + x_2} + (x_1 - 4)^2 + (x_2 - 10)^2$$

starting from $\mathbf{x}^{(0)} = (3, 1)$, and plotting $x_1 \in [2, 11], x_2 \in [0, 15]$.

16-32 Return to the unconstrained optimization of Exercise 16-30 starting from $\mathbf{x}^{(0)} = (3, 7)$.

- ✓ (a) Write the second-order Taylor approximation to the objective function at $\mathbf{x}^{(0)}$ for unknown $\Delta \mathbf{x}$ and $\lambda = 1$.
- ✓ (b) Compute the Newton direction $\Delta \mathbf{x}$ at $\mathbf{x}^{(0)}$ and verify that it is a stationary point of your second-order Taylor approximation.
- ✓ (c) Beginning with your Newton direction, complete 2 iterations of Newton's method Algorithm 16E.
- ☐ (d) Plot progress of your search on a contour map like the one of Exercise 16-30(a).

16-33 Do Exercise 16-32 on the NLP of Exercise 16-31 starting from $\mathbf{x}^{(0)} = (13, 1)$.

16-34 Return to the unconstrained optimization of Exercise 16-31 and consider BFGS Algorithm 16F starting at $\mathbf{x}^{(0)} = (2, 3)$.

- ✓ (a) Compute the first direction that would be pursued by Algorithm 16F.

- ✓ (b) Assuming that the optimal step is $\lambda = 0.026$ in that direction, compute the new solution $\mathbf{x}^{(1)}$, the next deflection matrix \mathbf{D} , and the next BFGS search direction $\Delta \mathbf{x}$.
- (c) Verify that your deflection matrix of part (b) satisfies quasi-Newton conditions [16.43] (within roundoff error) and is symmetric.
- (d) Verify algebraically that your direction of part (b) is improving.
- ☐ (e) Plot your first move and second direction on a contour map like that of Exercise 16-31(a).

16-35 Do Exercise 16-34 on the NLP of Exercise 16-31 starting from $\mathbf{x}^{(0)} = (6, 1)$ and using $\lambda = 0.32$ in part (b).

16-36 Consider the unconstrained NLP

$$\min \max \{ 10 - x_1 - x_2, 6 + 6x_1 - 3x_2, 6 - 3x_1 + 6x_2 \}$$

- ✓ (a) Explain why Nelder–Mead search is appropriate for solving this unconstrained optimization.
- ✓ (b) Do 3 iterations (moves) of Nelder–Mead Algorithm 16G, starting from initial ensemble $(5, 0), (10, 5), (5, 5)$.
- (c) Plot the progress of your search through centroids $\mathbf{x}^{(i)}$ and connecting the 3 points of each ensemble with dashed lines.

16-37 Do Exercise 16-36 for the NLP

$$\max \min \{ 20 - x_1 - x_2, 6 + 3x_1 - x_2, 6 - x_1 + 3x_2 \}$$

starting with ensemble $(0, 0), (1, 2), (2, 2)$.

16-38 Compute the Nelder–Mead Algorithm 16G ensemble that would result from applying the shrinking step to each of the following ($\mathbf{y}^{(1)}$ best objective value, etc.).

- ✓ (a) $\mathbf{y}^{(1)} = (1, 2, 1), \mathbf{y}^{(2)} = (5, 4, 5), \mathbf{y}^{(3)} = (3, 2, 7), \mathbf{y}^{(4)} = (7, 2, 7)$
- (b) $\mathbf{y}^{(1)} = (10, 8, 10), \mathbf{y}^{(2)} = (4, 6, 2), \mathbf{y}^{(3)} = (0, 0, 0), \mathbf{y}^{(4)} = (0, 10, 6)$

REFERENCES

- Bazaraa, Mokhtar, Hanif D. Sherali, and C. M. Shetty (2006), *Nonlinear Programming - Theory and Algorithms*, Wiley Interscience, Hoboken, New Jersey.
- Luenberger, David G. and Yinyu Ye (2008), *Linear and Nonlinear Programming*, Springer, New York, New York.
- Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009). *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.

Constrained Nonlinear Programming

In this chapter we introduce models and methods for constrained nonlinear optimization. We begin by modeling a series of real applications, emphasizing differences between linear and nonlinear constraints, convex and nonconvex feasible sets, separable and nonseparable objective functions, and so on. Then, a variety of solution methods are developed, some fairly general purpose and others restricted to special classes on NLPs. Theoretical development draws on Chapters 3, 5, 6, and 16.

17.1 CONSTRAINED NONLINEAR PROGRAMMING MODELS

As usual, we begin our investigation of constrained nonlinear programming with some examples. In this section we formulate three cases, illustrating the broad range on constrained NLP, and in Section 17.2 we present three more, representing some classic special cases. All are based on real application contexts drawn from published reports.

APPLICATION 17.1: BEER BELGE LOCATION ALLOCATION

One common class of nonlinear programs arises from the **location** and customer **allocation** of distribution and service facilities. The task confronted by a large Belgian brewery we will call Beer Belge illustrates.¹ Beer Belge wishes to realign its 17 depots to more efficiently solve its 24,000 customers throughout Belgium. For purposes of this analysis the customers can be aggregated in 650 regions.

Assigning index dimensions

$i \triangleq$ depot number ($i = 1, \dots, 17$)

$j \triangleq$ customer region number ($j = 1, \dots, 650$)

¹Based in part on L. F. Gelders, L. M. Pintelon, and L. N. Van Wassenhove (1987), "A Location-Allocation Problem in a Large Belgian Brewery," *European Journal of Operational Research*, 28, 196–206.

company analysts can determine from maps and past experience

- $h_j \triangleq$ x -coordinate of the center of customer region j
- $k_j \triangleq$ y -coordinate of the center of customer region j
- $d_j \triangleq$ number of delivery trips required to region j per year

We wish to choose locations for the depots and allocate customer region demand to minimize total travel cost.

Beer Belge Location-Allocation Model

As with all such **location-allocation** problems, decision variables in a model of Beer Belge’s case must settle two distinct but related questions. First, we need to know where the depots will be located. Then we must allocate the trips required in various regions among the depots. The following address both:

- $x_i \triangleq$ x -coordinate of depot i ’s location
- $y_i \triangleq$ y -coordinate of depot i ’s location
- $w_{i,j} \triangleq$ number of trips per year from depot i to customer region j

To obtain an objective function, we adopt the common assumption that roundtrip travel costs from depot i to customer region j is proportional to the straight-line (Euclidean) distance between their locations. Then, minimizing total transportation costs of the location-allocation amounts to

$$\min \sum_i \sum_j (\text{number of trips from } i \text{ to } j) (\text{distance from } i \text{ to } j)$$

Using symbolic constants and decision variables defined above, we obtain

$$\min \sum_{i=1}^{17} \sum_{j=1}^{650} w_{i,j} \sqrt{(x_i - h_j)^2 + (y_i - k_j)^2}$$

To complete a model, we must add constraints assuring that all required trips to each region are made. The result is

$$\begin{aligned} \min \quad & \sum_{i=1}^{17} \sum_{j=1}^{650} w_{i,j} \sqrt{(x_i - h_j)^2 + (y_i - k_j)^2} && \text{(total travel)} \\ \text{st} \quad & \sum_{i=1}^{17} w_{i,j} = d_j \quad j = 1, \dots, 650 && \text{(allocate trips)} \\ & w_{i,j} \geq 0 \quad i = 1, \dots, 17; \quad j = 1, \dots, 650 \end{aligned} \tag{17.1}$$

EXAMPLE 17.1: FORMULATING LOCATION-ALLOCATION MODELS

Two airstrips are to be constructed in the jungle to service three remote oil fields. The first oil field requires 25 tons of supplies per month. The second, which is 75 miles east and 330 miles north of the first, requires 14 tons. The third, which is 225 miles west and 40 miles south of the first, needs 34 tons per month. Formulate

a location-allocation model to locate and operate the airstrips to minimize the ton-miles flown per month.

Solution: Coordinates in this problem may be introduced by taking the first oil field as $(0, 0)$. Then the second is $(-75, 330)$ and the third is $(225, -40)$.

Decisions involve both the location of the airstrips and allocation of workload between the strips. Thus we employ decision variables $(x_1, y_1) \triangleq$ coordinates of the first airstrip, $(x_2, y_2) \triangleq$ coordinates of the second, and $w_{i,j} \triangleq$ tons flown from airstrip i to oil field j . The resulting model is

$$\begin{aligned} \min \quad & w_{1,1}\sqrt{(x_1 - 0)^2 + (y_1 - 0)^2} + w_{1,2}\sqrt{(x_1 + 75)^2 + (y_1 - 330)^2} \quad (\text{ton-miles}) \\ & + w_{1,3}\sqrt{(x_1 - 225)^2 + (y_1 + 40)^2} + w_{2,1}\sqrt{(x_2 - 0)^2 + (y_2 - 0)^2} \\ & + w_{2,2}\sqrt{(x_2 + 75)^2 + (y_2 - 330)^2} + w_{2,3}\sqrt{(x_2 - 225)^2 + (y_2 + 40)^2} \\ \text{s.t.} \quad & w_{1,1} + w_{2,1} = 25 \quad (\text{field 1}) \\ & w_{1,2} + w_{2,2} = 14 \quad (\text{field 2}) \\ & w_{1,3} + w_{2,3} = 34 \quad (\text{field 3}) \\ & w_{i,j} \geq 0 \quad i = 1, 2; \quad j = 1, 2, 3 \end{aligned}$$

Here the objective function totals flight distances times tons shipped, and constraints allocate needed shipments among airstrips.

Linearly Constrained Nonlinear Programs

Location-allocation model (17.1) illustrates the many large-scale nonlinear programs having all constraints linear. Only its objective function renders the model an NLP.

Linearly constrained nonlinear programs form an important special class because much of the tractability of linear programs extends to cases with only the objective function nonlinear. Fortunately, linearly constrained NLPs are also very common.

Principle 17.1 Most large-scale nonlinear programs that can be solved effectively have all or nearly all their constraints linear.

APPLICATION 17.2: TEXACO GASOLINE BLENDING

Beginning as early as the 1950s, oil companies have used mathematical programming to help plan gasoline blending at refineries. Texaco is no exception.²

Figure 17.1 sketches gasoline processing in refineries. Crude oil is first distilled into a range of materials from light ones such as gasoline to much heavier ones such as fuel oil. The “straight-run” gasoline produced in distilling is not nearly enough to meet market demands economically. Thus most of the distillates that are heavier or lighter than gasoline are re-formed and cracked to produce other gasoline forms. A final processing stage combines some additives with these various stocks of

²Based on C. W. DeWitt, L. S. Lasdon, A. D. Waren, D. A. Brenner, and S. A. Melhem (1989), “OMEGA: An Improved Gasoline Blending System for Texaco,” *Interfaces*, 19:1, 85–101.

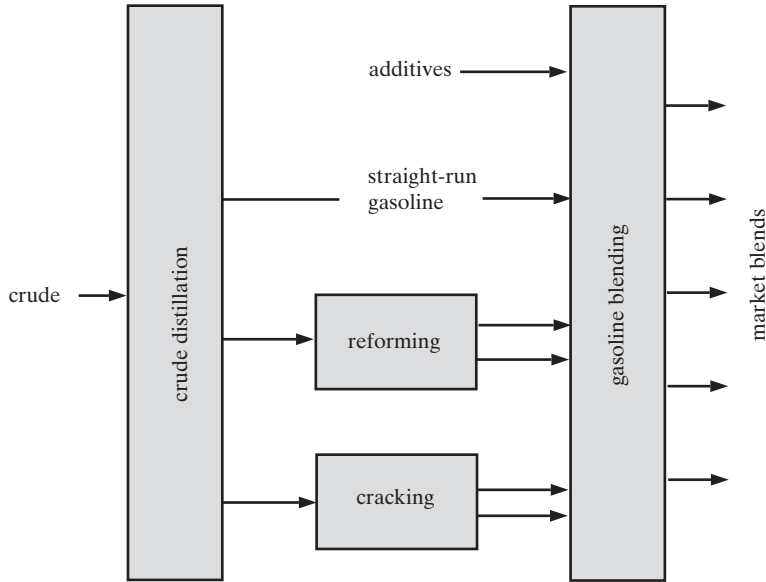


FIGURE 17.1 Gasoline Refinery Flow

gasoline to produce market blends (e.g., premium unleaded, regular unleaded) with suitable quality indexes (e.g., octane, lead content, sulfur content, volatility).

Typical operations plan blending for a month at a time. Using index dimensions

- $i \triangleq$ gasoline or additive stock number ($i = 1, \dots, m$)
- $j \triangleq$ market blend number ($j = 1, \dots, n$)
- $k \triangleq$ quality index number ($k = 1, \dots, h$)

We will assume that the following constants are known as each optimization begins:

- $p_j \triangleq$ estimated selling price per unit of output blend j
- $r_j \triangleq$ quantity of output blend j required
- $s_i \triangleq$ quantity of input stock i available
- $v_i \triangleq$ estimated cost per unit of input stock i
- $a_{i,k} \triangleq$ k th quality index of input stock i
- $\ell_{j,k} \triangleq$ lowest acceptable level of quality index k in output blend j
- $u_{j,k} \triangleq$ highest acceptable level of quality index k in output blend j

We want to find a maximum profit (sales income minus stock costs) plan to meet all blending requirements with available stocks.

Texaco Gasoline Blending Model

Clearly, the decisions in gasoline blending involve how much of which stocks to use. Thus we will develop a model with decision variables

$$x_{i,j} \triangleq \text{quantity of input stock } i \text{ used in output blend } j$$

In accord with principle [17.1](#), much of the model will be linear. We begin with the objective to maximize sales income minus stock cost:

$$\max \sum_{i=1}^m \sum_{j=1}^n (p_j - v_i)x_{i,j} \quad (\text{profit})$$

Linear constraints enforce stock availabilities and output demands:

$$\begin{aligned} \sum_{j=1}^n x_{i,j} &\leq s_i & i = 1, \dots, m \\ \sum_{i=1}^m x_{i,j} &\geq r_j & j = 1, \dots, n \end{aligned}$$

Most of the blending constraints also take the linear form of the blending LPs in Section 4.2. That is, we require the average level of each quality index to fall between upper and lower limits for each output blend with constraints

$$\ell_{j,k} \leq \frac{\sum_{i=1}^m a_{i,k}x_{i,j}}{\sum_{i=1}^m x_{i,j}} \leq u_{j,k} \quad j = 1, \dots, n: \text{ linear } k$$

What introduces nonlinearity in gasoline blending are two classes of quality indexes that do not combine linearly. The volatility quality measures perform logarithmically to produce constraints

$$\ell_{j,k} \leq \ln \left(\frac{\sum_{i=1}^m a_{i,k}x_{i,j}}{\sum_{i=1}^m x_{i,j}} \right) \leq u_{j,k} \quad j = 1, \dots, n: \text{ volatility } k$$

Octane measures are even more complex. Typical schemes use known constants $b_{i,k}$, $c_{i,k}$, $d_{i,k}$, and $e_{i,k}$ to specify octane limits with fourth-order expressions

$$\begin{aligned} \ell_{j,k} &\leq \frac{\sum_{i=1}^m b_{i,k}x_{i,j}}{\sum_{i=1}^m x_{i,j}} + \frac{\sum_{i=1}^m c_{i,k}(x_{i,j})^2}{(\sum_{i=1}^m x_{i,j})^2} + \frac{\sum_{i=1}^m d_{i,k}(x_{i,j})^3}{(\sum_{i=1}^m x_{i,j})^3} \\ &+ \frac{\sum_{i=1}^m e_{i,k}(x_{i,j})^4}{(\sum_{i=1}^m x_{i,j})^4} \leq u_{j,k} \quad j = 1, \dots, n; \text{ octane } k \end{aligned}$$

Collecting all the above and adding variable-type constraints completes the full gasoline blending model:

$$\begin{aligned} \max \quad & \sum_{i=1}^m \sum_{j=1}^n (p_j - v_i)x_{i,j} && (\text{profit}) \\ \text{s.t.} \quad & \sum_{j=1}^n x_{i,j} \leq s_i && i = 1, \dots, m \quad (\text{availability}) \\ & \sum_{i=1}^m x_{i,j} \geq r_j && j = 1, \dots, n \quad (\text{demand}) \end{aligned}$$

$$\ell_{j,k} \leq \frac{\sum_{i=1}^m a_{i,k} x_{i,j}}{\sum_{i=1}^m x_{i,j}} \leq u_{j,k} \quad j = 1, \dots, n; \text{ linear } k \quad (\text{blend 1})$$

$$\ell_{j,k} \leq \ln \left(\frac{\sum_{i=1}^m a_{i,k} x_{i,j}}{\sum_{i=1}^m x_{i,j}} \right) \leq u_{j,k} \quad j = 1, \dots, n; \text{ volatility } k \quad (\text{blend 2}) \quad (17.2)$$

$$\ell_{j,k} \leq \frac{\sum_{i=1}^m b_{i,k} x_{i,j}}{\sum_{i=1}^m x_{i,j}} + \frac{\sum_{i=1}^m c_{i,k} (x_{i,j})^2}{(\sum_{i=1}^m x_{i,j})^2} + \frac{\sum_{i=1}^m d_{i,k} (x_{i,j})^3}{(\sum_{i=1}^m x_{i,j})^3} + \frac{\sum_{i=1}^m e_{i,k} (x_{i,j})^4}{(\sum_{i=1}^m x_{i,j})^4} \leq u_{j,k} \quad j = 1, \dots, n; \text{ octane } k \quad (\text{blend 3})$$

$$x_{i,j} \geq 0 \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

This time it is the last two systems of main constraints that make the model a nonlinear program. All other elements are linear.

Engineering Design Models

The mostly linear location-allocation [model (17.1)] and gasoline blending [model (17.2)] nonlinear programs treated so far are typical of large-scale applications. Still, the NLPs arising in **engineering design** often have quite a different character.

Principle 17.2 | Optimal engineering design of structures and processes frequently leads to constrained nonlinear programs with relatively few variables but highly nonlinear constraints and objective functions.

APPLICATION 17.3: OXYGEN SYSTEM ENGINEERING DESIGN

To illustrate the smaller but more nonlinear models arising in engineering, we consider the design of the oxygen production system for a basic oxygen furnace in the steel industry.³ Figure 17.2(a) shows the main components of the system. The oxygen plant produces at a constant rate, with output compressed and stored in a tank. The furnace then uses the oxygen in a cycle like the one depicted in Figure 17.2(b). Relatively low demand level d_1 applies during the first t_1 minutes of the cycle, followed by a burst to demand d_2 from time t_1 through t_2 .

We seek a minimum cost sizing of the system components. In particular, we wish to choose values for 4 decision variables:

- $x_1 \triangleq$ production rate of the oxygen plant
- $x_2 \triangleq$ pressure in the storage tank
- $x_3 \triangleq$ compressor power
- $x_4 \triangleq$ storage tank volume

Also, physical limitations require that the storage tank pressure be at least p_0 .

³Based on F. C. Jen, C. C. Pegels, and T. M. DuPuis (1967), "Optimal Capacities of Production Facilities," *Management Science*, 14, B573–B580.

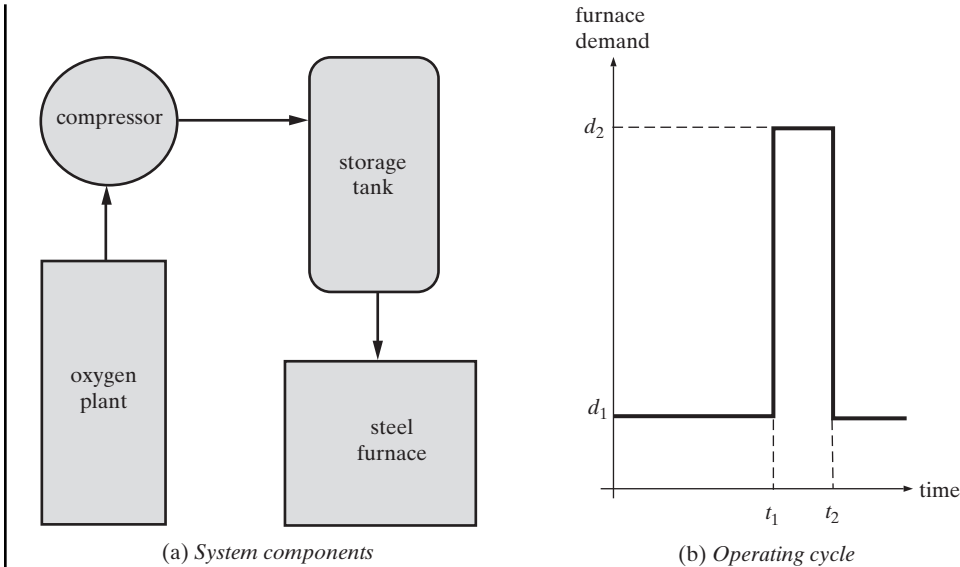


FIGURE 17.2 Oxygen System Engineering Design Application

Oxygen System Engineering Design Model

The procurement and operation costs of this oxygen system has 4 parts:

$$\left(\begin{array}{c} \text{total} \\ \text{cost} \end{array} \right) = \left(\begin{array}{c} \text{oxygen} \\ \text{plant cost} \end{array} \right) + \left(\begin{array}{c} \text{compressor} \\ \text{cost} \end{array} \right) + \left(\begin{array}{c} \text{storage tank} \\ \text{cost} \end{array} \right) + \left(\begin{array}{c} \text{electrical} \\ \text{power cost} \end{array} \right)$$

Using prior experience with similar systems, engineers in this application estimate oxygen plant cost to be linear in the production rate as

$$\text{oxygen plant cost} = 61.8 + 5.72 (\text{production rate})$$

Compressor costs grow with compressor power in the nonlinear function

$$\text{compressor cost} = 0.0175 (\text{power})^{0.85}$$

Storage vessel costs perform similarly in the required volume

$$\text{storage tank costs} = 0.0094 (\text{volume})^{0.75}$$

Finally, electricity for the compressor is proportional to the power and time of operation as

$$\text{electrical power cost} = 0.006 (\text{time of operation}) (\text{compressor power})$$

One constraint of the model comes from the requirement that the oxygen production rate outputs at least what is required over the cycle, that is,

$$t_2 x_1 \geq d_1 t_1 + d_2 (t_2 - t_1)$$

Also, the minimum pressure p_0 imposes bound

$$x_2 \geq p_0$$

The remainder of the constraints enforce the physical relations among the decision variables. First, the power of the compressor is related to the pressure that it must generate in the storage tank. In particular, the compressor power must address the maximum stored inventory $(d_2 - x_1)(t_2 - t_1)$ just before the burst part of the operating cycle. Including standard temperature and gas constants gives the constraint

$$x_3 = 36.25 \frac{(d_2 - x_1)(t_2 - t_1)}{t_1} \ln \left(\frac{x_2}{p_0} \right)$$

Finally, we must relate storage volume to the pressure needed to keep the maximum required inventory. Using appropriate temperature and gas constants gives

$$x_4 = 348,300 \frac{(d_2 - x_1)(t_2 - t_1)}{x_2}$$

Collecting all the above and adding variable-type constraints, we obtain the following NLP model of our oxygen system design problem:

$$\begin{aligned}
 \min \quad & 61.8 + 5.72x_1 + 0.0175(x_3)^{0.85} + 0.0094(x_4)^{0.75} && \text{(total cost)} \\
 & + 0.006t_1x_3 && \\
 \text{s.t.} \quad & t_2x_1 \geq d_1t_1 + d_2(t_2 - t_1) && \text{(demand)} \\
 & x_2 \geq p_0 && \text{(pressure)} \\
 & x_3 = 36.25 \frac{(d_2 - x_1)(t_2 - t_1)}{t_1} \ln \left(\frac{x_2}{p_0} \right) && \begin{matrix} \text{(power vs.)} \\ \text{(pressure)} \end{matrix} \\
 & x_4 = 348,300 \frac{(d_2 - x_1)(t_2 - t_1)}{x_2} && \begin{matrix} \text{(volume vs.)} \\ \text{(pressure)} \end{matrix} \\
 & x_1, x_2, x_3, x_4 \geq 0 &&
 \end{aligned} \tag{17.3}$$

With constants

$$d_1 = 2.5, \quad d_2 = 40, \quad t_1 = 0.6, \quad t_2 = 1.0, \quad p_0 = 200$$

an optimal design uses production rate $x_1^* = 17.5$, storage pressure $x_2^* = 473.7$, compressor power $x_3^* = 468.8$, and storage volume $x_4^* = 6618$ for total cost approximately 173.7.

EXAMPLE 17.2: FORMULATING ENGINEERING DESIGN MODELS

A closed cylindrical tank is being designed to carry at least 20 cubic feet of chemicals. Metal for the top and sides costs \$2 per square foot, but the heavier metal of the base costs \$8 per square foot. Also, the height of the tank can be no more than twice its diameter to keep it from being top heavy. Formulate a constrained nonlinear program to find a design of minimum cost.

Solution: The decision variables the designer must choose are

$$\begin{aligned}
 x_1 &\triangleq \text{diameter of the tank} \\
 x_2 &\triangleq \text{height of the tank}
 \end{aligned}$$

Then, we obtain the NLP

$$\begin{aligned} \min \quad & 2\left(\pi x_1 x_2 + \pi \frac{(x_1)^2}{4}\right) + 8\left(\pi \frac{(x_1)^2}{4}\right) && \text{(metal cost)} \\ \text{s.t.} \quad & \pi \frac{(x_1)^2}{4} x_2 \geq 20 && \text{(volume)} \\ & x_2 \leq 2x_1 && \text{(height-to-diameter ratio)} \\ & x_1 \geq 0, x_2 \geq 0 && \end{aligned}$$

The objective function minimizes metal cost for the sides and top at \$2 per square foot, plus cost for the base at \$8. One main constraint makes the tank have the required volume, and the other enforces the height-to-diameter ratio limit.

17.2 CONVEX, SEPARABLE, QUADRATIC, AND POSYNOMIAL GEOMETRIC PROGRAMMING SPECIAL NLP FORMS

Linear programming is one especially tractable special case of constrained non-linear programming, but it is not the only one. Several other common forms have special characteristics that can be exploited in search algorithms.

In this section we define and illustrate models in four of these unusually tractable classes: **convex programs**, **separable programs**, **quadratic programs**, and **posynomial geometric programs**. Then, in Sections 17.7–17.10 we illustrate how their special properties can be exploited. As usual, the models presented are drawn from real application contexts, usually cases described in published accounts.

APPLICATION 17.4: PFIZER OPTIMAL LOT SIZING

One common application context yielding NLPs with special structure arises in managing **inventories** and choosing manufacturing **lot sizes**. Work at Pfizer, a large manufacturer of pharmaceuticals,⁴ offers a good example.

The production of pharmaceuticals involves a series of fermentation and organic synthesis steps done in large tanks holding several thousand gallons. Each “campaign” or **lot** of a product consists of one or more “batches” processed serially.

The main issue to be decided is how many batches should be included in a campaign or lot of each product. Large changeover times and costs are incurred each time a new campaign starts because of the need to clean and reconfigure equipment carefully for the next product. These changeover burdens must be balanced against the cost of holding inventories built up during campaigns. In particular, rigorous quality control requires that all batches of a campaign be held at the end of each manufacturing step until every one is ready for the next step.

Our specific (fictitious) version of this lot sizing problem will consider products $j = 1, \dots, 4$, each having steps $k = 1, 2$ of production ($k = 0$ refers to raw materials). Table 17.1 details values of the associated input parameters

⁴Based on P. P. Kleutghen and J. C. McGee (1985), “Development and Implementation of an Integrated Inventory Management Program at Pfizer Pharmaceuticals,” *Interfaces*, 15:1, 69–87.

TABLE 17.1 Pfizer Lot Sizing Application Data

Product Number, j	Annual Demand, d_j	Changeover Time (weeks)		Process Time (weeks)		Value (\$000)		
		$t_{j,1}$	$t_{j,2}$	$p_{j,1}$	$p_{j,2}$	$v_{j,0}$	$v_{j,1}$	$v_{j,2}$
1	150	0.5	0.7	1.5	3.2	10	14	27
2	220	1.3	2.0	4.0	1.5	50	70	110
3	55	0.3	0.2	2.5	4.2	18	29	40
4	90	0.9	1.8	2.0	3.5	43	69	178

- $d_j \triangleq$ number of batches of product j demanded annually
- $t_{j,k} \triangleq$ changeover time per campaign of product j at step k (in weeks)
- $p_{j,k} \triangleq$ process time per batch of product j in step k (in weeks)
- $v_{j,k} \triangleq$ value per batch of product j at the end of step k (in thousands of dollars)

We will also assume that 3000 total production weeks are available to manufacture the 4 products, that holding inventory cost $\frac{1}{2}\%$ per week of the product value, and that changeover activities cost $c = \$12,000$ thousand per week.

Pfizer Optimal Lot Sizing Model

To model our version of the Pfizer application, introduce decision variables

$$x_j \triangleq \text{number of batches of product } j \text{ in each campaign or lot}$$

To begin, notice that the implied number of campaigns per year is

$$\frac{\text{annual demand}}{\text{campaign size}} = \frac{d_j}{x_j}$$

Then the total annual cost objective function can be expressed as

$$\min \sum_{j=1}^4 \frac{d_j}{x_j} \left[\left(\begin{array}{c} \text{changeover costs} \\ \text{per campaign of } j \end{array} \right) + \left(\begin{array}{c} \text{holding costs per} \\ \text{campaign of } j \end{array} \right) \right] \tag{17.4}$$

Changeover costs for product j are easily expressed as

$$\left(\begin{array}{c} \text{changeover costs} \\ \text{per campaign of } j \end{array} \right) = c \sum_{k=1}^2 t_{j,k}$$

To compute the corresponding inventory holding costs, consider the cycle illustrated in Figure 17.3. Each campaign of product j begins with x_j batches valued at raw material cost $v_{j,0}$. Over the $x_j p_{j,1}$ weeks it takes all batches to complete the first step of processing, their value increases to $x_j v_{j,1}$. Then the next $x_j p_{j,2}$ weeks raise the value to $x_j v_{j,2}$. Finally, the finished pharmaceutical is distributed to customers over the $52/(d_j/x_j)$ weeks until the next campaign is completed. Inventory cost per campaign is then $\frac{1}{2}\%$ of the area under the implied inventory value curve.

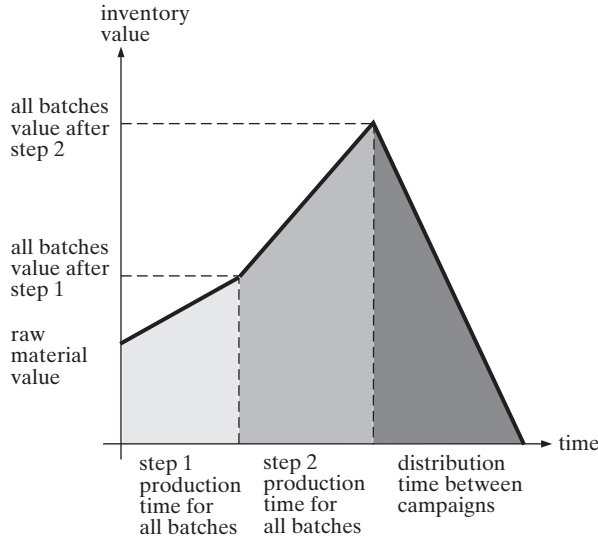


FIGURE 17.3 Pfizer Lot Sizing Application Inventory Cycle

Substituting in expression (17.4) gives the complete objective function

$$\min \sum_{j=1}^4 \frac{d_j}{x_j} \left[c \sum_{k=1}^2 t_{j,k} + 0.005 \left(\sum_{k=1}^2 \frac{1}{2} p_{j,k} (v_{j,k-1} + v_{j,k}) (x_j)^2 + \frac{52v_{j,2}}{2d_j} (x_j)^2 \right) \right]$$

The only main constraint in this model will enforce the limit on production time available. Again summing per campaign, we have

$$\sum_{j=1}^4 \frac{d_j}{x_j} \left[\left(\text{changeover time} \right)_{\text{per campaign of } i} + \left(\text{production time} \right)_{\text{per campaign of } j} \right] \leq \left(\text{time available} \right)$$

or

$$\sum_{j=1}^4 \frac{d_j}{x_j} \left(\sum_{k=1}^2 t_{j,k} + \sum_{k=1}^2 p_{j,k} x_j \right) \leq 3000 \tag{17.5}$$

Substituting parameter values of Table 17.1 and simplifying produces the complete model

$$\begin{aligned} \min \quad & 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} \quad (\text{total cost}) \\ & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} \tag{17.6} \\ \text{s.t.} \quad & \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} \leq 221.5 \quad (\text{production time}) \\ & x_1, \dots, x_4 \geq 0 \end{aligned}$$

An optimal plan runs

$$x_1^* = 7.161, \quad x_2^* = 5.665, \quad x_3^* = 2.911, \quad x_4^* = 4.135 \quad (17.7)$$

campaigns per year of the 4 products. Total annual cost will be approximately \$6,837,000.

Convex Programs

Lot sizing model (17.6) is special, in part, because it forms a convex program.

Definition 17.3 A constrained nonlinear program in functional form

$$\begin{aligned} & \max \text{ or } \min f(\mathbf{x}) \\ \text{s.t. } & g_i(\mathbf{x}) \begin{cases} \geq \\ \leq \\ = \end{cases} b_i \quad i = 1, \dots, m \end{aligned}$$

is a **convex program** if f is concave for a maximize or convex for a minimize, each g_i of a \geq constraint is concave, each g_i of a \leq constraint is convex, and each g_i of an $=$ constraint is linear.

In Section 16.4 we defined convex and concave functions (definition [16.23]) and showed their importance in characterizing the most tractable objective functions. Convex programs extend these ideas to constraints. Every \geq constraint should be concave (after collecting terms involving the decision variables on the left-hand side), every \leq constraint convex, and every $=$ constraint linear.

To see that lot sizing model (17.6) satisfies these requirements, we first consider its objective function:

$$\begin{aligned} \min f(\mathbf{x}) \triangleq & 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} \\ & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} \end{aligned} \quad (17.8)$$

Notice that it is a sum of terms

$$\alpha_j x_j + \frac{\beta_j}{x_j}$$

with positive constants α_j and β_j . The first such terms are linear and so convex (principle [16.28]). But linear functions are also concave, which makes reciprocal terms convex for $x_j \geq 0$ (principle [16.32]). It follows that the objective function is a sum of convex functions, and so itself convex (principle [16.29]). This is just what is needed in definition [17.3].

Now consider constraints

$$\begin{aligned}
 g_1(\mathbf{x}) &\triangleq \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} \leq 221.5 \\
 g_2(\mathbf{x}) &\triangleq x_1 \geq 0 \\
 g_3(\mathbf{x}) &\triangleq x_2 \geq 0 \\
 g_4(\mathbf{x}) &\triangleq x_3 \geq 0 \\
 g_5(\mathbf{x}) &\triangleq x_4 \geq 0
 \end{aligned} \tag{17.9}$$

Definition [17.3](#) requires g_1 convex and g_2, \dots, g_5 concave. Function g_1 conforms for much the same reason as the objective: It is the positive-weighted sum of reciprocals of nonnegative linear functions. Other functions g_2, \dots, g_5 are concave because they are linear.

EXAMPLE 17.3: RECOGNIZING CONVEX PROGRAMS

Determine whether each of the following mathematical programs is a convex program.

(a) $\max \quad 3w_1 - w_2 + 8 \ln(w_1)$
 s.t. $4(w_1)^2 - w_1w_2 + (w_2)^2 \leq 100$
 $w_1 + w_2 = 4$
 $w_1, w_2 \geq 0$

(b) $\min \quad 3w_1 - w_2 + 8 \ln(w_1)$
 s.t. $(w_1)^2 + (w_2)^2 \geq 10$
 $w_1 + w_2 = 4$
 $w_1, w_2 \geq 0$

(c) $\max \quad w_1 + 7w_2$
 s.t. $w_1w_2 \leq 14$
 $(w_1)^2 + (w_2)^2 = 40$
 $w_1, w_2 \geq 0$

(d) $\min \quad w_1 + 7w_2$
 s.t. $w_1 + w_2 \leq 14$
 $w_1 - w_2 \geq 0$
 $2w_1 + 5w_2 = 18$
 $w_1, w_2 \geq 0$

Solution: We apply definition [17.3](#).

(a) The objective function of this model is concave because it is the sum of a linear function and $8 \ln(w_1)$, which has negative second derivative (principle [16.27](#)). Also, the first main constraint function is convex because the Hessian matrix

$$\nabla^2 f(\mathbf{w}) = \begin{pmatrix} 8 & -1 \\ -1 & 2 \end{pmatrix}$$

is positive definite (principle [16.27](#)). Since all 3 other constraints are linear, the model is a convex program. It maximizes a concave objective, subject to a convex \leq constraints, a linear $=$ constraint, and concave (because linear) \geq constraints.

(b) This NLP is not a convex program because its objective function, which is the concave one of part (a), is inappropriate for a minimize. Also, the first main constraint

$$(w_1)^2 + (w_2)^2 \geq 10$$

is convex in a \geq form.

(c) This model is also not a convex program. Its first main constraint involves function $g_1(w_1, w_2) = w_1 w_2$, which is neither convex nor concave. Its Hessian matrix is

$$\nabla^2 g_1(\mathbf{w}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Furthermore, the second main constraint is a nonlinear equality.

(d) This model is a linear program and so a convex program. Linear objectives and constraints, which are both convex and concave (principle [16.28](#)), fulfill all requirements of definition [17.3](#).

Special Tractability of Convex Programs

To appreciate the special tractability of convex programs, recall the characterization in Section 3.4 of the models most convenient for improving search. The best objective functions are those that are unimodal (definition [16.9](#)), and principle [16.24](#) has already established that maximizing a concave function, or minimizing a convex, produces an objective with that desirable property.

For constraints we would like the implied feasible set to be convex (definition [3.27](#)). Requirements for a convex program assure exactly that.

Principle 17.4 The feasible set defined by constraints

$$g_i(\mathbf{x}) \begin{cases} \geq \\ \leq \\ = \end{cases} b_i \quad i = 1, \dots, m$$

is convex if each g_i of a \geq constraint is concave, each g_i of a \leq constraint is convex, and each g_i of an $=$ constraint is linear.

We can see why property [17.4](#) is true by considering a single convex constraint

$$g(\mathbf{x}) \leq b$$

and two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ that satisfy it. If the corresponding feasible set is to be convex, every point along the line segment between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ must also satisfy the constraint, that is (definition [3.29](#)), every point representable as

$$\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})$$

for some $\lambda \in [0, 1]$. Using

$$\begin{aligned}g(\mathbf{x}^{(1)}) &\leq b \\g(\mathbf{x}^{(2)}) &\leq b\end{aligned}$$

we may multiply the first by $(1 - \lambda)$, the second by λ , and sum to conclude that

$$(1 - \lambda)g(\mathbf{x}^{(1)}) + (\lambda)g(\mathbf{x}^{(2)}) \leq (1 - \lambda)b + (\lambda)b$$

which simplifies to

$$g(\mathbf{x}^{(1)}) + \lambda(g(\mathbf{x}^{(2)}) - g(\mathbf{x}^{(1)})) \leq b$$

With g convex, the left side of this expression is at most $g(\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)}))$ (definition [16.23](#)). Thus

$$g(\mathbf{x}^{(1)} + \lambda(\mathbf{x}^{(2)} - \mathbf{x}^{(1)})) \leq b$$

which shows that the line segment point for λ is feasible are required. A similar argument demonstrates that feasible sets for concave \geq constraints are convex, and we already know that linear constraints yield convex feasible sets (principle [3.32](#)).

Combining the convexity of objective functions for convex programs with the convexity of their feasible sets leads to their main convenience for improving search.

Principle 17.5 Every local optimum of a convex program is a global optimum.

Improving search procedures that produce locally optimal solutions automatically yield global optima.

Separable Programs

Besides being a convex program, the objective and constraint functions of Pfizer lot sizing model (17.6) also have a special separable property.

Definition 17.6 Function $s(\mathbf{x})$ is **separable** if it can be expressed as the sum

$$s(x_1, \dots, x_n) \triangleq \sum_{j=1}^n s_j(x_j)$$

of single-variable functions $s_1(x_1), \dots, s_n(x_n)$.

That is, a function is separable if it is the sum of 1-variable functions of its arguments.

To see that Pfizer objective function (17.8) is separable, consider the 1-variable functions

$$\begin{aligned}f_1(x_1) &\triangleq 66.21x_1 + \frac{2160}{x_1} \\f_2(x_2) &\triangleq 426.8x_2 + \frac{8712}{x_2} \\f_3(x_3) &\triangleq 61.20x_3 + \frac{330}{x_3} \\f_4(x_4) &\triangleq 268.1x_4 + \frac{2916}{x_4}\end{aligned}$$

Clearly, the full objective $f(\mathbf{x})$ separates into a sum of these functions of single decision variables.

Similar definitions establish that constraint functions $g_1(\mathbf{x})$ through $g_5(\mathbf{x})$ are separable. For example,

$$g_1(\mathbf{x}) = g_{1,1}(x_1) + g_{1,2}(x_2) + g_{1,3}(x_3) + g_{1,4}(x_4)$$

where $g_{1,1}(x_1) = 180/x_1$, $g_{1,2}(x_2) = 726/x_2$, $g_{1,3}(x_3) = 27.5/x_3$, and $g_{1,4}(x_4) = 243/x_4$.

When both the objective function and all constraint functions are separable in this way, we term an NLP a separable program.

Definition 17.7 A constrained nonlinear program in functional form

$$\max \text{ or } \min f(\mathbf{x})$$

$$\text{s.t. } g_i(\mathbf{x}) \begin{cases} \geq \\ \leq \\ = \end{cases} b_i \quad i = 1, \dots, m$$

is a **separable program** if f and every g_i is separable.

Pfizer model (17.6) provides an example.

EXAMPLE 17.4: RECOGNIZING SEPARABLE PROGRAMS

Return to the NLPs of Example 17.3, and determine whether each is a separable program.

Solution: We apply definition [17.7](#).

(a) The objective function and most constraints of this model are separable. However, it fails the definition of a separable program because the first main constraint contains the term w_1w_2 , which involves both variables.

(b) This model is a separable program. Each objective and constraint function can be expressed as a sum of separate functions, one involving only w_1 and the other only w_2 .

(c) Again, the objective and most constraint functions of this NLP are separable. Still, the first main constraint is not, because it includes the term w_1w_2 . Thus the model is not a separable program.

(d) This model is a linear program. By definition, linear functions involve sums of constant multiples of the decision variables. Thus all are separable, and every linear program is a separable program.

Special Tractability of Separable Programs

Separable programs, where objective and constraint functions are sums of 1-variable functions, have much in common with linear programs where the corresponding

functions are limited to scalar multiples of the decision variables. This relationship can be exploited by **piecewise-linear approximation** of each component function as suggested in Figure 17.4. That is, each of the single-variable components of the objective and constraint functions is approximated by a series of linear (straight-line) segments. Then, if the separable program also conforms to convex program definition [17.3], the approximation can be solved by linear programming.

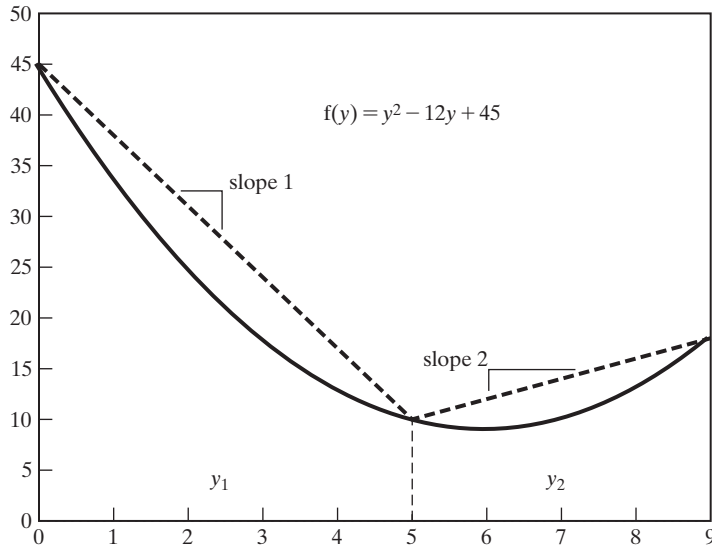


FIGURE 17.4 Piecewise-Linear Approximation of a One-Variable Function

Principle 17.8 Separable convex nonlinear programs can be addressed by linear programming through piecewise-linear approximation of the objective and constraint functions.

Details are provided in Section 17.9.

APPLICATION 17.5: QUADRATIC PORTFOLIO MANAGEMENT

We can illustrate another important class of constrained nonlinear programs by returning to an application setting that we encountered earlier in Chapter 8: **finance**. Financial managers are constantly planning and controlling market decisions, and a wide array of nonlinear programs have been employed to help.

We will consider a simple case of **portfolio management**—dividing investment funds to maximize return and minimize risk. Our manager, whom we will call Barney Backroom, must decide how to split the available funds among three classes of investments: common stocks, money markets, and corporate bonds. Table 17.2 shows the experience of the past 6 years on which Barney will base his decisions. He would like to average an 11% return on investments while accepting minimum risk.

TABLE 17.2 Return Experience for Portfolio Application

Category	Percent Returns for Year:						Average
	1	2	3	4	5	6	
Stocks	22.24	16.16	5.27	15.46	20.62	-0.42	13.22
Money market	9.64	7.06	7.68	8.26	8.55	8.26	8.24
Bonds	10.08	8.16	8.46	9.18	9.26	9.06	9.03

Quadratic Portfolio Management Model

The obvious decision variables for a model of this problem are

- $x_1 \triangleq$ fraction of the portfolio invested in common stocks
- $x_2 \triangleq$ fraction of the portfolio invested in money markets
- $x_3 \triangleq$ fraction of the portfolio invested in corporate bonds

Then one main constraint forces all funds to be invested:

$$x_1 + x_2 + x_3 = 1$$

One reasonable assumption about returns is that each class will achieve the average experienced in the years of Table 17.2. Then the goal of returning 11% can be expressed as

$$13.22x_1 + 8.24x_2 + 9.03x_3 \geq 11$$

The more difficult issue is how to model variability of return. One measure is the **variance**—the average squared deviation from the mean. If the three classes of investments varied independently, the variance of the overall return would be simply the sum of the variances of each class. However, financial markets in various commodities tend to interact. Thus a workable model should include the **covariances** relating movement in the different categories of investment.

Given a series of n observations like those in Table 17.2, covariances can be estimated as

$$v_{i,j} \triangleq \text{estimated covariance between categories } i \text{ and } j$$

$$= \frac{1}{n} \sum_{t=1}^n \begin{pmatrix} i \text{ value} \\ \text{in period } t \end{pmatrix} \begin{pmatrix} j \text{ value} \\ \text{in period } t \end{pmatrix} - \frac{1}{n^2} \left[\sum_{t=1}^n \begin{pmatrix} i \text{ value} \\ \text{in period } t \end{pmatrix} \right] \left[\sum_{t=1}^n \begin{pmatrix} j \text{ value} \\ \text{in period } t \end{pmatrix} \right]$$

Then the variance of the total return is approximately

$$\begin{aligned} \text{variance of return} &= \sum_{i=1}^n \sum_{j=1}^n v_{i,j} x_i x_j \\ &= \mathbf{xVx} \end{aligned}$$

for \mathbf{V} is the matrix of $v_{i,j}$.

Table 17.2's data for our example yields

$$\mathbf{V} = \begin{pmatrix} 66.51 & 2.61 & 2.18 \\ 2.61 & 0.63 & 0.48 \\ 2.18 & 0.48 & 0.38 \end{pmatrix}$$

Thus we may combine with the constraints above to obtain the model

$$\begin{aligned}
 \min \quad & 66.51(x_1)^2 + 2(2.61)x_1x_2 + 2(2.18)x_1x_3 + 0.63(x_2)^2 \text{ (variance)} \\
 & + 2(0.48)x_2x_3 + 0.38(x_3)^2 \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 = 1 \qquad \qquad \qquad \text{(invest 100\%)} \quad (17.10) \\
 & 13.22x_1 + 8.24x_2 + 9.03x_3 \geq 11 \qquad \qquad \qquad \text{(return)} \\
 & x_1, x_2, x_3 \geq 0
 \end{aligned}$$

An optimal portfolio invests a fraction $x_1^* = 0.47$ of funds in common stocks, $x_2^* = 0$ of funds in money markets, and $x_3^* = 0.53$ in corporate bonds to produce a minimum variance of 15.895.

Quadratic Programs Defined

Model (17.10) illustrates the special case of a quadratic program.

Definition 17.9 A constrained nonlinear program is a **quadratic program** if its objective function is quadratic, that is,

$$f(\mathbf{x}) \triangleq \sum_j c_j x_j + \sum_i \sum_j q_{i,j} x_i x_j = \mathbf{c} \cdot \mathbf{x} + \mathbf{x} \mathbf{Q} \mathbf{x}$$

and all constraints are linear.

Portfolio model (17.10) clearly qualifies. All 5 constraints are linear, and the objective function involves only squares and products of 2 variables. In the matrix format of definition [17.9](#), its objective function has

$$\mathbf{c} = \mathbf{0}, \quad \mathbf{Q} = \mathbf{V}$$

EXAMPLE 17.5: RECOGNIZING QUADRATIC PROGRAMS

Determine whether each of the following NLPs is a quadratic program. For those that are, also place the objective function in matrix format $\mathbf{c} \cdot \mathbf{w} + \mathbf{w} \mathbf{Q} \mathbf{w}$.

(a) $\max \quad 3w_1 - 5w_2 + 12(w_1)^2 + 8w_1w_2 + (w_2)^2$

s.t. $w_1 + w_2 = 9$

$w_1, w_2 \geq 0$

(b) $\min \quad w_1w_2w_3$

s.t. $(w_1)^2 + (w_2)^2 \leq 25$

(c) $\min \quad 5w_1 + 19w_2$

s.t. $w_1 + w_2 = 9$

$w_1, w_2 \geq 0$

Solution: We apply definition [17.9](#).

(a) This model is a quadratic program because its objective function involves only second-order terms, and its constraints are linear. Here

$$\mathbf{c} = \begin{pmatrix} 3 \\ -5 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} 12 & 4 \\ 4 & 1 \end{pmatrix}$$

Notice that the coefficient 8 on cross-product w_1w_2 is split so that

$$q_{1,2}w_1w_2 + q_{2,1}w_1w_2 = 4w_1w_2 + 4w_1w_2 = 8w_1w_2$$

(b) This model is not a quadratic program. Its objective involves a 3-way product, which is not quadratic, and its constraint is not linear.

(c) This model is a linear program and so quadratic. Objective function elements

$$\mathbf{c} = \begin{pmatrix} 5 \\ 19 \end{pmatrix}, \quad \mathbf{Q} = \mathbf{0}$$

Special Tractability of Quadratic Programs

Being nonlinear in only the objective function, and only quadratic there, quadratic programs have much in common with LPs. In particular, powerful dual and complementary slackness properties make possible efficient specialized algorithms for many cases. Section 17.7 provides details.

APPLICATION 17.6: COFFERDAM DESIGN

To illustrate another special type of constrained NLP arising often in engineering design applications, we consider the planning of a cofferdam.⁵ Cofferdams are used to block streams temporarily while construction is in progress. Figure 17.5 illustrates a common design. Each “cycle” of the dam consists of a large steel cylinder filled with soil and joined to the next with curved connecting plates.

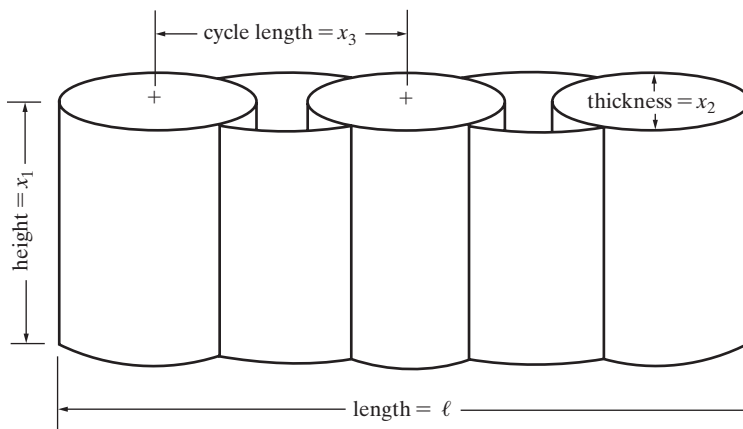


FIGURE 17.5 Cofferdam Application Structure

Given

$$\begin{aligned} \ell &\triangleq \text{total dam length (in feet)} \\ t &\triangleq \text{design life of the dam (in days)} \end{aligned}$$

⁵Based on F. Neghabat and R. M. Stark (1972), “A Cofferdam Design Optimization,” *Mathematical Programming*, 3, 263–275.

and other characteristics of site and materials, we wish to determine a minimum cost design. Main decision variables are

$$\begin{aligned}x_1 &\triangleq \text{height of the dam (in feet)} \\x_2 &\triangleq \text{average thickness of the dam (in feet)} \\x_3 &\triangleq \text{length of a cycle of the dam (in feet)}\end{aligned}$$

Cofferdam Application Model

To develop a model, we begin with the various elements of cost. Filling cost is roughly proportional to the dam volume. Using a cost of \$0.21 per cubic foot, this produces

$$\begin{aligned}\text{fill cost} &\approx 0.21(\text{length})(\text{height})(\text{thickness}) \\&= 0.21\ell x_1 x_2\end{aligned}$$

Similarly, steel cost depends on the area of the dam's front and back, plus that of the two cylinder sides passing through the dam in each cycle. Pricing steel at \$2.28 per square foot, we have

$$\begin{aligned}\text{steel cost} &\approx 2.28 \left[2(\text{length})(\text{height}) + 2 \left(\frac{\text{length}}{\text{cycle length}} \right) (\text{height})(\text{thickness}) \right] \\&= 4.56\ell x_1 + 4.56\ell \frac{x_1 x_2}{x_3}\end{aligned}$$

A very low dam would minimize construction cost, but flood risk must also be considered. Analysis of prior experience suggests that flood cost can be approximated as

$$\begin{aligned}\text{flood cost} &\approx (\text{cost per flood}) \left(\frac{\text{design life}}{\text{empirical}} \right) \\&= 40,000 \frac{t}{x_4}\end{aligned}$$

where each flood costs \$40,000, and x_4 is an intermediate decision variable related to dam height by

$$x_4 + 33.3 \leq 0.8x_1$$

Other constraints of the model come from possible failure modes. One possibility is slipping on the river bottom. This can be prevented if

$$1.0425(\text{height}) \leq \text{thickness} \quad \text{or} \quad 1.0425x_1 \leq x_2$$

The other main considerations are the tension stresses at cycle joints. These require that

$$(\text{height})(\text{cycle length}) \leq 2857 \quad \text{or} \quad x_1 x_3 \leq 2857$$

Collecting all the above, simplifying to make all constraint right-hand sides = 1, and assuming a dam of length $\ell = 800$ feet that should last $t = 365$ days, our cofferdam design task reduces to the following constrained NLP model:

$$\begin{aligned}
 \min \quad & 168x_1x_2 + 3648x_1 + 3648\frac{x_1x_2}{x_3} + \frac{1.46 \times 10^7}{x_4} \quad (\text{cost}) \\
 \text{s.t.} \quad & \frac{1.25x_4}{x_1} + \frac{41.625}{x_1} \leq 1 \quad (\text{empirical}) \\
 & \frac{1.0425x_1}{x_2} \leq 1 \quad (\text{slipping}) \\
 & 0.00035x_1x_3 \leq 1 \quad (\text{tension}) \\
 & x_1, x_2, x_3, x_4 > 0
 \end{aligned} \tag{17.11}$$

An optimal design has height $x_1^* = 62.65$ feet, average thickness $x_2^* = 65.32$ feet, cycle length $x_3^* = 45.60$ feet, intermediate variable $x_4^* = 16.82$, and total cost \$2.111 million.

Posynomial Geometric Programs

The objective and constraint functions of cofferdam model (17.11) have a special posynomial form.

Definition 17.10 Function $p(\mathbf{x})$ is a **posynomial** if it can be expressed

$$p(x_1, \dots, x_n) \triangleq \sum_k d_k \left(\prod_{j=1}^n (x_j)^{a_{k,j}} \right)$$

for given $d_k > 0$ and exponents $a_{k,j}$ of arbitrary sign.

For example, objective function

$$f(x_1, x_2, x_3, x_4) \triangleq 168x_1x_2 + 3648x_1 + 3648\frac{x_1x_2}{x_3} + \frac{1.46 \times 10^7}{x_4}$$

of model (17.11) is a posynomial with

$$\begin{aligned}
 d_1 &= 168, & d_2 &= 3648, & d_3 &= 3648, & d_4 &= 1.46 \times 10^7 \\
 a_{1,1} &= 1, & a_{1,2} &= 1, & a_{1,3} &= 0, & a_{1,4} &= 0 \\
 a_{2,1} &= 1, & a_{2,2} &= 0, & a_{2,3} &= 0, & a_{2,4} &= 0 \\
 a_{3,1} &= 1, & a_{3,2} &= 1, & a_{3,3} &= -1, & a_{3,4} &= 0 \\
 a_{4,1} &= 0, & a_{4,2} &= 0, & a_{4,3} &= 0, & a_{4,4} &= -1
 \end{aligned} \tag{17.12}$$

Notice that powers $a_{k,j}$ can have any sign, but coefficients d_k must be positive. Thus

$$h(x_1, x_2, x_3) \triangleq 13(x_1)^2(x_3) + 29(x_2)^{0.534}(x_3)^{-0.451}$$

is a posynomial, but the variation

$$h(x_1, x_2, x_3) \triangleq 13(x_1)^2(x_3) - 29(x_2)^{0.534}(x_3)^{-0.451}$$

is not because $d_2 = -29 < 0$.

Posynomial geometric programs are NLPs over posynomial functions and positive variables.

Definition 17.11 | An NLP is a **posynomial geometric program** if it can be expressed in the form

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 1 \quad i = 1, \dots, m \\ & \mathbf{x} > \mathbf{0} \end{aligned}$$

where f and all g_i are posynomial functions of \mathbf{x} .

With posynomial terms written out, the format is

$$\begin{aligned} \min \quad & \sum_{k \in K_0} d_k \prod_{j=1}^n (x_j)^{a_{k,j}} \\ \text{s.t.} \quad & \sum_{k \in K_i} d_k \prod_{j=1}^n (x_j)^{a_{k,j}} \leq 1 \quad i = 1, \dots, m \\ & x_j > 0 \quad j = 1, \dots, n \end{aligned} \quad (17.13)$$

where nonoverlapping the K_i index terms k in various posynomials.

Notice that we allow only a minimize objective function form and \leq constraints. Also, we have only positive-valued variables, and positive right-hand sides have been divided through to produce 1's. Usual conversions to reverse directions for other cases fail in the geometric programming case because they change the sign of some d_k and thus destroy the posynomial property.

Cofferdam model (17.11) satisfies all these conditions. In detail format (17.13), model (17.12) shows coefficients for objective function terms in $K_0 = \{1, 2, 3, 4\}$. Those for constraint sets $K_1 = \{5, 6\}$, $K_2 = \{7\}$, and $K_3 = \{8\}$ are

$$\begin{aligned} d_5 &= 1.25, & d_6 &= 41.625, & d_7 &= 1.0425, & d_8 &= 0.00035 \\ a_{5,1} &= -1, & a_{5,2} &= 0, & a_{5,3} &= 0, & a_{5,4} &= 1 \\ a_{6,1} &= -1, & a_{6,2} &= 0, & a_{6,3} &= 0, & a_{6,4} &= 0 \\ a_{7,1} &= 1, & a_{7,2} &= -1, & a_{7,3} &= 0, & a_{7,4} &= 0 \\ a_{8,1} &= 1, & a_{8,2} &= 0, & a_{8,3} &= 1, & a_{8,4} &= 0 \end{aligned}$$

EXAMPLE 17.6: RECOGNIZING POSYNOMIAL GEOMETRIC PROGRAMS

Determine whether each of the following NLPs is a posynomial geometric program. For those that are, also detail coefficients d_k and $a_{k,j}$ in standard form (17.13).

$$\begin{aligned}
 \text{(a) min } & 144 \frac{w_1}{\sqrt{w_2}} + 6w_3 \\
 & 19w_1 + (w_2)^2 \leq w_3 \\
 & w_1 w_2 w_3 \leq 44 \\
 & w_1, w_2, w_3 > 0 \\
 \text{(b) max } & 144 \frac{w_1}{\sqrt{w_2}} + 6w_3 \\
 & 19w_1 - (w_2)^2 \leq w_3 \\
 & w_1 w_2 w_3 \geq 44 \\
 & w_1, w_2, w_3 > 0
 \end{aligned}$$

Solution: We apply definitions [17.10] and [17.11].

(a) Dividing both constraints by their right-hand sides places this model in posynomial geometric program format (17.13). Coefficients are

$$\begin{aligned}
 K_0 &= \{1, 2\}, & K_1 &= \{3, 4\}, & K_2 &= \{5\} \\
 d_1 &= 144, & d_2 &= 6, & d_3 &= 19, & d_4 &= 1, & d_5 &= \frac{1}{44} \\
 a_{1,1} &= 1, & a_{1,2} &= -0.5, & a_{1,3} &= 0 \\
 a_{2,1} &= 0, & a_{2,2} &= 0, & a_{2,3} &= 1 \\
 a_{3,1} &= 1, & a_{3,2} &= 0, & a_{3,3} &= -1 \\
 a_{4,1} &= 0, & a_{4,2} &= 2, & a_{4,3} &= -1 \\
 a_{5,1} &= 1, & a_{5,2} &= 1, & a_{5,3} &= 1
 \end{aligned}$$

(b) This model is not a posynomial geometric program for several reasons. First, its objective maximizes a posynomial, and definition [17.11] requires a minimize. Similarly, the second main constraint has \geq form, not the \leq as appropriate for geometric programs. Finally, the first main constraint function has a negative coefficient, so it is not a posynomial.

Special Tractability of Posynomial Geometric Programs

Posynomial functions need not be convex, and thus geometric programs [17.11] are often not convex programs. For example,

$$h(x_1, x_2) \triangleq (x_1)^2 x_2 + 7x_2 \tag{17.14}$$

has Hessian matrix at $\mathbf{x} = (1, 1)$

$$\nabla^2 h(1, 1) = \begin{pmatrix} 2 & 2 \\ 2 & 0 \end{pmatrix}$$

This matrix is not positive semidefinite, and thus (principle [16.27]) f is not convex over even $\mathbf{x} > \mathbf{0}$.

Still, posynomial geometric programs can be made convex with a suitable change of variables.

Principle 17.12 | Posynomial geometric programs convert to convex programs when original variables x_j are replaced by $z_j \triangleq \ln(x_j)$.

For example, in the case of function (17.14), substituting $z_j \triangleq \ln(x_j)$ or $x_j = e^{z_j}$ produces

$$h(z_1, z_2) \triangleq (e^{z_1})^2 (e^{z_2}) + 7(e^{z_2}) = e^{2z_1+z_2} + 7(e^{z_2})$$

Both exponential terms are convex under composition principle [16.31], so their sum is also convex.

Notice the role that details of definition [17.11] play in making this transformation work. First, a logarithmic transformation would be impossible if any x_j could be zero or negative. Also, the transformation could produce a mixed-sign sum of convex terms if all coefficients c_k were not positive in definition [17.10]. Finally, convex constraint functions would not be appropriate for a convex program (definition [17.3]) unless all constraints were of \leq form.

Further transformations of posynomial geometric programs can lead to even greater tractability. Section 17.10 provides details.

17.3 LAGRANGE MULTIPLIER METHODS

If we can see how to view a constrained nonlinear program as one with only = constraints (i.e., no inequalities of any form), calculus methods predating most of the numerical search techniques of this book can sometimes be applied to find an optimal solution. We briefly explore such **Lagrange multiplier techniques** in this section. Also, Lagrangian ideas will be seen to motivate more general approaches in later sections of the chapter. (See also Lagrangian large-scale methods in Section 13.2.)

Reducing to Equality Form

Lagrange multiplier solution techniques are most easily applied to models in equality constrained format.

Principle 17.13 | Lagrange multiplier solution techniques address NLPs in pure equality form

$$\begin{array}{ll} \min \text{ or } \max & f(\mathbf{x}) \\ \text{s.t.} & g_i(\mathbf{x}) = b_i \quad \text{for all } i = 1, \dots, m \end{array}$$

That is, they consider only a set of constraints assumed active, which can be taken as '='s.

There is one equality for each original equality and one for each active inequality. Notice that not even variable-type inequalities such as nonnegativity constraints are allowed.

To see how such models can arise, consider the Pfizer lot sizing model of Section 17.2:

$$\begin{aligned}
 \min \quad & f(\mathbf{x}) \triangleq 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} \quad (\text{total cost}) \\
 & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} \\
 \text{s.t.} \quad & \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} \leq 221.5 \quad (\text{production time}) \\
 & x_1, \dots, x_4 \geq 0
 \end{aligned} \tag{17.15}$$

If none of the constraints are active at an optimal \mathbf{x}^* , then that solution will be an unconstrained optimum of the objective function alone. Since we have shown in Section 17.2 that the objective function is convex, we may compute \mathbf{x}^* by finding a stationary point (principle 16.22), that is, a point where all partial derivatives = 0. Then, we can solve x_1^* as

$$\frac{\partial f}{\partial x_1} = 66.21 - \frac{2160}{(x_1)^2} = 0 \quad \text{so that} \quad x_1^* = \sqrt{\frac{2160}{66.21}} = 5.712$$

Similarly, unconstrained $x_2^* = 4.518$, $x_3^* = 2.322$, and $x_4^* = 3.298$.

Checking the main constraint of (17.15) gives

$$\frac{180}{5.712} + \frac{726}{4.518} + \frac{27.5}{2.322} + \frac{243}{3.298} = 277.7 \neq 221.5$$

Thus our unconstrained solution is infeasible, and the main constraint must be treated as active at a true optimum. Assuming that nonnegativity constraints will remain inactive, we may deal with the model in pure equality form

$$\begin{aligned}
 \min \quad & f(\mathbf{x}) \triangleq 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} \quad (\text{total cost}) \\
 & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} \\
 \text{s.t.} \quad & \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} = 221.5 \quad (\text{production time})
 \end{aligned} \tag{17.16}$$

Lagrangian Function and Lagrange Multipliers

Lagrangian techniques begin with conversion of equality-constrained model 17.13 to an unconstrained form by weighting constraints in the objective function with **Lagrange multipliers**, v_i . The result is a **Lagrangian function** of both \mathbf{x} and \mathbf{v} .

Definition 17.14 | The Lagrangian function associated with a nonlinear program over equality constraints $g_1(\mathbf{x}) = b_1, \dots, g_m(\mathbf{x}) = b_m$ is

$$L(\mathbf{x}, \mathbf{v}) \triangleq f(\mathbf{x}) + \sum_{i=1}^m v_i [b_i - g_i(\mathbf{x})]$$

where v_i is the Lagrange multiplier for constraint i .

For example, the Lagrangian function for Pfizer equality form (17.16) is

$$L(x_1, x_2, x_3, x_4, \nu) \triangleq 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} + \nu \left(221.5 - \frac{180}{x_1} - \frac{726}{x_2} - \frac{27.5}{x_3} - \frac{243}{x_4} \right) \quad (17.17)$$

Notice what happens when we form the Lagrangian. We have relaxed the given constrained model into unconstrained form. Still, the Lagrangian function value coincides with the original objective function value at every feasible point because

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \sum_{i=1}^m \nu_i [b_i - g_i(\mathbf{x})] = f(\mathbf{x}) + \sum_{i=1}^m \nu_i(0) = f(\mathbf{x}) \quad (17.18)$$

EXAMPLE 17.7: FORMING THE LAGRANGIAN FUNCTION

Consider the equality-constrained nonlinear program

$$\begin{aligned} \min \quad & 6(x_1)^2 + 4(x_2)^2 + (x_3)^2 \\ \text{s.t.} \quad & 24x_1 + 24x_2 = 360 \\ & x_3 = 1 \end{aligned}$$

Form the corresponding Lagrangian function.

Solution: Here $m = 2$. Following format [17.14], we form the Lagrangian by rolling the 2 equality constraints into the objective function with Lagrange multiplier ν_1 and ν_2 . The result is

$$L(x_1, x_2, x_3, \nu_1, \nu_2) \triangleq 6(x_1)^2 + 4(x_2)^2 + (x_3)^2 + \nu_1(360 - 24x_1 - 24x_2) + \nu_2(1 - x_3)$$

Stationary Points of the Lagrangian Function

Think now about stationary points of the Lagrangian [i.e., points where gradient $\nabla L(\mathbf{x}^*, \mathbf{v}^*) = \mathbf{0}$]. Components of gradient $\nabla L(\mathbf{x}, \mathbf{v})$ are partial derivatives with respect to the \mathbf{x} and \mathbf{v} parts of a solution, respectively. Setting both $= \mathbf{0}$ produces the key **stationary-point conditions**.

Principle 17.15 Solution $(\mathbf{x}^*, \mathbf{v}^*)$ is a stationary point of Lagrangian function $L(\mathbf{x}, \mathbf{v})$ if it satisfies

$$\sum_i \nabla g_i(\mathbf{x}^*) \nu_i^* = \nabla f(\mathbf{x}^*) \quad \text{or} \quad \sum_i \frac{\partial g_i}{\partial x_j} \nu_i = \frac{\partial f}{\partial x_j} \quad \text{for all } j$$

and

$$g_i(\mathbf{x}^*) = b_i \quad \text{for all } i$$

EXAMPLE 17.8: FORMING STATIONARY CONDITIONS FOR LAGRANGIANS

Return to the model and Lagrangian function of Example 17.7. State the corresponding stationary-point conditions on x_1^* , x_2^* , x_3^* , v_1^* , and v_2^* .

Solution: We construct conditions [17.15](#) for Lagrangian function

$$L(x_1, x_2, x_3, v_1, v_2) \triangleq 6(x_1)^2 + 4(x_2)^2 + (x_3)^2 + v_1(360 - 24x_1 - 24x_2) + v_2(1 - x_3)$$

Gradients

$$\nabla f(\mathbf{x}) = (12x_1, 8x_2, 2x_3)$$

$$\nabla g_1(\mathbf{x}) = (24, 24, 0)$$

$$\nabla g_2(\mathbf{x}) = (0, 0, 1)$$

Thus the required conditions are

$$\begin{aligned} 24v_1^* &= 12x_1^* \\ 24v_1^* &= 8x_2^* \\ &+ 1v_2^* = 2x_3^* \\ 24x_1^* + 24x_2^* &= 360 \\ 1x_3^* &= 1 \end{aligned}$$

Lagrangian Stationary Points and the Original Model

The value of Lagrangian functions [17.14](#) in solving equality-constrained NLPs [17.13](#) becomes apparent upon careful examination of Lagrangian stationary conditions [17.15](#). We originally formed the Lagrangian to obtain a relaxed model that matches the original objective function at feasible points. For any fixed choice \mathbf{v} of Lagrange multipliers, an unconstrained optimum \mathbf{x}^* of that relaxed model must be a stationary point (principle [16.19](#)). This is exactly what the first part of conditions [17.15](#) demand.

But the second part of [17.15](#) requires any stationary point of the Lagrangian to satisfy all constraints of the original model. Thus if the \mathbf{x}^* components of a stationary point can be shown to optimize the relaxation with multipliers fixed at the \mathbf{v}^* of a stationary point, then that \mathbf{x}^* solves a relaxed form, achieves feasibility, and has the same objective value as the original $f(\mathbf{x}^*)$. It must solve the constrained model.

Principle 17.16 | If $(\mathbf{x}^*, \mathbf{v}^*)$ is a stationary point of Lagrangian function $L(\mathbf{x}, \mathbf{v})$ and \mathbf{x}^* is an unconstrained optimum of $L(\mathbf{x}, \mathbf{v}^*)$, then \mathbf{x}^* is an optimum of the corresponding equality-constrained NLP [17.13](#).

Lagrange Multiplier Procedure

The Lagrangian approach to solving equality-constrained NLPs exploits sufficient optimality condition [17.16]. Specifically, we:

1. Reduce the given model to pure equality form [17.13].
2. Form Lagrangian function [17.14].
3. Solve conditions [17.15] for a stationary point of the Lagrangian function.
4. Try to establish that the \mathbf{x} part of the stationary point is optimal for the Lagrangian with $\mathbf{v} = \mathbf{v}^*$ and thus (principle [17.16]) optimal for the original model.

Lagrangian (17.17) shows the result of steps 1 and 2 for our Pfizer lot sizing application. Solving for a stationary point, we first compute

$$\begin{aligned}
 \frac{\partial L}{\partial x_1} &= 66.21 - \frac{2160}{(x_1)^2} + \frac{180v}{(x_1)^2} = 0 & \text{or} & \quad x_1^* = \sqrt{\frac{2160 - 180v^*}{66.21}} \\
 \frac{\partial L}{\partial x_2} &= 426.8 - \frac{8712}{(x_2)^2} + \frac{726v}{(x_2)^2} = 0 & \text{or} & \quad x_2^* = \sqrt{\frac{8712 - 726v^*}{426.8}} \\
 \frac{\partial L}{\partial x_3} &= 61.20 - \frac{330}{(x_3)^2} + \frac{27.5v}{(x_3)^2} = 0 & \text{or} & \quad x_3^* = \sqrt{\frac{330 - 27.5v^*}{61.20}} \\
 \frac{\partial L}{\partial x_4} &= 268.1 - \frac{2916}{(x_4)^2} + \frac{243v}{(x_4)^2} = 0 & \text{or} & \quad x_4^* = \sqrt{\frac{2916 - 243v^*}{268.1}}
 \end{aligned} \tag{17.19}$$

Then setting

$$\frac{\partial L}{\partial v} = 221.5 - \frac{180}{x_1} - \frac{726}{x_2} - \frac{27.5}{x_3} - \frac{243}{x_4} = 0$$

and substituting gives

$$221.5 = \frac{180}{\sqrt{\frac{2160 - 180v^*}{66.21}}} + \frac{726}{\sqrt{\frac{8712 - 726v^*}{426.8}}} + \frac{27.5}{\sqrt{\frac{330 - 27.5v^*}{61.20}}} + \frac{243}{\sqrt{\frac{2916 - 243v^*}{268.1}}}$$

Now taking advantage of the fact that changeover costs in the objective function of model (17.16) are multiples (by \$12,000 per week) of changeover times, we may factor

$$221.5 = \frac{1}{\sqrt{12 - v^*}} \left(\frac{180}{\sqrt{\frac{180}{66.21}}} + \frac{726}{\sqrt{\frac{726}{426.8}}} + \frac{27.5}{\sqrt{\frac{27.5}{61.20}}} + \frac{243}{\sqrt{\frac{243}{268.1}}} \right)$$

and solve

$$\begin{aligned}
 v^* &= 12 - \frac{1}{221.5} \left[\sqrt{180(66.21)} + \sqrt{726(426.8)} + \sqrt{27.5(61.20)} + \sqrt{243(268.1)} \right]^2 \\
 &= -6.865
 \end{aligned}$$

Finally, substituting in (17.19) gives $x_1^* = 7.161$, $x_2^* = 5.665$, $x_3^* = 2.911$, and $x_4^* = 4.135$.

We know that this \mathbf{x}^* part of the stationary point solution yields an optimum because it agrees with values reported in Section 17.2. To complete step 4 of the Lagrangian approach and apply principle [17.16](#), however, we must examine the Lagrangian objective with $\nu = \nu^* = -6.865$:

$$\begin{aligned} L(x_1, x_2, x_3, x_4 - 6.865) &\triangleq 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} + 61.20x_3 + \frac{330}{x_3} \\ &+ 268.1x_4 + \frac{2916}{x_4} - 6.865 \left(221.5 - \frac{180}{x_1} - \frac{726}{x_2} - \frac{27.5}{x_3} - \frac{243}{x_4} \right) \\ &= 66.21x_1 + \frac{3395.7}{x_1} + 426.8x_2 + \frac{13,696}{x_2} + 61.20x_3 + \frac{518.79}{x_3} \\ &+ 268.1x_4 + \frac{4584.2}{x_4} - 1520.6 \end{aligned}$$

This function is convex for the same reasons that the original objective was in Section 17.2. Thus the computed stationary point is indeed a global minimum (principle [16.22](#)).

EXAMPLE 17.9: OPTIMIZING WITH LAGRANGIAN METHODS

Use Lagrange multiplier methods to compute an optimal solution to the model of Examples 17.7 and 17.8.

Solution: Stationary-point conditions from Example 17.8 are

$$\begin{aligned} 24v_1^* &= 12x_1^* \\ 24v_1^* &= 8x_2^* \\ &+ 1v_2^* = 2x_3^* \\ 24x_1^* + 24x_2^* &= 360 \\ 1x_3^* &= 1 \end{aligned}$$

Using the first 3 to substitute for x_1, \dots, x_3 in the last 2 of these constraints yields

$$\begin{aligned} 24(2v_1^*) + 24(3v_1^*) &= 360 \quad \text{or} \quad v_1^* = 3 \\ 1\left(\frac{1}{2}v_2^*\right) &= 1 \quad \text{or} \quad v_2^* = 2 \end{aligned}$$

Then solving for corresponding x_j^* produces the unique stationary point

$$(x_1^*, x_2^*, x_3^*, v_1^*, v_2^*) = (6, 9, 1, 3, 2)$$

It remains to be sure that the computed point is a minimum of the Lagrangian function

$$\begin{aligned} L(x_1, x_2, x_3, v_1^*, v_2^*) &\triangleq 6(x_1)^2 + 4(x_2)^2 + (x_3)^2 + 3(360 - 24x_1 - 24x_2) + 2(1 - x_3) \\ &= 6(x_1)^2 + 4(x_2)^2 + (x_3)^2 - 72x_1 - 72x_2 - 2x_3 + 1082 \end{aligned}$$

But this function is obviously convex (squares plus linear), so that the computed stationary point does indeed solve the original constrained model (principle [17.16](#)).

Interpretation of Lagrange Multipliers

Those readers who have followed large parts of this book will recall that we employed **dual variables** v_i on constraints of linear programs (principle [6.20](#), Section 6.3) to analyze the sensitivity of results to changes in the right-hand-side coefficients of the model. It is no accident that the same notation is used here for Lagrange multipliers.

Principle 17.17 | The optimal Lagrange multiplier, v_i^* associated with constraint $g_i(\mathbf{x}) = b_i$ can be interpreted as the rate of change in optimal value per unit increase in right-hand side b_i .

To see that this interpretation applies, we need only examine the Lagrangian function at an optimal $(\mathbf{x}^*, \mathbf{v}^*)$:

$$L(\mathbf{x}^*, \mathbf{v}^*) = f(\mathbf{x}^*) + \sum_{i=1}^m v_i^* [b_i - g_i(\mathbf{x}^*)]$$

The rate of change with RHS b_i is

$$\frac{\partial L}{\partial b_i} = v_i^*$$

For a specific example, return to the $v^* = -6.865$ on the sole constraint of Pfizer equality model (17.16). This quantity is the partial derivative of Lagrangian function (17.17) with respect to the right-hand side, which represents a production capacity in weeks. Thus because the Lagrangian function and the true objective function coincide at stationary points (expression (17.18)), this v^* tells us that small increases in the RHS would decrease (improve) the optimal objective value at the rate of \$6865 per week.

EXAMPLE 17.10: INTERPRETING OPTIMAL LAGRANGE MULTIPLIERS

Use the optimal Lagrange multipliers of Example 17.9 to analyze sensitivity of results to changes in constraint right-hand sides.

Solution: From Example 17.7, model constraints are

$$\begin{aligned} 24x_1 + 24x_2 &= 360 \\ x_3 &= 1 \end{aligned}$$

Corresponding optimal Lagrange multipliers are $v_1^* = 3$ and $v_2^* = 2$. We see that an increase in either right-hand side would increase (degrade) the optimal value of 541. At least for small changes, every unit increase in RHS 360 adds $v_1^* = 3$, and each unit increase in RHS 1 costs $v_2^* = 2$.

Limitations of the Lagrangian Approach

Although Lagrange multiplier methods work well for some models, it should be apparent that they have serious limitations:

- Stationary-point conditions [17.15] can be solved only if they are linear or very simple nonlinear functions. In other cases, solving those conditions may be more difficult than directly searching for an optimal solution to the original model.
- If a given model has many inequality constraints, it may be an explosively combinatorial task to determine which will be active at an optimal solution, so that equality-constrained Lagrangian methods may be applied.
- We can be certain that the stationary point computed from system [17.15] is a global optimum only if the original model functions were tractable enough to apply principle [17.16]. Other cases may produce ambiguous results.

There is also another, more subtle difficulty with applying Lagrange multiplier techniques to some NLPs. Principle [17.16] tells us when the \mathbf{x}^* part of an optimum for the Lagrangian function must be optimal in the original model. But the converse is not always true. That is, an optimal solution in the original model need not correspond to a stationary point of the associated Lagrangian. Although most models occurring in application do have optima satisfying conditions [17.15], we will present an example in the next section where the property fails.

EXAMPLE 17.11: UNDERSTANDING LIMITS OF LAGRANGIAN METHODS

Consider the equality-constrained nonlinear program

$$\begin{aligned} \max \quad & w_1 w_2 \\ \text{s.t.} \quad & 9(w_1)^4 - 17(w_1)^3 + 6(w_1)^2 + 3w_1 + 11e^{w_2} = 100 \end{aligned}$$

Describe the difficulties that would be encountered in trying to address this model by Lagrange multiplier methods.

Solution: Using Lagrange multiplier ν on the single constraint yields Lagrangian

$$L(w_1, w_2, \nu) \triangleq w_1 w_2 + \nu[100 - 9(w_1)^4 + 17(w_1)^3 - 6(w_1)^2 - 3w_1 - 11e^{w_2}]$$

Corresponding stationary-point conditions [17.15] are

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= w_2 - 36\nu(w_1)^3 + 51\nu(w_1)^2 - 12\nu w_1 - 3\nu &= 0 \\ \frac{\partial L}{\partial w_2} &= w_1 - 11\nu e^{w_2} &= 0 \\ \frac{\partial L}{\partial \nu} &= 100 - 9(w_1)^4 + 17(w_1)^3 - 6(w_1)^2 - 3w_1 - 11e^{w_2} &= 0 \end{aligned}$$

Solving these highly nonlinear conditions would probably be as difficult as solving the original NLP. Furthermore, the original objective function is neither convex nor concave, and the constraint function is still less tractable. Even if a stationary point could be computed, it would probably be impossible to argue that the w_1^*, w_2^* part represented an optimal solution.

17.4 KARUSH–KUHN–TUCKER OPTIMALITY CONDITIONS

The Lagrangian stationary-point conditions [17.15](#) are often difficult to solve directly for an optimum, but they do provide useful conditions that optima must (usually) satisfy. In this section we develop the elaborated form known as **Karush–Kuhn–Tucker (KKT) conditions**, which we will see are intimately related to whether a point is a local optimum of the given NLP. (See also LP Section 6.7.)

Fully Differentiable NLP Model

The Lagrangian discussion of Section 17.3 deals only with equality constraints. KKT conditions address the fully differentiable nonlinear program.

Definition 17.18 Differentiable nonlinear programs have the general form

$$\begin{aligned} & \max \text{ or } \min && f(\mathbf{x}) \\ \text{s.t.} &&& g_i(\mathbf{x}) \geq b_i && \text{for all } i \in G \\ &&& g_i(\mathbf{x}) \leq b_i && \text{for all } i \in L \\ &&& g_i(\mathbf{x}) = b_i && \text{for all } i \in E \end{aligned}$$

where f and all g_i are differentiable functions, and sets G , L , and E index the \geq , \leq , and $=$ constraints, respectively.

Complementary Slackness Conditions

The difficulty in extending Lagrangian stationarity conditions [17.15](#) to inequality cases arises in knowing what inequalities will be active at a local optimum \mathbf{x} (i.e., hold as equality). When we know that an inequality will be active, we may treat it as an equality and include it in the Lagrangian. If it will be inactive, we want it left out.

One way to formalize such requirements is to assign a Lagrange variable ν_i to every constraint but require those for inactive inequalities to $= 0$. That is, we enforce **complementary slackness** constraints like those of [6.26](#) (Section 6.3) for linear programs.

Principle 17.19 Either inequality constraints should be active at a local optimum or the corresponding Lagrange variable should $= 0$, that is,

$$\nu_i [b_i - g_i(\mathbf{x})] = 0 \quad \text{for all inequalities } i$$

We can illustrate with quadratic portfolio application (17.10).

$$\begin{aligned} \min & \quad 66.51(x_1)^2 + 2(2.61)x_1x_2 + 2(2.18)x_1x_3 \\ & \quad + 0.63(x_2)^2 && \text{(variance)} \\ & \quad + 2(0.48)x_2x_3 + 0.38(x_3)^2 && \text{(17.20)} \\ \text{s.t.} & \quad x_1 + x_2 + x_3 = 1 && \text{(invest 100\%)} \\ & \quad 13.22x_1 + 8.24x_2 + 9.03x_3 \geq 11 && \text{(return)} \\ & \quad x_1, x_2, x_3 \geq 0 \end{aligned}$$

Numbering constraints in the order given, the corresponding complementary slackness conditions are

$$\begin{aligned}
 \nu_2(11 - 13.22x_1 - 8.24x_2 - 9.03x_3) &= 0 \\
 \nu_3(-x_1) &= 0 \\
 \nu_4(-x_2) &= 0 \\
 \nu_5(-x_3) &= 0
 \end{aligned}
 \tag{17.21}$$

Notice that none is needed on equality constraint $i = 1$.

Lagrange Multiplier Sign Restrictions

We saw with interpretation [17.17] that Lagrange multipliers should reflect the rate of change in optimal value per unit increase in right-hand side b_i . Just as with linear programming results [6.20], this interpretation implies Lagrange multiplier sign restrictions when the constraints are inequalities. For example, we know that increasing RHS b_i relaxes a \leq inequality, so that it can only increase the optimal value in a maximize problem or decrease it in a minimize problem. Other cases are similar.

Principle 17.20 Lagrange multipliers ν_i on constraints i of [17.18] should satisfy the following sign restrictions:

Objective	i is \leq	i is \geq	i is $=$
Minimize	$\nu_i \leq 0$	$\nu_i \geq 0$	Unrestricted
Maximize	$\nu_i \geq 0$	$\nu_i \leq 0$	Unrestricted

Again illustrating with minimizing portfolio management model (17.20), the needed sign restrictions are

$$\nu_1 \text{ URS}; \quad \nu_2, \nu_3, \nu_4, \nu_5 \geq 0
 \tag{17.22}$$

because the first constraint is an equality, and the rest are \geq 's of a minimize model.

KKT Conditions and KKT Points

We are now ready to state Karush–Kuhn–Tucker conditions for general (differentiable) model [17.18].

Principle 17.21 Solutions \mathbf{x} and \mathbf{v} satisfy the **Karush–Kuhn–Tucker conditions** for differentiable nonlinear program [17.18] if they fulfill complementary slackness conditions [17.19], sign restrictions [17.20], gradient equation

$$\sum_i \nabla g_i(\mathbf{x}) \nu_i = \nabla f(\mathbf{x})$$

and primal constraints

$$\begin{aligned}
 g_i(\mathbf{x}) &\geq b_i && \text{for all } i \in G \\
 g_i(\mathbf{x}) &\leq b_i && \text{for all } i \in L \\
 g_i(\mathbf{x}) &= b_i && \text{for all } i \in E
 \end{aligned}$$

Any \mathbf{x} for which there exist a corresponding \mathbf{v} satisfying these conditions is called a **KKT point**.

Our portfolio model (17.20) has objective function gradient

$$\nabla f(x_1, x_2, x_3) = \begin{pmatrix} 133.02x_1 + 5.22x_2 + 4.36x_3 \\ 5.22x_1 + 1.26x_2 + 0.96x_3 \\ 4.36x_1 + 0.96x_2 + 0.76x_3 \end{pmatrix}$$

and those of the 5 linear constraints are

$$\begin{aligned} \nabla g_1(x_1, x_2, x_3) &= (1, 1, 1) \\ \nabla g_2(x_1, x_2, x_3) &= (13.22, 8.24, 9.03) \\ \nabla g_3(x_1, x_2, x_3) &= (1, 0, 0) \\ \nabla g_4(x_1, x_2, x_3) &= (0, 1, 0) \\ \nabla g_5(x_1, x_2, x_3) &= (0, 0, 1) \end{aligned}$$

Thus the gradient equation part of KKT conditions [17.21](#) is

$$\begin{aligned} 1\nu_1 + 13.22\nu_2 + \nu_3 &= 133.02x_1 + 5.22x_2 + 4.36x_3 \\ 1\nu_1 + 8.24\nu_2 + \nu_4 &= 5.22x_1 + 1.26x_2 + 0.96x_3 \\ 1\nu_1 + 9.03\nu_2 + \nu_5 &= 4.36x_1 + 0.96x_2 + 0.76x_3 \end{aligned} \quad (17.23)$$

The rest of the conditions are primal constraints

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 13.22x_1 + 8.24x_2 + 9.03x_3 &\geq 11 \\ x_1, x_2, x_3 &\geq 0 \end{aligned} \quad (17.24)$$

complementary slackness (17.21), and sign restrictions (17.22).

Notice the direct parallel to Lagrangian stationary-point conditions [17.15](#). Both sets of conditions require the objective function gradient to be expressible as a multiplier-weighted combination of constraint gradients, while primal constraints are also satisfied. The new elements are complementary slackness conditions and sign restrictions arising from inequalities.

EXAMPLE 17.12: FORMULATING KKT CONDITIONS

Consider the nonlinear program

$$\begin{aligned} \max \quad & 2w_1 + 7w_2 \\ \text{s.t.} \quad & (w_1 - 2)^2 + (w_2 - 2)^2 = 1 \\ & w_1 \leq 2 \\ & w_2 \leq 2 \\ & w_1 \geq 0 \\ & w_2 \geq 0 \end{aligned}$$

State the Karush–Kuhn–Tucker conditions for this model.

Solution: We apply definition [17.21](#). Numbering constraints in the order given,

$$\begin{aligned}\nabla f(w_1, w_2) &= (2, 7) \\ \nabla g_1(w_1, w_2) &= (2w_1, 2w_2) \\ \nabla g_2(w_1, w_2) &= (1, 0) \\ \nabla g_3(w_1, w_2) &= (0, 1) \\ \nabla g_4(w_1, w_2) &= (1, 0) \\ \nabla g_5(w_1, w_2) &= (0, 1)\end{aligned}$$

Thus the KKT conditions consist of primal constraints

$$\begin{aligned}(w_1 - 2)^2 + (w_2 - 2)^2 &= 1 \\ w_1 &\leq 2 \\ w_2 &\leq 2 \\ w_1 &\geq 0 \\ w_2 &\geq 0\end{aligned}$$

gradient equation

$$\begin{pmatrix} 2w_1 \\ 2w_2 \end{pmatrix} \nu_1 + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \nu_2 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \nu_3 + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \nu_4 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \nu_5 = \begin{pmatrix} 2 \\ 7 \end{pmatrix}$$

complementary slackness

$$\begin{aligned}\nu_2(2 - w_1) &= 0 \\ \nu_3(2 - w_2) &= 0 \\ \nu_4(0 - w_1) &= 0 \\ \nu_5(0 - w_2) &= 0\end{aligned}$$

and sign restrictions

$$\begin{aligned}\nu_2, \nu_3 &\geq 0 \\ \nu_3, \nu_4 &\leq 0\end{aligned}$$

Improving Feasible Directions and Local Optima Revisited

To see the importance of KKT conditions [17.21](#) in constrained nonlinear programming, we must return to the elementary improving search notions of Sections 3.2 and 3.3. Move directions $\Delta \mathbf{x}$ pursued by implementations of improving search should be both improving and feasible. That is (definitions [3.13](#) and [3.14](#)), they should improve the objective and maintain feasibility for sufficiently small steps λ .

If there is such an **improving feasible direction** available at a current solution in the search, the point cannot be even locally optimal (principle [3.16](#)). Progress is still possible in every neighborhood of the current point by advancing in the available direction.

When no improving feasible direction exists, the current solution is under mild assumptions, at least a local optimum (principle [3.17](#)). Still, cases such as Figure 3.8 show that the absence of improving feasible directions does not always imply local optimality. We can only be certain the search will stop.

Principle 17.22 | Absence of an improving feasible direction at the current point of an improving search, which causes an improving search to stop, provides a working definition of when a local optimum has been reached.

When does a direction $\Delta \mathbf{x}$ improve at current \mathbf{x} ? First-order Taylor series approximation (definition [16.17](#))

$$f(\mathbf{x} + \Delta \mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot \Delta \mathbf{x}$$

suggests that improvement depends on the sign of $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x}$. This yields conditions [3.21](#) and [3.22](#).

Principle 17.23 | The linear Taylor approximation to smooth objective function $f(\mathbf{x})$ shows the following about move direction $\Delta \mathbf{x}$:

Objective	$\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} > 0$	$\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} < 0$
Maximize	Improving	Nonimproving
Minimize	Nonimproving	Improving

If $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} = 0$, more information is required to classify $\Delta \mathbf{x}$.

A direction can still improve if $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} = 0$, but the absence of any $\Delta \mathbf{x}$ satisfying first-order condition [17.23](#) is a strong indication that no improving directions exist.

The corresponding feasibility conditions [3.25](#) are exact for linear constraints. Direction $\Delta \mathbf{x}$ is feasible if

$$\mathbf{a} \cdot \Delta \mathbf{x} \begin{cases} \leq 0 & \text{for active constraints} & \mathbf{a} \cdot \mathbf{x} \leq b \\ = 0 & \text{for all constraints} & \mathbf{a} \cdot \mathbf{x} = b \\ \geq 0 & \text{for active constraints} & \mathbf{a} \cdot \mathbf{x} \geq b \end{cases}$$

Inactive constraints need not be considered because they have no immediate impact on feasibility.

To generalize for nonlinear constraints we may again employ first-order Taylor series approximations (definition [16.17](#)):

$$g_i(\mathbf{x} + \Delta \mathbf{x}) \approx g_i(\mathbf{x}) + \nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x} \tag{17.25}$$

If g_i is active at \mathbf{x} , $g_i(\mathbf{x}) = b_i$. Thus feasibility in (17.25) depends on the sign of the term $\nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x}$.

Principle 17.24 | Direction $\Delta \mathbf{x}$ is feasible at \mathbf{x} for the linear Taylor approximation to constrained nonlinear program [17.18] if

$$\nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x} \begin{cases} \geq 0 & \text{for active } \geq \text{ constraints} \\ \leq 0 & \text{for active } \leq \text{ constraints} \\ = 0 & \text{for all } = \text{ constraints} \end{cases}$$

KKT Conditions and Existence of Improving Feasible Directions

We are now in a position to link Karush–Kuhn–Tucker conditions [17.21] with the existence of improving feasible directions at a current search point \mathbf{x} .

Principle 17.25 | Karush–Kuhn–Tucker conditions provide a first-order, working test of the absence of improving feasible directions. More specifically, \mathbf{x} is a KKT point if and only if no direction of movement from \mathbf{x} fulfills first-order tests [17.23] and [17.24] for an improving feasible direction.

To illustrate, recall that an optimal solution for portfolio management model (17.20) is

$$x_1^* = 0.43, \quad x_2^* = 0, \quad x_3^* = 0.57$$

Certainly, there are no improving feasible directions at this global optimum, and we may demonstrate that fact by finding corresponding v_i^* to satisfy KKT conditions (17.21) – (17.24). Values that will do the job are

$$v_1^* = -132.026, \quad v_2^* = 14.892, \quad v_3^* = 0.0, \quad v_4^* = 12.275, \quad v_5^* = 0.0 \quad (17.26)$$

It is easy to check that these primal and multiplier values satisfy complementary slackness conditions (17.21), sign restrictions (17.22), and primal constraints (17.24). In gradient equation (17.23)

$$\begin{aligned} 1v_1 + 13.22v_2 + v_3 &= 1(-132.026) + 13.22(14.892) + (0.0) = 64.8 \\ 133.02x_1 + 5.22x_2 + 4.36x_3 &= 133.02(0.47) + 5.22(0) + 4.36(0.53) = 64.8 \\ 1v_1 + 8.24v_2 + v_4 &= 1(-132.026) + 8.24(14.892) + (12.275) = 2.96 \\ 5.22x_1 + 1.26x_2 + 0.96x_3 &= 5.22(0.47) + 1.26(0.0) + 0.96(0.53) = 2.96 \\ 1v_1 + 9.03v_2 + v_5 &= 1(-132.026) + 9.03(14.892) + (0.0) = 2.45 \\ 4.36x_1 + 0.96x_2 + 0.76x_3 &= 4.36(0.47) + 0.96(0.0) + 0.76(0.53) = 2.45 \end{aligned} \quad (17.27)$$

Contrast this optimal point with $\mathbf{x} = (1, 0, 0)$, where direction $\Delta \mathbf{x} = (-1, 0, 1)$ satisfies first-order conditions [17.23] and [17.24] because

$$\nabla f(1, 0, 0) \cdot \Delta \mathbf{x} = (133.02, 5.22, 4.36) \cdot (-1, 0, 1) = -128.66 < 0$$

and active constraints

$$\begin{aligned} \nabla g_1(1, 0, 0) \cdot \Delta \mathbf{x} &= (1, 1, 1) \cdot (-1, 0, 1) = 0 \\ \nabla g_4(1, 0, 0) \cdot \Delta \mathbf{x} &= (0, 1, 0) \cdot (-1, 0, 1) \geq 0 \\ \nabla g_5(1, 0, 0) \cdot \Delta \mathbf{x} &= (0, 0, 1) \cdot (-1, 0, 1) \geq 0 \end{aligned}$$

(Note that $v_2 = v_3 = 0$ for complementary slackness.) But now the unique solution to gradient equation

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} v_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} v_4 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v_5 = (133.02, 5.22, 4.36)$$

is

$$v_1 = 133.02, \quad v_2 = -127.8, \quad v_3 = -128.66$$

which violates sign restrictions $v_4, v_5 \geq 0$. KKT conditions cannot be satisfied.

To see why principle [17.25](#) must always be true, we may think of improving feasible conditions [17.23](#) and [17.24](#) as a linear program in decision variables $\Delta \mathbf{x}$. Taking the minimize case,

$$\begin{aligned} \min \quad & \nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} \\ \text{s.t.} \quad & \nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x} \geq 0 \quad \text{for active } \geq \text{'s} \\ & \nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x} \leq 0 \quad \text{for active } \leq \text{'s} \\ & \nabla g_i(\mathbf{x}) \cdot \Delta \mathbf{x} = 0 \quad \text{for all } = \text{'s} \end{aligned} \tag{17.28}$$

Now we apply some linear programming duality from Section 6.4. Over multipliers v_i , the dual of (17.28) is

$$\begin{aligned} \max \quad & \sum_{i \text{ active}} (0)v_i = 0 \\ \text{s.t.} \quad & \sum_{i \text{ active}} \nabla g_i(\mathbf{x})v_i = \nabla f(\mathbf{x}) \\ & v_i \geq 0 \quad \text{for active } \geq \text{'s} \\ & v_i \leq 0 \quad \text{for active } \leq \text{'s} \end{aligned} \tag{17.29}$$

Notice that the feasibility requirements of dual (17.29) are identical to the gradient equation and sign restriction part of KKT conditions [17.21](#) at \mathbf{x} (assuming that $v_i = 0$ on inactive constraints to satisfy complementary slackness). If any v_i fulfill these conditions, the dual is feasible and its objective value is constant zero. It follows (principle [6.51](#)) that the optimal $\Delta \mathbf{x}$ in primal (17.28) has the same optimal value $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} = 0$. If KKT conditions are fulfilled, no $\Delta \mathbf{x}$ can fulfill all the conditions of [17.23](#) and [17.24](#).

On the other hand, if some $\Delta \mathbf{x}$ meets all the conditions of [17.23](#) and [17.24](#), there is a feasible solution to the primal with $\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} < 0$. Then the dual must be infeasible, because every dual solution bounds the primal optimum (principle [6.47](#)) and any would have objective value $= 0$. It follows that KKT conditions cannot be fulfilled at \mathbf{x} .

EXAMPLE 17.13: VERIFYING KKT AS A NO-DIRECTION CHECK

Consider the constrained NLP

$$\begin{aligned} \min \quad & (w_1)^2 + (w_2)^2 \\ \text{s.t.} \quad & w_1 + w_2 = 1 \\ & w_1, w_2 \geq 0 \end{aligned}$$

A global optimum is $w_1^* = w_2^* = \frac{1}{2}$.

- (a) State the KKT conditions for this problem.
- (b) Verify that $\Delta \mathbf{w} = (1, -1)$ satisfies the first-order conditions for an improving feasible direction at $\mathbf{w} = (0, 1)$, and that the corresponding KKT conditions have no solution.
- (c) Verify that KKT conditions hold at the optimal \mathbf{w}^* , so that no direction can meet first-order tests for being improving and feasible.

Solution:

(a) Following [17.21] with Lagrange multipliers v_1, v_2, v_3 on the three constraints, conditions are

$$\begin{aligned}
 w_1 + w_2 &= 1 && \text{(primal constraints)} \\
 w_1, w_2 &\geq 0 \\
 v_2(-w_1) &= 0 && \text{(complementary slackness)} \\
 v_3(-w_2) &= 0 \\
 \begin{pmatrix} 1 \\ 1 \end{pmatrix} v_1 + \begin{pmatrix} 1 \\ 0 \end{pmatrix} v_2 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v_3 &= \begin{pmatrix} 2w_1 \\ 2w_2 \end{pmatrix} && \text{(gradient equation)} \\
 v_2, v_3 &\geq 0 && \text{(sign restrictions)}
 \end{aligned}$$

(b) Direction $\Delta \mathbf{w} = (1, -1)$ meets improving test [17.23] at $\mathbf{w} = (0, 1)$ because

$$\nabla f(0, 1) \cdot \Delta \mathbf{w} = (0, 2) \cdot (1, -1) < 0$$

It is also feasible because active constraints have

$$\begin{aligned}
 \nabla g_1(0, 1) \cdot \Delta \mathbf{w} &= (1, 1) \cdot (1, -1) = 0 \\
 \nabla g_2(0, 1) \cdot \Delta \mathbf{w} &= (1, 0) \cdot (1, -1) \geq 0
 \end{aligned}$$

Solution $\mathbf{w} = (0, 1)$ satisfies the primal constraint part of KKT conditions in part (a), and making $v_2 = 0$ will assure complementary slackness. Solving gradient equation

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} v_1 + \begin{pmatrix} 1 \\ 0 \end{pmatrix} v_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

yields unique solution $v_1 = 2, v_2 = -2$, which violates the sign restriction on v_2 . KKT conditions cannot be satisfied.

(c) Optimum $w_1^* = w_2^* = \frac{1}{2}$ satisfies primal constraints and is active only in the first, thus corresponding $v_2 = v_3 = 0$ to conform to complementary slackness. This leaves gradient equation

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

which has solution $v_1 = 1$ satisfying all sign restrictions. KKT conditions do hold.

Sufficiency of KKT Conditions for Optimality

Since principle [17.25](#) shows that a KKT point is one that admits no direction satisfying first-order conditions for improving feasibility, it follows that KKT conditions are sufficient to establish optimality whenever the absence of improving feasible directions is sufficient. The most common case is **convex programs** (definition [17.3](#)).

Principle 17.26 | If \mathbf{x} is a KKT point of a convex program, \mathbf{x} is a global optimum.

For example, our portfolio application (17.20) is a convex program because all its constraints are linear, and its objective function is convex because it has positive-definite Hessian matrix

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 133.02 & 5.22 & 4.36 \\ 5.22 & 1.26 & 0.96 \\ 4.36 & 0.96 & 0.76 \end{pmatrix} \quad (17.30)$$

Thus, when we verified that $x_1^* = 0.47$, $x_2^* = 0.0$, $x_3^* = 0.53$ is a KKT point in computation (17.27), we proved that the solution was optimal (principle [17.26](#)).

Necessity of KKT Conditions for Optimality

A much more subtle issue than when KKT conditions are sufficient to establish a point's optimality is when they are necessary. That is, when must optimal points satisfy KKT conditions?

To see the issue, consider the NLP

$$\begin{aligned} \min \quad & (y_1)^2 + 4y_2 \\ \text{s.t.} \quad & (y_1 - 1)^2 + (y_2)^2 \leq 1 \\ & (y_1 + 1)^2 + (y_2)^2 \leq 1 \end{aligned} \quad (17.31)$$

It is easy to check that this model is a convex program because the objective and both constraints are convex. Also, the only feasible solution $y_1 = y_2 = 0$ has to be optimal.

For KKT conditions [17.21](#),

$$\nabla f(y_1, y_2) = (2y_1, 4), \quad \nabla g_1(y_1, y_2) = (2y_1 - 2, 2y_2), \quad \nabla g_2(y_1, y_2) = (2y_1 + 2, 2y_2)$$

so that the gradient equation part at $\mathbf{y} = (0, 0)$ becomes

$$\begin{pmatrix} -2 \\ 0 \end{pmatrix} v_1 + \begin{pmatrix} 2 \\ 0 \end{pmatrix} v_2 = \begin{pmatrix} 0 \\ 4 \end{pmatrix}$$

Clearly, there is no solution v_1, v_2 . Even though $y_1 = y_2 = 0$ is a global optimum of a convex program, it is not a KKT point.

Fortunately, such cases where KKT conditions are not necessary are rare in common models. Also, a variety of **constraint qualifications** have been derived to characterize models where every local or global optimum is a KKT point. We present here only the easiest to apply.

Principle 17.27 | A local optimum solution of a constrained differentiable NLP must be a KKT point if (1) all constraints are linear; or (2) the gradients of all constraints active at the local optimum are linearly independent.

EXAMPLE 17.14: VERIFYING NECESSITY OF KKT CONDITIONS

Without actually solving, verify that every local optimum of the following models must be a KKT point.

$$\begin{array}{ll} \text{(a) max} & (w_1)^2 + e^{w_2} + w_1 w_2 \\ \text{s.t.} & 3w_1 + w_2 \leq 9 \\ & w_1, w_2 \geq 0 \end{array} \qquad \begin{array}{ll} \text{(b) max} & (w_1)^2 + e^{w_2} + w_1 w_2 \\ \text{s.t.} & (w_1 - 1)^2 \leq 1 \\ & (w_2 - 2)^2 \leq 4 \end{array}$$

Solution: We apply constraint qualifications [17.27], which depend only on the constraints of the models.

(a) All constraints of this model are linear, so every local optimum must be a KKT point.

(b) Constraint gradients are

$$\nabla g_1(w_1, w_2) = (2w_1 - 2, 0) \quad \text{and} \quad \nabla g_2(w_1, w_2) = (0, 2w_2 - 4)$$

These constraints are linearly independent except at $\mathbf{w} = (1, 2)$, which is not feasible. Thus the active constraints at any local optimum will be linearly independent, and all such solutions must satisfy KKT conditions.

17.5 PENALTY AND BARRIER METHODS

One approach to solving constrained nonlinear programs is to convert them to a series of unconstrained ones. In this section we investigate such **sequential unconstrained min/maximization techniques (SUMT)**, also known as **penalty** and **barrier** methods. (See also Section 7.4 for the LP case.)

Penalty Methods

One scheme for transforming constrained into unconstrained NLPs uses penalty methods.

Definition 17.28 | **Penalty methods** drop constraints of nonlinear programs and substitute new terms in the objective function penalizing infeasibility in the form

$$\max \text{ or } \min F(\mathbf{x}) \triangleq f(\mathbf{x}) \pm \mu \sum_i p_i(\mathbf{x})$$

(+ for minimize problems and – for maximize problems), where μ is a positive **penalty multiplier** and the p_i are functions satisfying

$$p_i(\mathbf{x}) \begin{cases} = 0 & \text{if } \mathbf{x} \text{ satisfies constraint } i \\ > 0 & \text{otherwise} \end{cases}$$

Many alternatives are available for the **penalty functions** $p_i(\mathbf{x})$ associated with particular constraints.

Principle 17.29 Among the common penalty functions employed for constrained NLPs are

$\max\{0, b_i - g_i(\mathbf{x})\}$	and	$\max^2\{0, b_i - g_i(\mathbf{x})\}$	for \geq 's
$\max\{0, g_i(\mathbf{x}) - b_i\}$	and	$\max^2\{0, g_i(\mathbf{x}) - b_i\}$	for \leq 's
$ g_i(\mathbf{x}) - b_i $	and	$ g_i(\mathbf{x}) - b_i ^2$	for $=$'s

Each imposes no penalty when the corresponding constraint is satisfied, but adds a growing cost if it is violated.

APPLICATION 17.7: SERVICE DESK DESIGN

Penalty methods are most often used in engineering design applications where many of the constraints are nonlinear. We will illustrate with a contrived example to design the service desk of a catalog order company.

Figure 17.6 displays the problem. A service desk $\frac{1}{2}$ meter in width is to be centered around two 1-meter conveyors bringing orders from warehouse storage.

The conveyors are 6 meters apart (center-to-center) and protrude 0.75 meter into the work area. For employees to work efficiently behind the counter, there should be at least 2 meters clearance in front of the conveyors, and no part of the inside counter perimeter should total more than 10 meters from the conveyors. Within these limits we wish to maximize the customer room provided by the outside perimeter of the counter.

To model this simple example, introduce an origin halfway between the conveyors, and define decision variables

- $x_1 \triangleq$ half-length of the work area inside the counter
- $x_2 \triangleq$ width of the work area inside the counter

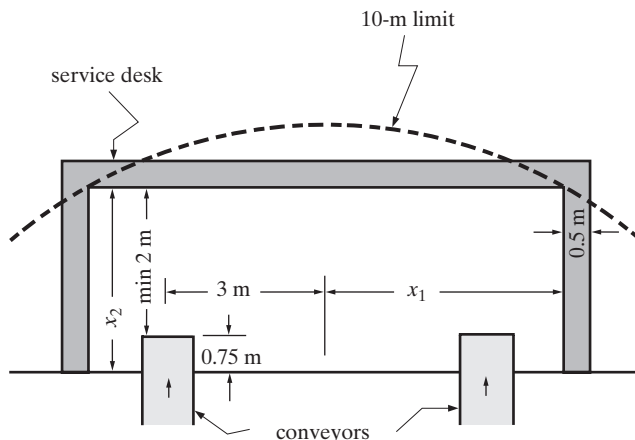


FIGURE 17.6 Service Desk Design Application

Then the problem can be modeled:

$$\begin{aligned}
 \max \quad & 2x_1 + 2x_2 + 2 && \text{(outer perimeter)} \\
 \text{s.t.} \quad & \frac{(x_1)^2}{(5)^2} + \frac{(x_2)^2}{(4)^2} \leq 1 && \text{(10-m distance limit)} \\
 & x_1 \geq 3.5 && \text{(outside conveyors)} \\
 & x_2 \geq 2.75 && \text{(2-m inside space)}
 \end{aligned} \tag{17.32}$$

The objective function maximizes the outer perimeter. The first (ellipse) constraint keeps the most distant point inside the counter at most a total of 10 meters from the conveyors. The lower bound on x_1 ensures that the counter falls outside the conveyors, and that of x_2 enforces the 2-meter inside clearance. An optimal design uses a desk with inside dimensions $2x_1^* = 2(3.63)$ meters by $x_2^* = 2.75$ meters and outside perimeter 14.76 meters.

Penalty Treatment of the Service Desk Application

Any of the penalty function alternatives in principle [17.29] might be used to deal with the \leq constraint and two \geq constraints of service desk model (17.32). We will choose the second, squared penalty forms. For example, in the first constraint

$$p_1(x_1, x_2) \triangleq \max^2 \left\{ 0, \frac{(x_1)^2}{25} + \frac{(x_2)^2}{16} - 1 \right\}$$

When the constraint is satisfied $[(x_1)^2/25 + (x_2)^2/16 - 1] \leq 0$ and $p_1(x_1, x_2) = 0$. However, violations of the constraint make $[(x_1)^2/25 + (x_2)^2/16 - 1] > 0$ and impose a penalty equal to the square of the violation. Proceeding in this manner with all constraints yields the unconstrained penalty model

$$\begin{aligned}
 \max \quad & 2x_1 + 2x_2 + 2 - \mu \left(\max^2 \left\{ 0, \frac{(x_1)^2}{25} + \frac{(x_2)^2}{16} - 1 \right\} \right. \\
 & \left. + \max^2 \{ 0, 3.5 - x_1 \} + \max^2 \{ 0, 2.75 - x_2 \} \right)
 \end{aligned} \tag{17.33}$$

Infeasible solutions in the constrained model (17.32) are now allowed, but they are discouraged by subtracting a penalty in the objective function.

EXAMPLE 17.15: FORMING PENALTY MODELS

Use absolute value (unsquared) penalty functions to reduce the following constrained NLP to an unconstrained penalty model.

$$\begin{aligned}
 \min \quad & (w_1)^4 - w_1 w_2 w_3 \\
 \text{s.t.} \quad & w_1 + w_2 + w_3 = 5 \\
 & (w_1)^2 + (w_2)^2 \leq 9 \\
 & w_3 w_2 \geq 1
 \end{aligned}$$

Solution: Using the first, unsquared penalty alternatives of [17.29], the corresponding unconstrained model [17.28] is

$$\begin{aligned} \min \quad & (w_1)^4 - w_1 w_2 w_3 + \mu(|w_1 + w_2 + w_3 - 5| \\ & + \max\{0, (w_1)^2 + (w_2)^2 - 9\} + \max\{0, 1 - w_3 w_2\}) \end{aligned}$$

where μ is a positive penalty multiplier.

Concluding Constrained Optimality with Penalties

By definition, the penalty terms of definitions [17.28] must = 0 at any \mathbf{x} feasible in the given constrained NLP. This provides a way to know when unconstrained optimization of the penalty problem yields an optimal solution for the original model.

Principle 17.30 If an optimal \mathbf{x}^* in unconstrained penalty problem [17.28] is feasible in the original constrained model, it is optimal in that NLP.

Any better solution to the constrained model would also have all penalty terms = 0, so it would have to best \mathbf{x}^* in penalty model objective value.

Differentiability of Penalty Functions

One consideration in choosing among the penalty options in [17.29] is differentiability. Most of Chapter 16's unconstrained methods that might be employed to optimize penalty model [17.28] assume that the function is smooth. None of the first options listed in principle [17.29] meet this differentiability requirement, but the second, squared options do.

Principle 17.31 Squared penalty options of principle [17.29] are differentiable whenever the underlying g_i are differentiable.

We can see why by examining a \leq inequality $g_i(\mathbf{x}) \leq b_i$ with g_i smooth. The corresponding squared penalty term can be expressed as

$$p_i(x) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ satisfies } g_i(\mathbf{x}) \leq b_i \\ [g_i(\mathbf{x}) - b_i]^2 & \text{otherwise} \end{cases}$$

Associated partial derivatives are

$$\frac{\partial p_i}{\partial x_j} = \begin{cases} 0 & \text{if } \mathbf{x} \text{ satisfies } g_i(\mathbf{x}) \leq b_i \\ 2[g_i(\mathbf{x}) - b_i] \frac{\partial g_i}{\partial x_j} & \text{otherwise} \end{cases}$$

Notice that these two expressions match at the boundary where $g_i(\mathbf{x}) = b_i$. Thus partial derivatives are well defined and continuous.

Exact Penalty Functions

We would also like penalty functions to be **exact**. That is, we would like there to be a large enough $\mu > 0$ that the unconstrained penalty model $F(\mathbf{x})$ yields an optimal solution in the original constrained form under principle 17.19 by driving out all infeasibility.

A trivial example shows that the squared alternatives of 17.29 may not be exact. Consider

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & y \geq 0 \end{aligned} \tag{17.34}$$

The squared choice in 17.29 produces penalty model

$$\begin{aligned} \min F(y) &\triangleq y + \mu \max^2 \{0, -y\} \\ &= \begin{cases} y & \text{if } y \geq 0 \\ y + \mu y^2 & \text{if } y < 0 \end{cases} \end{aligned}$$

Differentiating

$$\frac{dF}{dy} = \begin{cases} 1 & \text{if } y \geq 0 \\ 1 + 2\mu y & \text{if } y < 0 \end{cases}$$

shows that the only stationary point is

$$y^* = -\frac{1}{2\mu}$$

But this unconstrained minimum is negative and infeasible, regardless of the magnitude of μ .

Principle 17.32 Squared penalty alternatives 17.29 are usually not exact; that is, there will often be no choice of penalty multiplier μ for which the corresponding unconstrained optimum of penalty model F is optimal in the original NLP.

Suppose that we had used nondifferentiable penalty function $\max\{0, b_i - g_i(\mathbf{x})\}$ on application (17.34). The corresponding penalty model is

$$\min F(y) \triangleq y + \mu \max\{0, -y\}$$

Now for any $\mu > 1$, $F(y)$ is minimized at $y = 0$. That is, a finite μ is large enough to make the penalty optimum feasible in the original NLP.

All of the nonsquared penalty forms in 17.29 are exact in this way.

Principle 17.33 If nonsquared penalty forms of 17.29 are applied to a constrained non-linear program having an optimal solution, and mild assumptions hold, there exists a finite multiplier μ sufficiently large that an optimum in unconstrained penalty problem 17.28 is optimal in the given NLP.

Managing the Penalty Multiplier

In squared cases such as (17.33), a constrained optimum can be obtained with penalty methods only by letting $\mu \rightarrow \infty$. With exact methods of principle [17.33] there is a large enough finite μ to do the job. Either way, μ needs to grow large, and we cannot know how large when we begin.

Why not just use a very large μ from the start? Figure 17.7 shows the risk with trivial model

$$\begin{aligned} \min \quad & w \\ \text{s.t.} \quad & 3 \leq w \leq 5 \end{aligned} \quad (17.35)$$

When μ is comparatively large, the corresponding penalty objective function F becomes very steep. Small moves have dramatic impacts on its value. The result is an unconstrained model that is difficult to solve with any of the methods of Chapter 16.

These competing demands on the penalty multiplier μ motivate a sequential strategy that slowly increases the multiplier.

Principle 17.34 When addressing a constrained nonlinear program by penalty methods, the multiplier μ should be started at a relatively low value > 0 and increased as computation proceeds.

Sequential Unconstrained Penalty Technique (SUMT)

Formalization of principle [17.34]'s strategy for slowly increasing the penalty multiplier μ produces the **sequential unconstrained penalty technique** of Algorithm 17A. Multiplier μ begins relatively small and grows with each search. For each value of μ , unconstrained penalty problem [17.28] is solved beginning with the optimum of the preceding search. If the result is ever feasible in the original model, we stop with an optimum (principle [17.30]). Otherwise, we continue until the unconstrained optimum is sufficiently close to feasible.

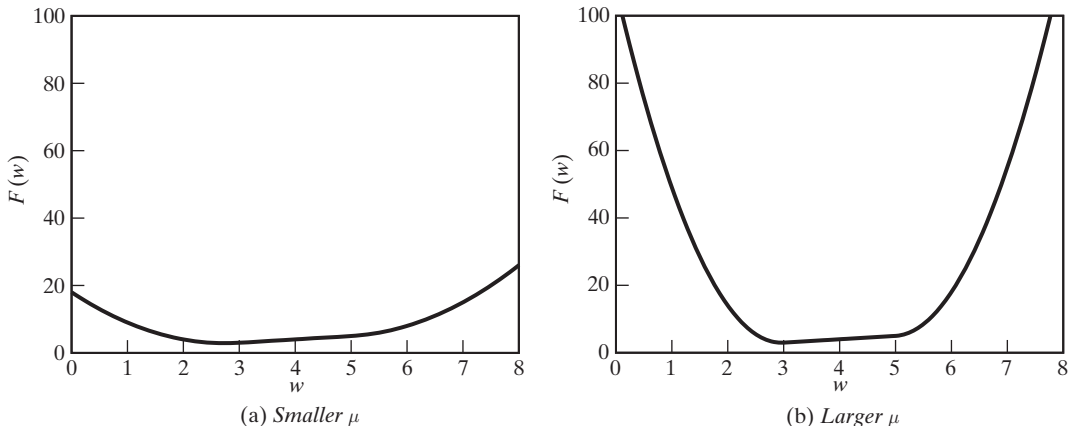


FIGURE 17.7 Effect of Penalty Multiplier on Unconstrained Tractability

ALGORITHM 17A: SEQUENTIAL UNCONSTRAINED PENALTY TECHNIQUE (SUMT)

- Step 0: Initialization.** Form penalty model [17.28], and choose initial penalty multiplier $\mu_0 > 0$ relatively small and starting solution $\mathbf{x}^{(0)}$. Also, initialize solution index $t \leftarrow 0$, and pick an escalation factor $\beta > 1$.
- Step 1: Unconstrained Optimization.** Beginning from $\mathbf{x}^{(t)}$, solve penalty optimization problem [17.28] with $\mu = \mu_t$ to produce optimum $\mathbf{x}^{(t+1)}$.
- Step 2: Stopping.** If $\mathbf{x}^{(t+1)}$ is feasible or sufficiently close to feasible in the constrained model given, stop and output $\mathbf{x}^{(t+1)}$.
- Step 3: Increase.** Enlarge the penalty multiplier as

$$\mu_{t+1} \leftarrow \beta \mu_t$$
 Then advance $t \leftarrow t + 1$, and return to Step 1.

Table 17.3 illustrates Algorithm 17A for service desk design model (17.32) and corresponding penalty form (17.33). With initial multiplier $\mu = \frac{1}{4}$, a first search produces the unconstrained optimum

$$\mathbf{x}^{(1)} = (9.690, 6.202)$$

which violates the first constraint of the original model by 5.160. Then, the multiplier is increased by factor $\beta = 4$ and a new unconstrained search initiated from $\mathbf{x}^{(1)}$. The resulting optimum $\mathbf{x}^{(2)}$ starts a third search after μ is quadrupled again. The process continues until μ is large enough that the unconstrained optimum approaches feasibility. There we stop to obtain constrained optimal solution $\mathbf{x}^* = (3.63, 2.75)$.

Barrier Methods

The penalty methods above begin anywhere and try to force the unconstrained optimum into the feasible set of the given NLP. Barrier methods adopt the alternative of beginning with a feasible solution and trying to prevent the unconstrained search from leaving the feasible region.

TABLE 17.3 Sequential Penalty Solution of Service Desk Application

<i>t</i>	μ	Optimal $\mathbf{x}^{(t+1)}$	Constraint Violation, <i>i</i>		
			1	2	3
0	$\frac{1}{4}$	(9.690, 6.202)	5.160	0.000	0.000
1	1	(6.632, 4.244)	1.885	0.000	0.000
2	4	(4.981, 3.188)	0.627	0.000	0.000
3	16	(4.221, 2.749)	0.185	0.000	0.001
4	64	(3.806, 2.748)	0.051	0.000	0.002
5	256	(3.677, 2.749)	0.013	0.000	0.001
6	1024	(3.643, 2.750)	0.003	0.000	0.000
7	4096	(3.634, 2.750)	0.001	0.000	0.000

Definition 17.35 | **Barrier methods** drop constraints of nonlinear programs and substitute new terms in the objective function discouraging any approach to the boundary of the feasible region in the form

$$\max \text{ or } \min F(\mathbf{x}) \triangleq f(\mathbf{x}) \pm \mu \sum_j q_j(\mathbf{x})$$

(+ for minimize problems and – for maximize problems), where μ is a positive **barrier multiplier** and the q_i are functions with

$$q_i(\mathbf{x}) \rightarrow \infty$$

as constraint i approaches being active.

Since the boundary cannot be avoided with equality constraints, barrier methods are applicable only when constraints are all inequalities. Many alternatives are available for the $q_i(\mathbf{x})$ associated with such constraints.

Principle 17.36 | Among the common barrier functions associated with inequality constrained NLPs are

$$\begin{aligned} -\ln[g_i(\mathbf{x}) - b_i] \quad \text{and} \quad \frac{1}{g_i(\mathbf{x}) - b_i} \quad \text{for } \geq \text{'s} \\ -\ln[b_i - g_i(\mathbf{x})] \quad \text{and} \quad \frac{1}{b_i - g_i(\mathbf{x})} \quad \text{for } \leq \text{'s} \end{aligned}$$

Each explodes toward $+\infty$ as the corresponding inequality approaches being satisfied as an equality.

Barrier Treatment of Service Desk Application

All constraints of service desk design model (17.32) are inequalities, so barrier methods could be applied. We will adopt the more common logarithmic forms. For example, the first constraint produces barrier term

$$q_1(x_1, x_2) \triangleq -\ln \left(1 - \frac{(x_1)^2}{25} - \frac{(x_2)^2}{16} \right)$$

When \mathbf{x} is well inside the feasible region, this barrier function affects the objective only modestly. But as $((x_1)^2/25 + (x_2)^2/16) \rightarrow 1$, the negative of the logarithm goes to $+\infty$. Similar treatment of all constraints yields the unconstrained barrier model

$$\begin{aligned} \max \quad & 2x_1 + 2x_2 + 2 + \mu \left[\ln \left(1 - \frac{(x_1)^2}{25} - \frac{(x_2)^2}{16} \right) \right] \\ & + \ln(x_1 - 3.5) + \ln(x_2 - 2.75) \end{aligned} \quad (17.36)$$

With $\mu > 0$, an approach to any part of the boundary is discouraged.

EXAMPLE 17.16: FORMING BARRIER MODELS

Use reciprocal barrier functions to reduce the following constrained NLP to an unconstrained barrier model.

$$\begin{aligned} \min \quad & (w_1)^4 - w_1 w_2 w_3 \\ \text{s.t.} \quad & (w_1)^2 + (w_2)^2 \leq 9 \\ & w_3 w_2 \geq 1 \end{aligned}$$

Solution: Using second, reciprocal barrier alternatives of [17.36], the corresponding unconstrained model [17.35] is

$$\min \quad (w_1)^4 - w_1 w_2 w_3 + \mu \left(\frac{1}{9 - (w_1)^2 - (w_2)^2} + \frac{1}{w_3 w_2 - 1} \right)$$

where μ is a positive barrier multiplier.

Converging to Optimality with Barrier Methods

Unlike penalty methods, barrier functions affect the objective function at feasible points. We can illustrate the difficulty this causes by returning to the trivial $\min y$, s.t. $y \geq 0$ example of (17.34). Using, say, the logarithmic barrier alternative of [17.36], the corresponding barrier problem is

$$\min \quad F(y) \triangleq y - \mu \ln(y)$$

Differentiating (with $y > 0$) yields

$$\frac{dF}{dy} = 1 - \frac{\mu}{y}$$

which has its only stationary point at

$$y^* = \mu$$

This unconstrained optimum never reaches the true optimum of $y = 0$ for any $\mu > 0$.

Similar behavior occurs for all barrier versions of constrained NLPs with an optimum on the boundary of the feasible set.

Principle 17.37 | The optimum of barrier function [17.35] can never equal the true optimum of the given constrained NLP if $\mu > 0$ and the optimum lies on the boundary of the feasible set.

As with penalty methods, however, there is a pattern to the unconstrained optima. As $\mu \rightarrow 0$, the unconstrained optimum comes closer and closer to the constrained solution.

Principle 17.38 | Although none may actually solve the given NLP, if mild assumptions hold, the sequence of unconstrained barrier function optima converges to an optimal solution to the given constrained NLP as multiplier $\mu \rightarrow 0$.

Managing the Barrier Multiplier

Property [17.38](#) makes it clear that we have to let barrier multipliers μ approach zero if we expect to obtain a constrained optimum. Why not simply start close to zero? Figure 17.8 illustrates the difficulty with the trivial model (17.35).

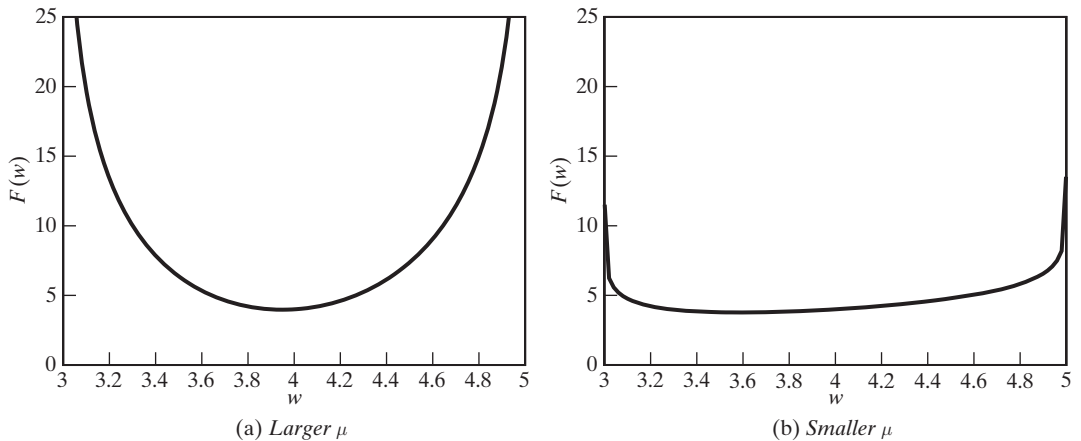


FIGURE 17.8 Effect of Barrier Multiplier on Unconstrained Tractability

When μ is comparatively large, a strong barrier keeps the search far from boundary areas, where it might bog down. But small μ 's allow the search to approach the boundary.

The best strategy is to start big and reduce the multiplier slowly.

Principle 17.39 When addressing a constrained nonlinear program by barrier methods, the multiplier μ should be started at a relatively high value > 0 and decreased as the search proceeds.

Sequential Unconstrained Barrier Technique

Formalization of principle [17.39](#)'s strategy for slowly decreasing the barrier multiplier μ produces the **sequential unconstrained barrier technique** of Algorithm 17B. Processing begins at an interior feasible $\mathbf{x}^{(0)}$ where none of the constraints are active. Multiplier μ starts relatively large and becomes smaller with each search. For each value μ , unconstrained barrier problem [17.35](#) is solved beginning with the optimum of the preceding search. We stop when μ is sufficiently close to zero. (Readers may wish to compare with barrier methods for linear programming in Section 7.4.)

Table 17.4 illustrates this for service desk design model (17.32) and corresponding barrier form (17.36). Unlike penalty methods, which can start anywhere, a barrier search must begin at an interior feasible point of the constrained NLP. Here we employ

$$\mathbf{x}^{(0)} = (3.52, 2.77)$$

ALGORITHM 17B: SEQUENTIAL UNCONSTRAINED BARRIER TECHNIQUE

Step 0: Initialization. Form barrier function [17.35], and choose initial barrier multiplier $\mu_0 > 0$ relatively large and feasible interior starting solution $\mathbf{x}^{(0)}$. Also initialize solution index $t \leftarrow 0$, and pick a reduction factor $\beta < 1$.

Step 1: Unconstrained Optimization. Beginning from $\mathbf{x}^{(t)}$, solve barrier optimization problem [17.35] with $\mu = \mu_t$, to produce optimum $\mathbf{x}^{(t+1)}$.

Step 2: Stopping. If μ is sufficiently small, stop and output $\mathbf{x}^{(t+1)}$.

Step 3: Reduce. Decrease the penalty multiplier as

$$\mu_{t+1} \leftarrow \beta \mu_t$$

Then advance $t \leftarrow t + 1$, and return to Step 1.

TABLE 17.4 Sequential Barrier Solution of Service Desk Application

t	μ	$\mathbf{x}^{(t+1)}$	t	μ	$\mathbf{x}^{(t+1)}$
0	4	(3.573, 2.794)	2	1	(3.608, 2.769)
1	2	(3.588, 2.786)	3	$\frac{1}{2}$	(3.629, 2.752)

The bulk of the computation involves a sequence of unconstrained barrier model searches with decreasing multipliers μ . Table 17.4 shows the first used $\mu_0 = 4$ to obtain the unconstrained optimum

$$\mathbf{x}^{(1)} = (3.573, 2.794)$$

Then the multiplier was decreased by a factor of $\beta = \frac{1}{2}$ and a new search initiated from $\mathbf{x}^{(1)}$. The resulting $\mathbf{x}^{(2)}$ starts a third search after μ is halved again. Searches continue until μ is close enough to zero that an approximate optimum is at hand. Then we stop with constrained optimum $\mathbf{x}^* = (3.63, 2.75)$.

17.6 REDUCED GRADIENT ALGORITHMS

Chapter 5's simplex algorithm for linear programming is the most widely employed of all optimization procedures. In this section we develop natural extensions to the nonlinear case known as **reduced gradient** algorithms, or more generally as the **variable elimination method**.

Standard Form for NLPs with Linear Constraints

Most of our development of reduced gradient methods will assume a linearly constrained nonlinear program. In particular, we assume that constraints have the standard form like Section 5.1's definition [5.6].

Definition 17.40 **Standard form** for linearly constrained nonlinear programs is

$$\begin{array}{ll} \min \text{ or } \max & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

We will also assume for simplicity that the rows of matrix \mathbf{A} are linearly independent, which can always be achieved by dropping redundant constraints.

APPLICATION 17.8: FILTER TUNING

For a tiny example on which to illustrate reduced gradient notions, consider the tuning of an electronic filter. Two parameters

$x_1 \triangleq$ value of the first tuning parameter

$x_2 \triangleq$ value of the second tuning parameter

must be chosen to minimize distortion

$$f(x_1, x_2) \triangleq (x_1 - 5)^2 - 2x_1x_2 + (x_2 - 10)^2$$

with x_1 in the range $[0, 3]$, x_2 in the range $[0, 5]$, and their total at most 6. The result is the linearly constrained nonlinear program

$$\begin{array}{ll} \min & f(x_1, x_2) \triangleq (x_1 - 5)^2 - 2x_1x_2 + (x_2 - 10)^2 \\ \text{s.t.} & x_1 + x_2 \leq 6 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 5 \end{array} \quad (17.37)$$

Figure 17.9 graphs this tuning model. A global optimum occurs at $\mathbf{x}^{(3)} = (1.75, 4.25)$.

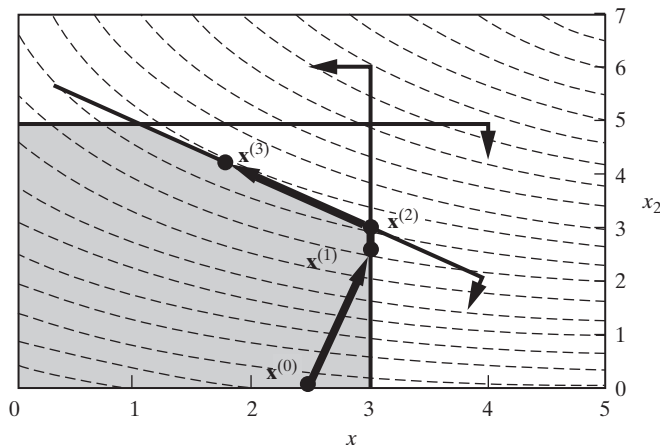


FIGURE 17.9 Reduced Gradient Search of Filter Tuning Application

We may convert to standard form [17.40] by adding slacks $x_3, x_4,$ and x_5 :

$$\begin{aligned}
 \min \quad & f(x_1, x_2) \triangleq (x_1 - 5)^2 - 2x_1x_2 + (x_2 - 10)^2 \\
 \text{s.t.} \quad & +x_1 + x_2 + x_3 = 6 \\
 & +x_1 + x_4 = 3 \\
 & +x_2 + x_5 = 5 \\
 & x_1, x_2, x_3, x_4, x_5 \geq 0
 \end{aligned} \tag{17.38}$$

Conditions for Feasible Directions with Linear Constraints

Any improving search algorithm attempts to move along improving feasible directions. Principle [3.25] of Section 3.3 has already detailed the requirements for maintaining feasibility with standard-form linear constraints.

Principle 17.41 At \mathbf{x} feasible for standard-form linear constraints [17.40], $\Delta \mathbf{x}$ is a feasible direction if and only if

$$\begin{aligned}
 \mathbf{A} \Delta \mathbf{x} &= \mathbf{0} \\
 \Delta x_j &\geq 0 \quad \text{for all } j \text{ with } x_j = 0
 \end{aligned}$$

To illustrate, consider point $\mathbf{x}^{(0)}$ at (2.5, 0) of Tuning Application Figure 17.9. Including corresponding values for slacks, the full standard-form solution is

$$\mathbf{x}^{(0)} = (2.5, 0, 3.5, 0.5, 5)$$

The only active inequality is the nonnegativity constraint on x_2 . Thus the corresponding conditions [17.41] for a feasible direction $\Delta \mathbf{x}$ are

$$\begin{aligned}
 + \Delta x_1 + \Delta x_2 + \Delta x_3 &= 0 \\
 + \Delta x_1 + \Delta x_4 &= 0 \\
 + \Delta x_2 + \Delta x_5 &= 0 \\
 \Delta x_2 &\geq 0
 \end{aligned} \tag{17.39}$$

Bases of the Main Linear Equalities

Section 5.2 developed the idea of **bases** or basic column sets of the matrix \mathbf{A} . Bases are maximal sets of linearly independent columns.

The important characteristic of a basis, and the corresponding **basic variables**, is that we can solve for the values of basic variables once all other, **nonbasic variables** have been fixed. That is, we can view the basics as functions of the nonbasics.

Principle 17.42 Identification of a basic set of variables in a system of linear equations partitions solutions into independent, nonbasic versus dependent, basic components.

Basic, Nonbasic, and Superbasic Variables

The simplex algorithms of Chapter 5 proceed through **basic solutions** in which all nonbasic variables take on lower bound value zero. We saw in Section 5.2 how this restriction can lead to a search through extreme points of the feasible set.

With nonlinear models, we know that an optimal solution may very well fall in the interior of the feasible set, or at a nonextreme point of the boundary. This does not change the fact that basic variables are dependent on our choice of nonbasics. It only means that nonbasics cannot be restricted to $= 0$. A new **superbasic** category of variables arises, which are nonbasics at positive value.

Principle 17.43 | Reduced gradient algorithms classify variables as basics, nonbasics at bound value zero, and superbasics nonbasic at values > 0 .

To illustrate, return to tuning application standard form (17.38), and choose x_1 , x_3 , and x_5 basic. Initial solution

$$\mathbf{x}^{(0)} = (2.5, 0, 3.5, 0.5, 5)$$

of Figure 17.9 is implied by nonbasic values $x_2^{(0)} = 0.0$ and $x_4^{(0)} = 0.5$. Setting all nonbasics $= 0$ would produce an extreme point of the feasible set. But with $x_4^{(0)} > 0$, and thus superbasic, nonextreme $\mathbf{x}^{(0)}$ can be represented.

EXAMPLE 17.17: DISTINGUISHING BASIC, NONBASIC, AND SUPERBASIC

Consider the standard-form nonlinear program

$$\begin{aligned} \max \quad & f(\mathbf{w}) \triangleq 50 - (w_1)^2 + 6w_1 - (w_2)^2 + 6w_2 + w_3 \\ \text{s.t.} \quad & + w_1 - w_2 + 3w_3 = 1 \\ & + 3w_1 + 2w_2 + 2w_4 = 6 \\ & w_1, w_2, w_3, w_4 \geq 0 \end{aligned}$$

- (a) Show that w_3 and w_4 form a basic set of variables.
 (b) Assuming the basis of part (a), classify remaining variables as nonbasic at lower bound or superbasic for the solution $\mathbf{w} = (0, 2, 1, 1)$.

Solution:

- (a) A basis is a maximal set of linearly independent vectors. The columns of w_3 and w_4 are linearly independent because the corresponding matrix

$$\mathbf{B} = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$$

has nonzero determinant $= 6$, and two is the maximum number of linearly independent 2-vectors.

- (b) The specified solution has w_1 nonbasic at value zero, and w_2 superbasic because its value is positive.

Maintaining Equalities by Solving Main Constraints for Basic Variables

Return now to the feasibility requirements of [17.41]. Let \mathbf{B} be a submatrix of \mathbf{A} formed by a basic set of columns, and \mathbf{N} the submatrix of all other columns. Then divide direction vector $\Delta \mathbf{x}$ into corresponding parts denoted $\Delta \mathbf{x}^{(B)}$ for components on columns in \mathbf{B} and $\Delta \mathbf{x}^{(N)}$ for those in \mathbf{N} .

Conditions [17.41] require that

$$\mathbf{A} \Delta \mathbf{x} = \mathbf{B} \Delta \mathbf{x}^{(B)} + \mathbf{N} \Delta \mathbf{x}^{(N)} = \mathbf{0}$$

Solving for basic components as a function of nonbasic variables,

$$\Delta \mathbf{x}^{(B)} = -\mathbf{B}^{-1} \mathbf{N} \Delta \mathbf{x}^{(N)} \tag{17.40}$$

Thus we can find a direction maintaining the equalities by choosing directional components for nonbasic variables and solving for basics as in (17.40).

Principle 17.44 Direction $\Delta \mathbf{x} \triangleq (\Delta \mathbf{x}^{(B)}, \Delta \mathbf{x}^{(N)})$ maintains feasibility in standard-form equality constraints $\mathbf{A} \mathbf{x} = \mathbf{b}$ if

$$\Delta \mathbf{x}^{(B)} = -\mathbf{B}^{-1} \mathbf{N} \Delta \mathbf{x}^{(N)}$$

where \mathbf{B} is a basis submatrix of $\mathbf{A} \triangleq (\mathbf{B}, \mathbf{N})$.

For example, solving feasibility conditions (17.39) of our tuning application for basic variable components Δx_1 , Δx_2 , and Δx_3 produces

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_3 \\ \Delta x_5 \end{pmatrix} = - \begin{pmatrix} 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \Delta x_2 \\ \Delta x_4 \end{pmatrix} \tag{17.41}$$

EXAMPLE 17.18: SOLVING FOR BASIC DIRECTION COMPONENTS

Return to the maximizing nonlinear program of Example 17.17 and express basic components of a move direction at $\mathbf{w} = (0, 2, 1, 1)$ as a function of the nonbasic components so that the resulting direction is feasible for the main constraints.

Solution: We apply principle [17.44] in solving $\mathbf{A} \Delta \mathbf{w} = \mathbf{0}$ conditions for basic components. The result is

$$\begin{aligned} \Delta w_3 &= -\left(\frac{1}{3} \Delta w_1 - \frac{1}{3} \Delta w_2\right) \\ \Delta w_4 &= -\left(\frac{3}{2} \Delta w_1 + 1 \Delta w_2\right) \end{aligned}$$

Active Nonnegativities and Degeneracy

The second part of feasible direction conditions [17.41] requires that $\Delta x_j \geq 0$ whenever the corresponding nonnegativity constraint $x_j \geq 0$ is active. As with the simplex algorithm (see Section 5.6), we will not strictly enforce these requirements on

basic j . Algorithms developed below endeavor to keep basic variables strictly positive, that is,

$$x_j > 0 \quad \text{for all } j \in B \quad (17.42)$$

For example, basic components $j = 1, 3, 5$ of tuning example solution $\mathbf{x}^{(0)} = (2.5, 0, 3.5, 0.5, 5)$ are all positive.

Under this **nondegeneracy assumption**, only nonnegativity constraints for nonbasics can be active, and we enforce

$$\Delta x_j \geq 0 \quad \text{for all } j \in N \quad \text{with } x_j = 0 \quad (17.43)$$

Reduced Gradients

We now know from principle [17.44](#) how to produce a feasible direction by limiting independent choices to components for a nonbasic set of variables. It remains to construct a feasible direction that improves the objective.

For small steps the change in objective along direction $\Delta \mathbf{x}$ is the first-order Taylor approximation term

$$\nabla f(\mathbf{x}) \cdot \Delta \mathbf{x}$$

Subdividing the gradient $\nabla f(\mathbf{x})$ into basic and nonbasic parts ($\nabla f(\mathbf{x})^{(B)}, \nabla f(\mathbf{x})^{(N)}$), we can eliminate the basic components to see fully the impact of choices for the nonbasics:

$$\begin{aligned} \nabla f(\mathbf{x}) \cdot \Delta \mathbf{x} &= \nabla f(\mathbf{x})^{(B)} \cdot \Delta \mathbf{x}^{(B)} + \nabla f(\mathbf{x})^{(N)} \cdot \Delta \mathbf{x}^{(N)} \\ &= \nabla f(\mathbf{x})^{(B)} \cdot (-\mathbf{B}^{-1}\mathbf{N}\Delta \mathbf{x}^{(N)}) + \nabla f(\mathbf{x})^{(N)} \cdot \Delta \mathbf{x}^{(N)} \\ &= (\nabla f(\mathbf{x})^{(N)} - \nabla f(\mathbf{x})^{(B)}\mathbf{B}^{-1}\mathbf{N})\Delta \mathbf{x}^{(N)} \end{aligned} \quad (17.44)$$

These derived coefficients on directional components are called the **reduced gradient**.

Definition 17.45 | The reduced gradient associated with basis matrix \mathbf{B} at current solution \mathbf{x} is $\mathbf{r} \triangleq (\mathbf{r}^{(B)}, \mathbf{r}^{(N)})$ with

$$\begin{aligned} \mathbf{r}^{(B)} &\triangleq \mathbf{0} \\ \mathbf{r}^{(N)} &\triangleq \nabla f(\mathbf{x})^{(N)} - \nabla f(\mathbf{x})^{(B)}\mathbf{B}^{-1}\mathbf{N} \end{aligned}$$

To illustrate, return to tuning model (17.38) at $\mathbf{x}^{(0)} = (2.5, 0, 3.5, 0.5, 5)$. The corresponding gradient is

$$\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 2(x_1 - 5) - 2x_2 \\ -2x_1 + 2(x_2 - 10) \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -5 \\ -25 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Now using system (17.41), which derives $\mathbf{B}^{-1}\mathbf{N}$ to express feasible direction conditions with $x_1, x_3,$ and x_5 basic,

$$\mathbf{r}^{(B)} = (r_1, r_3, r_5) = (0, 0, 0) \tag{17.45}$$

and

$$\begin{aligned} \mathbf{r}^{(N)} &= (r_2, r_4) \\ &= \nabla f^{(N)} - \nabla f^{(B)}(\mathbf{B}^{-1}\mathbf{N}) \\ &= (-25, 0) - (-5, 0, 0) \begin{pmatrix} 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \\ &= (-25, 5) \end{aligned} \tag{17.46}$$

EXAMPLE 17.19: COMPUTING REDUCED GRADIENTS

Return to the nonlinear program of Examples 17.17 and 17.18 with basis $B = \{3, 4\}$. Compute the corresponding reduced gradient at solution $\mathbf{w} = (0, 2, 1, 1)$.

Solution: With objective function

$$f(\mathbf{w}) \triangleq 50 - (w_1)^2 + 6w_1 - (w_2)^2 + 6w_2 + w_3$$

the gradient at the \mathbf{w} specified is

$$\nabla f(\mathbf{w}) = \begin{pmatrix} -2w_1 + 6 \\ -2w_2 + 6 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \\ 1 \\ 0 \end{pmatrix}$$

Now applying definition [17.45], basic components of the reduced gradient become

$$r_3 = r_4 = 0$$

Corresponding nonbasic components are

$$\begin{aligned} r_1 &= 6 - (1, 0) \cdot \left(\frac{1}{3}, \frac{3}{2}\right) = \frac{17}{3} \\ r_2 &= 2 - (1, 0) \cdot \left(-\frac{1}{3}, 1\right) = \frac{7}{3} \end{aligned}$$

Reduced Gradient Move Direction

Reduced gradient algorithms seek to move nonbasics in reduced gradient direction $\Delta \mathbf{x}^{(N)} = \pm \mathbf{r}^{(N)}$ (+for maximize models, - for minimize models). However, some adjustment must be made to avoid decreasing any x_j already = 0 [i.e., to enforce feasibility requirement (17.43)].

Principle 17.46 | The reduced gradient algorithm moves from feasible point \mathbf{x} in direction $\Delta\mathbf{x}$ derived from reduced gradient [17.45] as (+ to maximize, – to minimize)

$$\Delta x_j \leftarrow \begin{cases} \pm r_j & \text{if } \pm r_j > 0 \text{ or } x_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

on nonbasic components $j \in N$ and

$$\Delta\mathbf{x}^{(B)} \leftarrow -\mathbf{B}^{-1}\mathbf{N}\Delta\mathbf{x}^{(N)}$$

for basics.

In the minimizing tuning application of (17.45)–(17.46), construction [17.46] makes nonbasic components

$$\Delta x_2 = -r_2 = 25 \quad \text{and} \quad \Delta x_4 = -r_4 = -5 \quad (17.47)$$

Corresponding basic components are derived from expression (17.41) as

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_3 \\ \Delta x_5 \end{pmatrix} = -\begin{pmatrix} 0 & 1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 25 \\ -5 \end{pmatrix} = \begin{pmatrix} 5 \\ -30 \\ -25 \end{pmatrix} \quad (17.48)$$

We have constructed direction $\Delta\mathbf{x}$ of [17.46] to be feasible (assuming all basics positive). If $\Delta\mathbf{x} = \mathbf{0}$, the current \mathbf{x} can be shown to be a KKT point, and the algorithm stops. Otherwise [using (17.44) and [17.45]],

$$\begin{aligned} \nabla f(\mathbf{x}) \cdot \Delta\mathbf{x} &= \mathbf{r}^{(N)} \cdot \Delta\mathbf{x}^{(N)} \\ &= \sum_{\Delta x_j = \pm r_j \neq 0} (r_j)(\pm r_j) \end{aligned}$$

shows that $\Delta\mathbf{x}$ has the proper sign to be an improving direction.

EXAMPLE 17.20: CONSTRUCTING REDUCED GRADIENT DIRECTIONS

Return to the nonlinear program of Examples 17.17 to 17.19. Construct the move direction that would be pursued by the reduced gradient algorithm at solution $\mathbf{w} = (0, 2, 1, 1)$.

Solution: Using reduced gradient results of Example 17.19 and construction [17.46], nonbasic components for this maximizing model are

$$\Delta w_1 = +r_1 = \frac{17}{3} \quad \text{and} \quad \Delta w_2 = +r_2 = \frac{7}{3}$$

Then, basic components derive from the representation of Example 17.18 as

$$\Delta w_3 = -\left(\frac{1}{3} \Delta w_1 - \frac{1}{3} \Delta w_2\right) = -\frac{10}{9}$$

$$\Delta w_4 = -\left(\frac{3}{2} \Delta w_1 + 1 \Delta w_2\right) = -\frac{65}{6}$$

Line Search in Reduced Gradient Methods

Having specified reduced gradient direction [17.46], we next need to decide how far to follow it. As with most nonlinear methods, a line search will be required to determine the largest λ for which the direction improves. However, constraints add feasibility considerations. Our direction $\Delta \mathbf{x}$ will satisfy equality constraints $\mathbf{A}(\mathbf{x} + \lambda \Delta \mathbf{x}) = \mathbf{b}$ for arbitrarily large λ , but nonnegativity constraints cannot be ignored. The line search must be limited by the same sort of “minimum ratio” check employed in LP algorithms of Chapters 5, 6 and 7.

Principle 17.47 The step size λ applied at each step of the reduced gradient algorithm is determined by the one-dimensional optimization

$$\begin{aligned} &\min \text{ or } \max && f(\mathbf{x} + \lambda \Delta \mathbf{x}) \\ &\text{s. t.} && 0 \leq \lambda \leq \lambda_{\max} \end{aligned}$$

where \mathbf{x} is the current point, $\Delta \mathbf{x}$ is the move direction, and λ_{\max} is the maximum feasible step

$$\lambda_{\max} = \min \left\{ \frac{x_j}{-\Delta x_j} : \Delta x_j < 0 \right\}$$

For example, tuning example direction (17.47)–(17.48) is negative for components $j = 3, 4, 5$. Thus the maximum feasible step at $\mathbf{x}^{(0)} = (2.5, 0, 3.5, 0.5, 5)$ is

$$\lambda_{\max} = \min \left\{ \frac{3.5}{30}, \frac{5}{5}, \frac{5}{25} \right\} = 0.1 \tag{17.49}$$

Distortion function $f(\mathbf{x}^{(0)} + \lambda \Delta \mathbf{x})$ decreases for all $\lambda \in [0, 0.1]$, so that the step size chosen will be the full $\lambda = 0.1$.

EXAMPLE 17.21: COMPUTING MAXIMUM FEASIBLE STEPS

Example 17.20 computed reduced gradient move direction

$$\Delta \mathbf{w} = \left(\frac{17}{3}, \frac{7}{3}, -\frac{10}{9}, -\frac{65}{6} \right) \text{ at solution } \mathbf{w} = (0, 2, 1, 1)$$

of a standard-form, linearly constrained NLP. Determine the maximum feasible step in this direction.

Solution: For a standard-form model with linear equality main constraints, the only possible loss of feasibility occurs when some variable drops to its lower bound of 0.

This occurs here at (principle [17.47])

$$\lambda_{\max} = \min \left\{ \frac{1}{10/9}, \frac{1}{65/6} \right\} = \frac{6}{65}$$

Basis Changes in Reduced Gradient Methods

One final issue relates to nondegeneracy assumption (17.42). All our analysis has been based on basic variables always having positive values. As with the simplex algorithms of Chapter 5 (see Section 5.6), such nondegeneracy cannot always be guaranteed. Still, it is sufficient for functioning of the reduced gradient algorithm that we replace a variable in the basis if the most recent move forced it to $= 0$.

With many nonbasics (and superbasics) changing during the move, it is not as easy as with simplex to decide which nonbasic should enter the basis. To be assured of keeping a basis, we need only be careful to select a nonbasic actually affecting the blocking basic in computation [17.44].

Principle 17.48 When movement in reduced gradient direction [17.46] is blocked by a nonnegativity constraint on a basic variable x_i , that variable should be replaced in the basis by a nonbasic x_j , preferably superbasic, such that the coefficient of $-\mathbf{B}^{-1}\mathbf{N}$ relating i and j in [17.44] is nonzero.

Superbasics are preferred because they have the positive value desired for a basic.

In the move along tuning application direction (17.47)–(17.48), the blocking variable of step computation (17.49) was nonbasic x_4 . No basis adjustment is required.

EXAMPLE 17.22: CHANGING THE BASIS IN REDUCED GRADIENT

The nonlinear programs of Examples 17.17 to 17.21 computed maximum feasible step size $\lambda_{\max} = \frac{65}{6}$ in the direction

$$\Delta \mathbf{w} = \left(\frac{17}{3}, \frac{7}{3}, -\frac{10}{9}, -\frac{65}{6} \right) \quad \text{at solution} \quad \mathbf{w} = (0, 2, 1, 1)$$

with w_3 and w_4 nonbasic. Assume that a full step $\lambda = \frac{65}{6}$ is chosen by the line search.

- Determine whether a basis change is now needed.
- If a change is required, select a new basis.

Solution: We apply principle [17.48].

- After a full step, the new solution will be

$$\mathbf{w} + \lambda \Delta \mathbf{w} = \left(\frac{34}{65}, \frac{144}{65}, \frac{35}{39}, 0 \right)$$

Since basic variable w_4 drops to 0, a basis change is required.

- We must replace w_4 in the basis with a nonbasic variable that influenced its value on this move—preferably one that is now superbasic. Reference back to Example 17.20 shows that Δw_4 was affected (had nonzero coefficients) by both non-basics. With both nonbasics now superbasic, we arbitrarily choose w_2 to produce new basis $\{w_2, w_3\}$.

Reduced Gradient Algorithm

All the building blocks of a reduced gradient search are now in place. Algorithm 17C provides details.

ALGORITHM 17C: REDUCED GRADIENT SEARCH

Step 0: Initialization. Choose stopping tolerance $\varepsilon > 0$ and any starting feasible solution $\mathbf{x}^{(0)}$. Then construct a corresponding basis B with as many basic $x_j^0 > 0$ as possible, and set solution index $t \leftarrow 0$.

Step 1: Reduced Gradient Direction. Compute reduced gradient \mathbf{r} at $\mathbf{x}^{(t)}$ as in [17.45], and use \mathbf{r} to generate move direction $\Delta\mathbf{x}^{t+1}$ per [17.46].

Step 2: Stopping. If $\|\Delta\mathbf{x}^{t+1}\| \leq \varepsilon$, stop and output local optimum $\mathbf{x}^{(t)}$.

Step 3: Feasibility Limit. Compute feasibility limiting step λ_{\max} according to [17.47] ($\lambda_{\max} = \infty$ if $\Delta\mathbf{x}^{t+1} \geq \mathbf{0}$).

Step 4: Line Search. Perform a one-dimensional optimization to determine λ_{t+1} solving

$$\begin{aligned} \min \text{ or } \max \quad & f(\mathbf{x} + \lambda\Delta\mathbf{x}^{t+1}) \\ \text{s.t.} \quad & 0 \leq \lambda \leq \lambda_{\max} \end{aligned}$$

Step 5: New Point. Advance

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda_{t+1}\Delta\mathbf{x}^{t+1}$$

Step 6: Basis Change. If any basic $x_j^{(t+1)} = 0$, replace one such j in the basis with some superbasic j' .

Step 7: Advance. Increment $t \leftarrow t + 1$, and return to Step 1.

Computation begins at any feasible point and a corresponding basis. Each iteration follows reduced gradient direction [17.46] until either objective progress stops or the feasibility limit is reached. Bases are changes as in [17.48] whenever a basic variable dropping to zero blocks progress. Termination occurs when the computed direction is sufficiently close to the zero vector.

Reduced Gradient Search of Filter Tuning Application

Figure 17.9 has already displayed the sequence of points visited by reduced gradient Algorithm 17C in solving our tuning application from initial point (2.5, 0). Table 17.5 provides details.

The first move of the search follows the directions (17.47)–(17.48) for a full step $\lambda_{\max} = 0.1$. No basis change is required because the blocking variable is nonbasic. Thus gradient computations are simply repeated to produce new direction

$$\Delta\mathbf{x} = (0, 21, -21, 0, -21)$$

Once again the direction improves all the way to maximum feasible step $\lambda_{\max} = 0.0238$. This time, however, the blocking variable is basic x_3 . Replacing x_3 in the basis with superbasic x_2 keeps basic variables strictly positive without losing linear independence of basic columns.

Recomputation produces the next move direction,

$$\Delta\mathbf{x} = (-10, 10, 0, 10, -10)$$

TABLE 17.5 Reduced Gradient Search of Filter Tuning Application

	x_1	x_2	x_3	x_4	x_5	
$\min f(\mathbf{x})$		$(x_1 - 5)^2 - 2x_1x_2 + (x_2 - 10)^2$				\mathbf{b}
\mathbf{A}	1	1	1	0	0	6
	1	0	0	1	0	3
	0	1	0	0	1	5
$t = 0$	B	N	B	N	B	
$\mathbf{x}^{(0)}$	2.5	0.0	3.5	0.5	5.0	$f(\mathbf{x}^{(0)}) = 106.25$
$\nabla f(\mathbf{x}^{(0)})$	-5.0	-25.0	0.0	0.0	0.0	
\mathbf{r}	0.0	-25.0	0.0	5.0	0.0	
$\Delta \mathbf{x}$	5.0	25.0	-30.0	-5.0	-25.0	$\lambda_{\max} = 0.1, \lambda = 0.1$
$t = 1$	B	N	B	N	B	
$\mathbf{x}^{(1)}$	3.0	2.5	0.5	0.0	2.5	$f(\mathbf{x}^{(1)}) = 42.25$
$\nabla f(\mathbf{x}^{(1)})$	-9.0	-21.0	0.0	0.0	0.0	
\mathbf{r}	0.0	-21.0	0.0	9.0	0.0	
$\Delta \mathbf{x}$	0.0	21.0	-21.0	0.0	-21.0	$\lambda_{\max} = 0.0238, \lambda = 0.0238$
$t = 2$	B	B	N	N	B	
$\mathbf{x}^{(2)}$	3.0	3.0	0.0	0.0	2.0	$f(\mathbf{x}^{(2)}) = 35.00$
$\nabla f(\mathbf{x}^{(2)})$	-10.0	-20.0	0.0	0.0	0.0	
\mathbf{r}	0.0	0.0	20.0	-10.0	0.0	
$\Delta \mathbf{x}$	-10.0	10.0	0.0	10.0	-10.0	$\lambda_{\max} = 0.2, \lambda = 0.125$
$t = 3$	B	B	N	N	B	
$\mathbf{x}^{(3)}$	1.75	4.25	0.0	1.25	0.75	$f(\mathbf{x}^{(3)}) = 28.75$
$\nabla f(\mathbf{x}^{(3)})$	-15.0	-15.0	0.0	0.0	0.0	
\mathbf{r}	0.0	0.0	15.0	0.0	0.0	
$\Delta \mathbf{x}$	0.0	0.0	0.0	0.0	0.0	Stop

Notice that $\Delta x_3 = 0$ even though $-r_3 = -20$, because decreasing x_3 would produce immediate infeasibility.

The maximum feasible step in the chosen $\Delta \mathbf{x}$ is $\lambda_{\max} = 0.2$. Still, a line search over $\lambda \in (0, 0.2]$ discovers a minimum at $\lambda = 0.125$. Thus the search advances only to

$$\mathbf{x}^{(3)} = (1.75, 4.25, 0, 1.25, 0.75)$$

This point proves (at least locally) optimal when $\Delta \mathbf{x} = \mathbf{0}$ computes as the next search direction.

Major and Minor Iterations in Reduced Gradient

At any point in a reduced gradient search, the superbasic variables represent a “free” set in that they can increase or decrease without losing feasibility. A refinement that has proved useful exploits this relative ease of movement by dividing the search into major iterations and minor iterations.

Definition 17.49 **Minor iterations** of reduced gradient procedures change only superbasic and basic variable values by adopting at $\Delta \mathbf{x}$ the move direction $\Delta \mathbf{x}$ with (+ for maximize, - for minimize)

$$\Delta x_j \leftarrow \begin{cases} \pm r_j & \text{if } x_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

for nonbasic components $j \in N$ and

$$\Delta \mathbf{x}^{(B)} \leftarrow -\mathbf{B}^{-1} \mathbf{N} \Delta \mathbf{x}^{(N)}$$

for basics. Major iterations follow construction [17.46] by also allowing changes in nonbasics = 0.

A minor iteration leaves nonbasics at bound zero fixed, changing only superbasics. When progress slows, we undertake a major iteration changing more nonbasics as in Algorithm 17C.

Second-Order Extensions of Reduced Gradient

Major/minor direction procedure [17.49] can be productively extended even further by employing second-order information on the objective function. Thinking of objective function $f(\mathbf{x})$ as a function of the superbasics alone, with other nonbasics fixed = 0, and basics taking implied values, we are left with an unconstrained optimization in the superbasics. Quasi-Newton methods of Section 16.7 can then be employed to quickly find a good choice of superbasic values. Afterward, having completed several minor iterations, we consider making other nonbasics positive.

Generalized Reduced Gradient Procedures for Nonlinear Constraints

To this point we have assumed that all constraints of the given nonlinear program are linear. **Generalized reduced gradient** algorithms can be developed that extend to nonlinear constraints.

Suppose that we are given the nonlinear equality-constrained standard form

$$\begin{aligned} \text{min or max} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) = b_i \quad i \in E \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{17.50}$$

Using first-order Taylor approximations, it is natural to consider linearizing constraints around a current $\mathbf{x}^{(t)}$ as

$$b_i = g_i(\mathbf{x}) \approx g_i(\mathbf{x}^{(t)}) + \nabla g_i(\mathbf{x}^{(t)}) \cdot (\mathbf{x} - \mathbf{x}^{(t)}) \tag{17.51}$$

Noting that feasibility implies $g_i(\mathbf{x}^{(t)}) = b_i$, this linearization simplifies to

$$\nabla g_i(\mathbf{x}^{(t)}) \cdot \mathbf{x} = \nabla g_i(\mathbf{x}^{(t)}) \cdot \mathbf{x}^{(t)} \quad \text{for all } i \in E \tag{17.52}$$

Including nonnegativity constraints with system (17.52) yields linear-constrained format

$$\begin{aligned}\mathbf{A}^{(t)}\mathbf{x} &= \mathbf{b}^{(t)} \\ \mathbf{x} &\geq \mathbf{0}\end{aligned}$$

with rows of $\mathbf{A}^{(t)}$ being $\nabla g_i(\mathbf{x}^{(t)})$ and components of $\mathbf{b}^{(t)}$ equaling $\nabla g_i(\mathbf{x}^{(t)}) \cdot \mathbf{x}^{(t)}$. We are now in a position to employ linear-constrained reduced gradient Algorithm 17C (or its second-order extensions).

Dealing with successive systems (17.52) is essentially the strategy of generalized reduced gradient algorithms. Still, there is a difficulty. Approximation (17.51) is not exact for nonlinear constraints. Thus enforcement of (17.52) is not guaranteed to keep \mathbf{x} feasible.

Generalized reduced gradient algorithms address this difficulty by following each reduced gradient move with **corrector steps** to restore feasibility. In essence, a penalty function is introduced (see Section 17.5) and a new move is chosen to minimize the penalized objective function. Once feasibility is restored, a new move can be computed using equations (17.52).

17.7 QUADRATIC PROGRAMMING METHODS

A constrained nonlinear program is a **quadratic program** or **QP** if its objective function is quadratic and all its constraints are linear (definition [17.9](#)). In this section we investigate special methods adapted to this class of NLPs.

General Symmetric Form of Quadratic Programs

It will be useful to express quadratic programs in general symmetric form.

Definition 17.50 Quadratic programs can be placed in the **general symmetric form**

$$\begin{aligned}\max \text{ or } \min & f(\mathbf{x}) \triangleq c_0 + \mathbf{c} \cdot \mathbf{x} + \mathbf{x}\mathbf{Q}\mathbf{x} \\ \text{s.t.} & \mathbf{a}^{(i)}\mathbf{x} \geq b_i && \text{for all } i \in G \\ & \mathbf{a}^{(i)}\mathbf{x} \leq b_i && \text{for all } i \in L \\ & \mathbf{a}^{(i)}\mathbf{x} = b_i && \text{for all } i \in E\end{aligned}$$

where \mathbf{Q} is a symmetric matrix, and sets G , L , and E index the \geq , \leq , and $=$ constraints, respectively.

Notice that nonnegativity and other variable-type restrictions are treated as main constraints.

The assumption that Q is symmetric ($= \mathbf{Q}^T$) merely simplifies notation. There is no loss of generality because a model with asymmetric $\bar{\mathbf{Q}}$ has the same objective value as one with symmetric

$$\mathbf{Q} = \frac{1}{2} (\bar{\mathbf{Q}} + \bar{\mathbf{Q}}^T)$$

Quadratic Program Form of the Filter Tuning Application

We illustrate quadratic programming methods with the tiny distortion tuning application model (17.53) (Section 17.7). In vector format [17.50], the model is

$$\begin{aligned}
 \min \quad & 125 + (-10, -20) \cdot \mathbf{x} + \mathbf{x} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \mathbf{x} \\
 \text{s.t.} \quad & (1, 0) \cdot \mathbf{x} \geq 0 \\
 & (0, 1) \cdot \mathbf{x} \geq 0 \\
 & (1, 1) \cdot \mathbf{x} \leq 6 \\
 & (1, 0) \cdot \mathbf{x} \leq 3 \\
 & (0, 1) \cdot \mathbf{x} \leq 5
 \end{aligned} \tag{17.53}$$

With $G = \{1, 2\}$, $L = \{3, 4, 5\}$, and $E = \emptyset$.

EXAMPLE 17.23: UNDERSTANDING STANDARD QP NOTATION

Return to the quadratic program used in Examples 17.17–17.19 of Section 17.6:

$$\begin{aligned}
 \max \quad & f(\mathbf{w}) \triangleq 50 - (w_1)^2 + 6w_1 - (w_2)^2 + 6w_2 + w_3 \\
 \text{s.t.} \quad & + w_1 - w_2 + 3w_3 = 1 \\
 & +3w_1 + 2w_2 + 2w_4 = 6 \\
 & w_1, w_2, w_3, w_4 \geq 0
 \end{aligned}$$

Identify elements c_0 , \mathbf{c} , \mathbf{Q} , G , L , E , $\mathbf{a}^{(i)}$, and b_i of general form [17.50].

Solution: Arranging objective function elements in matrix form [17.50] yields $c_0 = 50$,

$$\mathbf{c} = \begin{pmatrix} 6 \\ 6 \\ 1 \\ 0 \end{pmatrix} \text{ and } \mathbf{Q} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

With $E = \{1, 2\}$, $G = \{3, 4, 5, 6\}$, and $L = \emptyset$, corresponding constraint coefficients are

$$\begin{aligned}
 \mathbf{a}^{(1)} &= (1, -1, 3, 0), & b_1 &= 1 \\
 \mathbf{a}^{(2)} &= (3, 2, 0, 2), & b_2 &= 6 \\
 \mathbf{a}^{(3)} &= (1, 0, 0, 0), & b_3 &= 0 \\
 \mathbf{a}^{(4)} &= (0, 1, 0, 0), & b_4 &= 0 \\
 \mathbf{a}^{(5)} &= (0, 0, 1, 0), & b_5 &= 0 \\
 \mathbf{a}^{(6)} &= (0, 0, 0, 1), & b_6 &= 0
 \end{aligned}$$

Equality-Constrained Quadratic Programs and KKT Conditions

It is instructive to begin our investigation of quadratic programming with the pure equality case:

$$\begin{array}{ll} \max \text{ or } \min & f(\mathbf{x}) \triangleq \mathbf{c}\mathbf{x} + \mathbf{x}\mathbf{Q}\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \end{array} \quad (17.54)$$

Here $G = L = \emptyset$ in general form [17.50], and coefficient vectors $\mathbf{a}^{(i)}$ for equalities $i \in E$ have been collected as rows of a matrix \mathbf{A} .

With all constraints equalities, Karush–Kuhn–Tucker conditions (principle [17.21]) for model (17.54) require no sign restrictions or complementary slackness constraints. Furthermore, the objective function gradient (\mathbf{Q} symmetric) is

$$\nabla f(\mathbf{x}) = \mathbf{c} + 2\mathbf{Q}\mathbf{x}$$

and constraint gradients $\nabla g_i(\mathbf{x})$ are the rows of \mathbf{A} . Thus KKT conditions for model (17.54) reduce to

$$\sum_i \mathbf{a}^{(i)} v_i = \mathbf{c} + 2\mathbf{Q}\mathbf{x}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

What makes pure-equality quadratic programs special is that these conditions can be rearranged into a square system of linear equations.

Principle 17.51 Karush–Kuhn–Tucker optimality conditions for pure equality quadratic programs (17.54) are the linear equations

$$\begin{pmatrix} -2\mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \mathbf{b} \end{pmatrix}$$

EXAMPLE 17.24: FORMING KKT CONDITIONS FOR EQUALITY QPs

Form Karush–Kuhn–Tucker optimality conditions for the equality-constrained quadratic program

$$\begin{array}{ll} \min & 4(y_1)^2 - 6y_1y_2 + 5(y_2)^2 + y_3 \\ \text{s.t.} & + y_1 - 3y_2 - 9y_3 = 11 \\ & -y_1 + 7y_2 + 7y_3 = -9 \end{array}$$

at $\mathbf{y} = (2, 0, -1)$.

Solution: Here

$$\mathbf{A} = \begin{pmatrix} 1 & -3 & -9 \\ -1 & 7 & 7 \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{Q} = \begin{pmatrix} 4 & -3 & 0 \\ -3 & 5 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Thus KKT conditions [17.51](#) are

$$\begin{pmatrix} -8 & 6 & 0 & 1 & -1 \\ 6 & -10 & 0 & -3 & 7 \\ 0 & 0 & 0 & -9 & 7 \\ 1 & -3 & -9 & 0 & 0 \\ -1 & 7 & 7 & 0 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 11 \\ -9 \end{pmatrix}$$

Direct Solution of KKT Conditions for Quadratic Programs

The unusually simple form of KKT conditions [17.51](#) for equality-constrained quadratic programs suggests an approach to solution. We could simply form the KKT system of linear equations and solve for KKT point \mathbf{x} and corresponding Lagrange multipliers \mathbf{v} .

This is the approach taken in many methods.

Principle 17.52 Equality-constrained quadratic programs are often approached by direct solution of (linear) Karush–Kuhn–Tucker conditions [17.51](#).

Sophisticated methods of linear algebra may be used to compute answers, but the process remains essentially one solving the KKT system.

Unique solvability of system [17.51](#) would mean that model (17.54) has a unique KKT point. Since equality constraints assure that every local optimum is a KKT point (principle [17.27](#)), a unique [17.51](#) solution must correspond to a unique local (and thus global) maximum or minimum unless the model has no extrema at all. Other cases may have multiple KKT points, or none at all. Still, any local optimum must be a solution to system [17.51](#).

EXAMPLE 17.25: SOLVING KKT CONDITIONS FOR EQUALITY QPs

Solve the KKT conditions of the equality-constrained quadratic program in Example 17.24 to find a KKT point of the model.

Solution: The unique solution to this KKT system has primal solution

$$y_1 = -0.0834, \quad y_2 = -0.0992, \quad y_3 = -1.1984$$

and corresponding Lagrange multipliers

$$v_1 = -0.2486, \quad v_2 = -0.1768$$

Further analysis would be required to determine whether this KKT solution is a global maximum, a global minimum, or a saddle point.

Active Set Strategies for Quadratic Programming

Active set methods exploit the linear equation form of KKT conditions for equality constrained QPs by reducing general quadratic programs [17.50] to a sequence of equality cases. To see how, define

$S \triangleq$ set of indices of active constraints at current feasible solution

$\mathbf{x}^{(t)}$ in general QP model [17.50]

$\mathbf{A}_S \triangleq$ matrix with rows formed by the coefficient vectors

$\mathbf{a}^{(i)}$ of $i \in S$

Every equality constraint of E belongs to S , along with active inequalities of G and L .

Suppose that we require all active constraints $i \in S$ to continue being satisfied as equalities during our next move. Then an optimal move $\Delta \mathbf{x}$ from $\mathbf{x}^{(t)}$ should solve

$$\begin{aligned} \max \text{ or } \min \quad & f(\mathbf{x}^{(t)} + \Delta \mathbf{x}) = f(\mathbf{x}^{(t)}) + \nabla f(\mathbf{x}^{(t)}) \cdot \Delta \mathbf{x} + \Delta \mathbf{x} \mathbf{Q} \Delta \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}_S \Delta \mathbf{x} = \mathbf{0} \end{aligned} \quad (17.55)$$

The (17.55) objective merely rewrites $f(\mathbf{x}^{(t)} + \Delta \mathbf{x})$ in terms of the second-order Taylor representation (definition [16.17]), which is exact for quadratic functions. Constraints enforce the familiar requirements $\sum a_{i,j} \Delta x_k = 0$ (principle [3.25]) for a move to preserve linear equality constraints.

Notice that subproblem (17.55) is now in equality-constrained format (17.54). Thus we can compute a move $\Delta \mathbf{x}$ by solving the corresponding KKT linear equations [17.51].

Definition 17.53 Active set methods for general quadratic programs compute the move $\Delta \mathbf{x}$ at current solution $\mathbf{x}^{(t)}$ by solving Karush–Kuhn–Tucker conditions

$$\begin{pmatrix} -2\mathbf{Q} & \mathbf{A}_S^T \\ \mathbf{A}_S & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{v}^{(S)} \end{pmatrix} = \begin{pmatrix} \nabla f(\mathbf{x}^{(t)}) \\ \mathbf{0} \end{pmatrix}$$

where \mathbf{A}_S is the coefficient matrix of active constraints and $\mathbf{v}^{(S)}$ is the corresponding Lagrange multiplier vector. All v_i for $i \notin S$ are fixed = 0.

As with models having only equality constraints, a KKT solution to system [17.53] may not exist, or not correspond to the desired minimum for a minimize problem or maximum for a maximize problem over the active constraints. Still, definition [17.53] provides good results when the objective function is reasonably well behaved.

To illustrate, return to tuning model (17.53), which is a convex program. At $\mathbf{x}^{(0)} = (2.5, 0)$, only nonnegativity constraint $i = 2$ is active, so

$$S = \{2\} \quad \text{and} \quad \mathbf{A}_S = (0, 1)$$

With $\nabla f(\mathbf{x}^{(t)}) = (-5, -25)$, the corresponding move-finding KKT system ([17.53]) is

$$\begin{pmatrix} -2 & 2 & 0 \\ 2 & -2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} -5 \\ -25 \\ 0 \end{pmatrix}$$

This system has unique solution

$$\Delta x_1 = 2.5, \quad \Delta x_2 = 0, \quad v_2 = -30 \tag{17.56}$$

Step Size with Active Set Methods

If a move $\Delta \mathbf{x} \neq 0$ results from subproblem (17.55), the usual update

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta \mathbf{x}$$

optimizes the objective over the active constraints. However, we have ignored inactive constraints in forming (17.55). A full step in direction $\Delta \mathbf{x}$ may cause some such constraints to be violated.

To account for this possibility, we introduce a now-familiar maximum step rule:

Principle 17.54 | If the $\Delta \mathbf{x}$ computed from the active constraints at solution $\mathbf{x}^{(t)}$ is nonzero, active set algorithms adopt step λ in direction $\Delta \mathbf{x}$, where

$$\begin{aligned} \lambda_G &\leftarrow \min \left\{ \frac{\mathbf{a}^{(i)} \mathbf{x}^{(t)} - b_i}{-\mathbf{a}^{(i)} \Delta \mathbf{x}} : \mathbf{a}^{(i)} \Delta \mathbf{x} < 0, \quad i \in G \right\} \\ \lambda_L &\leftarrow \min \left\{ \frac{b_i - \mathbf{a}^{(i)} \mathbf{x}^{(t)}}{\mathbf{a}^{(i)} \Delta \mathbf{x}} : \mathbf{a}^{(i)} \Delta \mathbf{x} > 0, \quad i \in L \right\} \\ \lambda &\leftarrow \min \{ 1, \lambda_G, \lambda_L \} \end{aligned}$$

The first two possibilities for λ check inactive \geq and \leq constraints, respectively, and the 1 in the last step provides for the possibility that the full move is feasible. For example, we would compute the appropriate step in tuning application direction $\Delta \mathbf{x} = (2.5, 0)$ of (17.56) from $\mathbf{x}^{(0)} = (2.5, 0)$ as

$$\begin{aligned} \lambda_G &= +\infty \\ \lambda_L &= \min \left\{ \frac{3.5}{2.5}, \frac{0.5}{2.5} \right\} = 0.2 \\ \lambda &= \min \{ 1, +\infty, 0.2 \} = 0.2 \end{aligned} \tag{17.57}$$

EXAMPLE 17.26: CHOOSING STEP SIZE IN ACTIVE SET QP

Suppose that an active set search of a quadratic program with constraints

$$\begin{aligned} 2y_1 + 3y_2 &\geq 10 \\ 1y_1 + 7y_2 &\leq 40 \\ 1y_1 + 3y_2 &= 17 \end{aligned}$$

has reached $\mathbf{y} = (2, 5)$ and computed (definition 17.53) move $\Delta \mathbf{y} = (-3, 1)$.

- (a) Determine the appropriate step size λ to apply.
- (b) How would the λ change if the second constraint were $1y_1 + 7y_2 \leq 80$?

Solution: We apply rule [17.54].

(a) Only the last, equality constraint is active. Changes in the other constraints per unit step in direction $\Delta \mathbf{y}$ are

$$\mathbf{a}^{(1)} \cdot \Delta \mathbf{y} = (2, 3) \cdot (-3, 1) = -3$$

$$\mathbf{a}^{(2)} \cdot \Delta \mathbf{y} = (1, 7) \cdot (-3, 1) = 4$$

Thus

$$\lambda_G = \frac{19 - 10}{3}, \quad \lambda_L = \frac{40 - 37}{4}, \quad \lambda = \min \left\{ 1, 3, \frac{3}{4} \right\} = \frac{3}{4}$$

(b) With this revised right-hand side,

$$\lambda_G = 3, \quad \lambda_L = \frac{80 - 37}{4}, \quad \lambda = \min \left\{ 1, 3, \frac{43}{4} \right\} = 1$$

Although step sizes up to $\lambda = \min \left\{ 3, \frac{43}{4} \right\}$ are feasible, the optimal one of KKT computation [17.53] occurs at $\lambda = 1$.

Stopping at a KKT Point with Active Set Methods

Update $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}$ advances us toward an optimum to (a reasonable well behaved) general quadratic program [17.50] as long as move direction $\Delta \mathbf{x} \neq \mathbf{0}$. Should we stop when $\Delta \mathbf{x} = \mathbf{0}$? It depends on whether Lagrange multipliers \mathbf{v} computed from linear system [17.53] for active constraints complete a KKT solution for the full model.

Principle 17.55 If $\Delta \mathbf{x} = \mathbf{0}$ in a solution to conditions [17.53], active set methods will stop at a KKT point for the full model [17.50] if all corresponding Lagrange multipliers \mathbf{v} satisfy sign restrictions

Objective	Active $i \in G$	Active $i \in L$
Minimize	$v_i \geq 0$	$v_i \leq 0$
Maximize	$v_i \leq 0$	$v_i \geq 0$

Lagrange multipliers satisfying conditions [17.55] suffice for a KKT point in the full model [17.50] because the corresponding optimality conditions are

$$\mathbf{c} + 2\mathbf{Q}\mathbf{x}^{(t)} = \sum_i \mathbf{a}^{(i)} v_i = \mathbf{A}_S^T \mathbf{v}^{(S)} \tag{17.58}$$

together with sign restrictions [17.55], complementary slackness on all inequalities, and feasibility in primal constraints. But (17.58) is the first part of the linear system solved in [17.53]; complementary slackness is automatic because only active constraints are allowed to have $v_i \neq 0$; primal feasibility is enforced by the second part of equation system [17.53] and step size rule [17.54]. Thus the only additional requirements for a KKT point are the sign restrictions of principle [17.55].

EXAMPLE 17.27: STOPPING IN ACTIVE SET SEARCH OF QPs

Consider an active set search of a maximizing quadratic program with currently active constraints

$$\begin{aligned} w_1 + 2w_2 &\geq 4 \\ w_3 - 8w_4 + w_5 &\leq 2 \\ 3w_1 + 2w_2 + 2w_3 + 2w_4 + 2w_5 &= 16 \end{aligned}$$

Determine whether the procedure would stop if solution of linear equation system (17.53) produces:

- (a) $\Delta \mathbf{w} = (0, 0, 0, 0, 0)$, $\mathbf{v} = (-33, 10, 14)$
- (b) $\Delta \mathbf{w} = (0, 0, 0, 0, 0)$, $\mathbf{v} = (33, 10, 14)$
- (c) $\Delta \mathbf{w} = (2, -1, -1, 0, 1)$, $\mathbf{v} = (33, 10, 14)$

Solution: We apply principle (17.55).

- (a) For this maximize model, $v_1 = -33$ is appropriate for a \geq constraint, and $v_2 = 10$ is suitable for a \leq . The search would terminate with the current solution a KKT point.
- (b) For this maximize model, $v_1 = 33$ violates sign restrictions of (17.55). The search would not terminate.
- (c) Here the move direction $\Delta \mathbf{w} \neq \mathbf{0}$. The search would continue.

Dropping a Constraint from the Active Set

Clearly, the active set must change if further progress is to be achieved when sign restriction (17.55) are not fulfilled even though system (17.53) produced a move $\Delta \mathbf{x} = \mathbf{0}$. In particular, one or more now active constraint $i \in S$ must be allowed to become a strict inequality.

To see which active constraint to drop from S , focus again on the Lagrange multipliers computed in solving system (17.53). We know these multipliers can be interpreted (principle (17.17)) as the change in (17.55) optimal value with its constraint right-hand sides. For example, $v_i > 0$ for a \leq inequality i of a minimize subproblem indicates that allowing inequality i to become strict (i.e., relaxing to $\mathbf{a}^{(i)} \cdot \Delta \mathbf{x} \leq 0$) will help the objective function. That is, a violation of sign conditions (17.55) indicates an active constraint that could be productively dropped.

Principle 17.56 When solution of system (17.53) produces a $\Delta \mathbf{x} = \mathbf{0}$ but v_i not all satisfying the sign restrictions of (17.55), active set algorithms drop from S some i with a violating v_i .

EXAMPLE 17.28: DROPPING CONSTRAINTS IN ACTIVE SET QP

For each of the cases in Example 17.27 where the procedure did not terminate, determine how the active set S should be modified.

Solution: We apply principle [17.56](#).

- (a) The procedure stops in this case and no modification of S is required.
- (b) For this case we drop constraint $i = 1$ from S and re-solve linear system [17.53](#) because Lagrange multiplier $v_1 = -33$ violates sign restrictions [17.55](#).
- (c) No change is needed in S for this case because the move direction $\Delta \mathbf{w} \neq \mathbf{0}$.

Active Set Solution of the Filter Tuning Application

Algorithm 17D collects principles [17.53](#) – [17.56](#) in an active set procedure for quadratic programs. Figure 17.10 and Table 17.6 detail the application of Algorithm 17D to tuning model (17.53), beginning at $\mathbf{x}^{(0)} = (2.5, 0.0)$.

ALGORITHM 17D: ACTIVE SET METHOD FOR QUADRATIC PROGRAMS

Step 0: Initialization. Pick starting feasible solution $\mathbf{x}^{(0)}$, and initialize working active set S with indices of all constraints active at $\mathbf{x}^{(0)}$. Also, choose stopping tolerance $\epsilon > 0$, and initialize iteration index $t \leftarrow 0$.

Step 1: Subproblem. With $\mathbf{A}_S \triangleq$ coefficient matrix of active constraints in S , solve the Karush–Kuhn–Tucker conditions

$$\begin{pmatrix} -2\mathbf{Q} & \mathbf{A}_S^T \\ \mathbf{A}_S & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}^{t+1} \\ \mathbf{v}_S^{(t+1)} \end{pmatrix} = \begin{pmatrix} \nabla f(\mathbf{x}^{(t)}) \\ \mathbf{0} \end{pmatrix}$$

of direction problem ([17.55](#)) for move direction $\Delta \mathbf{x}^{t+1}$ and active constraint Lagrange multipliers $\mathbf{v}_S^{(t+1)}$. Lagrange multipliers for nonactive $i \notin S$ are fixed $\mathbf{v}_i^{(t+1)} \leftarrow 0$.

Step 2: KKT Point. If $\|\Delta \mathbf{x}^{(t+1)}\| \leq \epsilon$ and $\mathbf{v}^{(t+1)}$ satisfies sign restrictions of [17.55](#), stop; the current $\mathbf{x}^{(t)}$ is a Karush–Kuhn–Tucker point of the given symmetric quadratic program [17.50](#). Otherwise, if $\|\Delta \mathbf{x}^{(t+1)}\| \leq \epsilon$, go to Step 3, and if not, proceed to Step 4.

Step 3: Active Dropping. Choose some $i \in S$ with Lagrange multiplier $v_i^{(t+1)}$ violating sign restrictions [17.55](#), and remove it from S . Then go to Step 6.

Step 4: Step Size. Compute the maximum appropriate step λ in direction $\Delta \mathbf{x}^{(t+1)}$ via

$$\lambda_G \leftarrow \min \left\{ \frac{\mathbf{a}^{(i)} \mathbf{x}^{(t)} - b_i}{-\mathbf{a}^{(i)} \Delta \mathbf{x}^{t+1}} : \mathbf{a}^{(i)} \Delta \mathbf{x}^{t+1} < 0, \quad i \in G \right\}$$

$$\lambda_L \leftarrow \min \left\{ \frac{b_i - \mathbf{a}^{(i)} \mathbf{x}^{(t)}}{\mathbf{a}^{(i)} \Delta \mathbf{x}^{t+1}} : \mathbf{a}^{(i)} \Delta \mathbf{x}^{t+1} > 0, \quad i \in L \right\}$$

$$\lambda \leftarrow \min \{ 1, \lambda_G, \lambda_L \}$$

(continued)

Step 5: Move. Step

$$\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \lambda \Delta \mathbf{x}^{(t+1)}$$

and update S with the indices of any newly active constraints.

Step 6: Advance. Increment $t \leftarrow t + 1$, and return to Step 1.

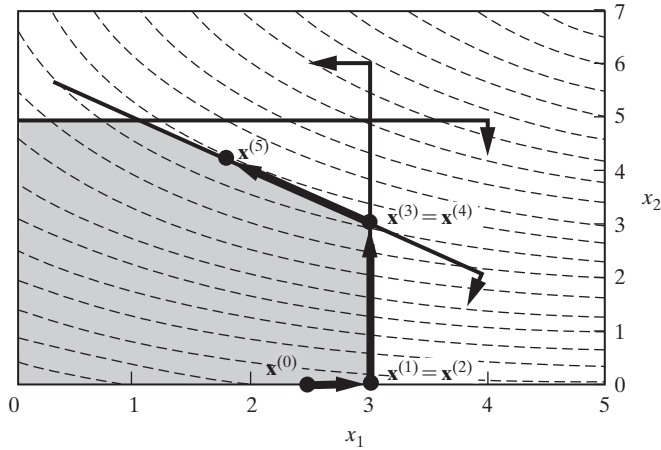


FIGURE 17.10 Active Set QP Solution of Filter Tuning Application

TABLE 17.6 Active Set QP Solution of Filter Tuning Application

		Variables		Constraints						
		x_1	x_2	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$		
$t = 0$	$\mathbf{x}^{(0)}$	2.5	0.0	S	no	yes	no	no	no	
	$\Delta \mathbf{x}$	2.5	0.0	\mathbf{v}	0.0	-30.0	0.0	0.0	0.0	$\lambda = 0.2000$
$t = 1$	$\mathbf{x}^{(1)}$	3.0	0.0	S	no	yes	no	yes	no	
	$\Delta \mathbf{x}$	0.0	0.0	\mathbf{v}	0.0	-26.0	0.0	-4.0	0.0	drop $i = 2$
$t = 2$	$\mathbf{x}^{(2)}$	3.00	0.00	S	no	no	no	yes	no	
	$\Delta \mathbf{x}$	0.0	13.0	\mathbf{v}	0.0	0.0	0.0	-30.0	0.0	$\lambda = 0.2308$
$t = 3$	$\mathbf{x}^{(3)}$	3.00	3.00	S	no	no	yes	yes	no	
	$\Delta \mathbf{x}$	0.0	0.0	\mathbf{v}	0.0	0.0	-20.0	10.0	0.0	drop $i = 4$
$t = 4$	$\mathbf{x}^{(4)}$	3.00	3.00	S	no	no	yes	no	no	
	$\Delta \mathbf{x}$	-1.25	1.25	\mathbf{v}	0.0	0.0	-15.0	0.0	0.0	$\lambda = 1.0000$
$t = 5$	$\mathbf{x}^{(5)}$	1.75	4.25	S	no	no	yes	no	no	
	$\Delta \mathbf{x}$	0.00	0.00	\mathbf{v}	0.0	0.0	-15.0	0.0	0.0	KKT point

Computations (17.56) and (17.57) have already established the first move direction $\Delta \mathbf{x} = (2.5, 0.0)$ and step size $\lambda = 0.2$, which lead to

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \Delta \mathbf{x} = (2.5, 0.0) + 0.2(2.5, 0.0) = (3.0, 0.0)$$

The active set $S = \{2, 4\}$ at $\mathbf{x}^{(1)}$ includes the previous $x_2 \geq 0$ and newly active $x_1 \leq 3$. Solution of the corresponding linear system [17.53] produces null direction $\Delta \mathbf{x} = 0$. We have not yet reached a KKT point for the full model because the sign of $v_2 = -26$ is wrong for a \geq constraint in a minimize problem (principle [17.55]). Thus we drop $i = 2$ from S and re-solve equations [17.53].

The new direction produces a nonzero move, and the search continues. At $t = 5$ the computed direction is again $\Delta \mathbf{x} = 0$. This time, however, Lagrange multipliers satisfy sign restrictions of principle [17.55]. The search terminates with KKT point (here global optimum) $\mathbf{x}^* = (1.75, 4.25)$.

17.8 SEQUENTIAL QUADRATIC PROGRAMMING

Sequential Quadratic Programming (SQP) is a method for solving a broad range of constrained NLPs by creatively combining Lagrangian ideas of Section 17.3 with Newton's Method concepts of Section 16.6, to construct a step-choosing computation that takes the form of Quadratic Programs (QPs) in Section 17.7. If the underlying problem is well behaved, repeated solution of such step-choice QPs, and adoption of the implied steps, converges to an optimum for the underlying NLP. We will show how these ideas combine in a solution strategy after constructing a slightly modified form of Pfizer Lot Sizing Application 17.4 on which to illustrate key ideas.

APPLICATION 17.9: MODIFIED VERSION OF PFIZER APPLICATION 17.4

Recall that Application 17.4 considered the problem of deciding the number of pharmaceutical production lots/campaigns x_j to run per year on products $j = 1, \dots, 4$ while balancing costs of production holding and changeovers against the capacity of resources all share. Our slightly modified model for this section is

$$\begin{aligned} \min \quad & f(\mathbf{x}) \triangleq 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} && \text{(total cost)} \\ & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} \\ (Pf2) \quad \text{s.t.} \quad & h(\mathbf{x}) \triangleq \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} - 221.5 = 0 && \text{(capacity)} \\ & g(\mathbf{x}) \triangleq x_1 - 5 \leq 0 && \text{(limit on product 1)} \end{aligned}$$

Capacity is taken as = because we know it could never be optimal to use less than all of it. Still, a new $x_i \leq 5$ inequality limit has been introduced, to insure that the model has both equalities and inequalities. Decision variables x_1, \dots, x_4 should also be ≥ 0 , but explicit constraints are unneeded because they are certain never to be active.

Lagrangian and Newton Background

We consider constrained NLPs in the generic form

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0 \quad \text{for all } i = 1, \dots, \ell \\ & h_k(\mathbf{x}) = 0 \quad \text{for all } k = 1, \dots, m \end{aligned} \tag{17.59}$$

Functions f , all g_i , and all h_k are assumed to be twice-differentiable, continuous functions of \mathbf{x} .

Rolling constraints into the objective function with multipliers $v_i \leq 0$ on the inequality constraints, and unrestricted w_k on the equalities produces the Lagrangian form

$$\min L(\mathbf{x}, \mathbf{v}, \mathbf{w}) \triangleq f(\mathbf{x}) - \sum_{i=1}^{\ell} v_i g_i(\mathbf{x}) - \sum_{k=1}^m w_k h_k(\mathbf{x}) \quad (17.60)$$

For our Application 17.9 model (Pf2) this becomes

$$\min L(\mathbf{x}, v, w) \triangleq f(\mathbf{x}) - v \cdot g(\mathbf{x}) - w \cdot h(\mathbf{x}) \quad (17.61)$$

where $\mathbf{x} \triangleq (x_1, x_2, x_3, x_4)$, and $v \leq 0$ and w URS are the Lagrange multipliers on the two constraints.

Lagrangian Section 17.3 considered the special case without inequality constraints and showed how to compute a stationary point $(\bar{\mathbf{x}}, \bar{\mathbf{w}})$ of Lagrangian (17.60) with $\nabla L(\bar{\mathbf{x}}, \bar{\mathbf{w}}) = 0$ (principle [17.15]). Under additional assumptions these become an optimal solution to the corresponding NLP.

In contrast to the direct solution approach of Lagrangian Section 17.3, Newton's Method of Section 16.6 (Algorithm 16E) solves (second-order) equation systems in first and second partial derivatives to determine a best next move for the unconstrained optimization underway. Rather than attempting to solve the model in one major round of computation, attractive moves $\Delta \mathbf{x}$ from the second-order equations are applied iteratively as the process repeats.

Sequential Quadratic Programming Strategy

SQP adopts elements of both these strategies.

- Like other Lagrangian approaches, SQP addresses form (17.60), but it does so iteratively through $(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)})$ improving at each iteration t .
- Like Newton's method, SQP addresses a second-order system to determine a best move $\Delta \mathbf{x}$ at each iteration t , but it applies this process to the current Lagrangian $L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)})$ rather than a single unconstrained objective.

Principle 17.57 | At each iteration t , SQP minimizes the second-order approximation to the Lagrangian

$$L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) + \nabla L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x} \nabla^2 L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) \Delta \mathbf{x}$$

as a function of move direction $\Delta \mathbf{x}$.

In terms of the original objective and constraint functions

$$\nabla L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) = \nabla f(\mathbf{x}^{(t)}) - \sum_{i=1}^{\ell} v_i^{(t)} \nabla g_i(\mathbf{x}^{(t)}) - \sum_{k=1}^m w_k^{(t)} \nabla h_k(\mathbf{x}^{(t)}) \quad (17.62)$$

and

$$\nabla^2 L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) = \nabla^2 f(\mathbf{x}^{(t)}) - \sum_{i=1}^{\ell} v_i^{(t)} \nabla^2 g_i(\mathbf{x}^{(t)}) - \sum_{k=1}^m w_k^{(t)} \nabla^2 h_k(\mathbf{x}^{(t)}) \quad (17.63)$$

Furthermore, assuring the new $\Delta \mathbf{x}^{(t)}$ leads to a feasible solution requires constraints

$$\begin{aligned} g_i(\mathbf{x}^{(t)}) + \nabla g_i(\mathbf{x}^{(t)}) \Delta \mathbf{x}^{(t+1)} &\leq 0 \quad i = 1, \dots, \ell \\ h_k(\mathbf{x}^{(t)}) + \nabla h_k(\mathbf{x}^{(t)}) \Delta \mathbf{x}^{(t+1)} &= 0 \quad k = 1, \dots, m \end{aligned} \quad (17.64)$$

ALGORITHM 17E: SEQUENTIAL QUADRATIC PROGRAMMING

Step 0: Initialization. Choose stopping tolerance $\epsilon > 0$, starting feasible solution $\mathbf{x}^{(0)}$, and starting Lagrange multipliers $\mathbf{v}^{(0)} \leq 0$ and $\mathbf{w}^{(0)}$ unrestricted in sign. Then initialize iteration index $t \leftarrow 0$.

Step 1: Quadratic Subproblem. Solve quadratic subproblem $QP(t)$ of principle [17.58] for optimal step $\Delta \mathbf{x}^{(t)}$ and associated optimal constraint multipliers $\mathbf{v}^{(t+1)}$ and $\mathbf{w}^{(t+1)}$. Also update primal solution $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta \mathbf{x}^{(t+1)}$.

Step 2: Stopping. If $\|\Delta \mathbf{x}^{(t)}\| \leq \epsilon$, stop; solutions $\mathbf{x}^{(t+1)}$, $\mathbf{v}^{(t+1)}$, and $\mathbf{w}^{(t+1)}$ together define an approximate KKT optimum for the original NLP. Otherwise, increment $t \leftarrow t + 1$, and return to Step 1.

We can simplify (17.62) to

$$\nabla L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)}) = \nabla f(\mathbf{x}^{(t)}) \quad (17.65)$$

because first-order terms on the constraints become essentially $= 0$ at feasible \mathbf{x} and $\Delta \mathbf{x}$ with complementary Lagrange multipliers \mathbf{v} and \mathbf{w} . Then, minimizing Lagrangian approximation [17.57], dropping the constant part $L(\mathbf{x}^{(t)}, \mathbf{v}^{(t)}, \mathbf{w}^{(t)})$, and expanding as in (17.65) and (17.63) leads at last to the quadratic program that gives SQP its name:

Principle 17.58 | At each iteration t , SQP solves the quadratic program

$$\begin{aligned} \min \quad & \nabla f(\mathbf{x}^{(t)}) \Delta \mathbf{x}^{(t)} + \frac{1}{2} \Delta \mathbf{x}^{(t)} [\nabla^2 f(\mathbf{x}^{(t)}) - \\ (QP_t) \quad & \sum_{i=1}^{\ell} \mathbf{v}_i^{(t)} \nabla^2 g_i(\mathbf{x}^{(t)}) - \sum_{k=1}^m \mathbf{w}_k^{(t)} \nabla^2 h_k(\mathbf{x}^{(t)})] \Delta \mathbf{x}^{(t)} \\ \text{s.t. constraints} \quad & (17.64) \end{aligned}$$

An optimal $\Delta \mathbf{x}^{(t)}$ yields the update $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \Delta \mathbf{x}^{(t+1)}$ and the associated optimal dual multipliers on constraints (17.64) complete the solution as $v_i^{(t+1)}$, $i = 1, \dots, \ell$, and $w_k^{(t+1)}$, $k = 1, \dots, m$.

Algorithm 17E details the full procedure. When computation stops at Step 2, solutions $\mathbf{x}^{(t+1)}$, $\mathbf{v}^{(t+1)}$ and $\mathbf{a}^{(1)} = (1, 1, 0, 0)$, $\mathbf{a}^{(2)} = (1, 0, 0, 0)$ have reached a feasible and complementary minimum of the Lagrangian [17.57], which is very nearly $= 0$. This established their approximate KKT optimality.

Application of Algorithm 17E to Modified Pfizer Application 17.9

To apply Algorithm 17E to our modified Pfizer application, we begin by deriving expressions for the relevant first and second Modified partial derivatives in Table 17.7.

TABLE 17.7 Partial Derivative Expressions for Modified Pfizer Application 17.9

$\nabla f(\mathbf{x}^{(t)}) \triangleq$	$(66.21 - 2160(x_1^{(t)})^{-2}, 426.8 - 8712(x_2^{(t)})^{-2},$ $61.20 - 330(x_3^{(t)})^{-2}, 268.1 - 2916(x_4^{(t)})^{-2})$
$\nabla^2 f(\mathbf{x}^{(t)}) \triangleq$	$(4320(x_1^{(t)})^{-3}, 17424(x_2^{(t)})^{-3}, 660(x_3^{(t)})^{-3}, 5832(x_4^{(t)})^{-3})$ (diagonals only)
$\nabla g(\mathbf{x}^{(t)}) \triangleq$	$(1, 0, 0, 0)$
$\nabla^2 g(\mathbf{x}^{(t)}) \triangleq$	(all zero)
$\nabla h(\mathbf{x}^{(t)}) \triangleq$	$(-180(x_1^{(t)})^{-2}, -726(x_2^{(t)})^{-2}, -27.5(x_3^{(t)})^{-2}, -243(x_4^{(t)})^{-2})$
$\nabla^2 h(\mathbf{x}^{(t)}) \triangleq$	$(360(x_1^{(t)})^{-3}, 1452(x_2^{(t)})^{-3}, 55(x_3^{(t)})^{-3}, 486(x_4^{(t)})^{-3})$ (diagonals only)

Notice that function g , being linear, has constant first partial derivatives and no second partials. Also, the Hessian matrices for functions f and h are diagonal because both are separable (definition [17.6](#)).

Table 17.8 details progress of Algorithm 17E using these expressions. All the building blocks of Table 17.7 are evaluated at each iteration along with the optimal $\Delta \mathbf{x}$ and corresponding v and w .

TABLE 17.8 SQP Solution of Modified Pfizer Application 17.9

$t = 0$	$x = (4.000, 6.000, 3.000, 5.000), v = 0.000, w = -10.000, f = 7034.9, g = -1.000, h = 2.267$ $\nabla f = (-68.790, 184.800, 24.533, 151.460), \nabla^2 f = (67.500, 80.667, 24.444, 46.656)$ $\nabla g = (1.000, 0.000, 0.000, 0.000), \nabla^2 g = \text{all zero}$ $\nabla h = (-11.250, -20.167, -3.056, -9.720), \nabla^2 h = (5.625, 6.722, 2.037, 3.888)$ $\Delta \mathbf{x} = (1.000, -0.0715, 0.0417, -0.789), \ \Delta \mathbf{x}\ = 1.2765, \text{QP obj value} = -208.57$
$t = 1$	$\mathbf{x} = (5.000, 5.929, 3.042, 4.211), v = -42.232, w = -8.639, f = 6878.9, g = 0.000, h = 3.706$ $\nabla f = (-20.190, 178.928, 25.532, 103.656), \nabla^2 f = (34.560, 83.621, 23.453, 78.102)$ $\nabla g = (1.000, 0.000, 0.000, 0.000), \nabla^2 g = \text{all zero}$ $\nabla h = (-7.200, -20.656, -2.972, -13.704), \nabla^2 h = (2.880, 6.698, 1.954, 6.508)$ $\Delta \mathbf{x} = (0.000, 0.0675, 0.0399, 0.1600), \ \Delta \mathbf{x}\ = 0.1782, \text{QP obj value} = 29.343$
$t = 2$	$\mathbf{x} = (5.000, 5.996, 3.082, 4.371), v = -85.943, w = -9.132, f = 6909.8, g = 0.000, h = 0.0983$ $\nabla f = (-20.190, 184.477, 26.449, 115.475), \nabla^2 f = (34.560, 80.828, 22.554, 69.835)$ $\nabla g = (1.000, 0.000, 0.000, 0.000), \nabla^2 g = \text{all zero}$ $\nabla h = (-7.200, -20.194, -2.896, -12.719), \nabla^2 h = (2.880, 6.736, 1.879, 5.820)$ $\Delta \mathbf{x} = (0.000, 0.0008, 0.0006, 0.0064), \ \Delta \mathbf{x}\ = 0.00065, \text{QP obj value} = 0.90214$
$t = 3$	$\mathbf{x} = (5.000, 5.996, 3.082, 4.377), v = -86.002, w = -9.1401, f = 6010.7, g = 0.000, h = -0.0008$ $\nabla f = (-20.190, 184.4542, 26.463, 115.921), \nabla^2 f = (34.560, 80.796, 22.540, 69.529)$ $\nabla g = (1.000, 0.000, 0.000, 0.000), \nabla^2 g = \text{all zero}$ $\nabla h = (-7.200, -20.188, -2.895, -12.682), \nabla^2 h = (2.880, 6.733, 1.878, 5.794)$ $\Delta \mathbf{x} = (0.000, -0.00003, -0.00007, 0.00000), \ \Delta \mathbf{x}\ = 0.00008, \text{QP obj value} = -0.00731$
$t = 4$	$\mathbf{x} = (5.000, 5.997, 3.082, 4.377), v = -86.004, w = -9.141, f = 6010.7, g = 0.000, h = 0.000$

Taking tolerance $\epsilon = 0.0001$, computation begins with $\mathbf{x}^{(0)} = (4, 6, 3, 5)$, $v = 0$, and $w = -10$. The presented $f(x^{(0)}) = 7034.9$ shows the corresponding objective value in the original model, but $h(\mathbf{x}^{(0)}) = 2.267 \neq 0$ demonstrates this start is infeasible in the equality constraint. The optimal $\Delta\mathbf{x}^{(0)}$ yields updated $\mathbf{x}^{(1)} = (5.000, 5.929, 3.042, 4.211)$ with improved objective value 6878.9. Still, its norm $1.2754 > \epsilon$, so computation continues.

Successive iterations continue to improve on the objective function and feasibility. Ultimately, $\Delta\mathbf{x}^{(3)}$ has norm $= 0.00008 < \epsilon$, and we stop with feasible approximate optimum $\mathbf{x} = (5.000, 5.997, 3.082, 4.377)$ with value $= 6010.7$, and corresponding dual multipliers $v = -86.004, w = -9.141$.

Notice that the last (QP) objective value $= -0.00731$ is very close to zero. This verifies that KKT conditions for the full model and its Lagrangian have been approximately satisfied.

Approximations to Reduce Computation

The above development has described the broad outline of the SQP method, but like other second-order procedures of Chapters 16 and 17, the computations required may be very bulky because Hessian matrices are needed at every iteration on the objective and all nonlinear constraints. This has led to approximate methods analogous to Quasi-Newton methods of Section 16.7 for SQP. Details are available in more focused references on NLP.

17.9 SEPARABLE PROGRAMMING METHODS

Separable functions decompose into sums of functions of single decision variables (definition [17.6](#)), and **separable programs** are NLPs over separable objective functions and constraints (definition [17.7](#)). That is, they take the general form

$$\begin{aligned}
 \max \text{ or } \min \quad & f(\mathbf{x}) \triangleq \sum_j f_j(x_j) \\
 \text{s.t.} \quad & g_i(\mathbf{x}) \triangleq \sum_j g_{i,j}(x_j) \geq b_i \quad \text{for all } i \in G \\
 & g_i(\mathbf{x}) \triangleq \sum_j g_{i,j}(x_j) \leq b_i \quad \text{for all } i \in L \quad (17.66) \\
 & g_i(\mathbf{x}) \triangleq \sum_j g_{i,j}(x_j) = b_i \quad \text{for all } i \in E \\
 & x_j \geq 0 \quad \text{for all } j
 \end{aligned}$$

Here G , L , and E index the \geq , \leq , and $=$ constraints, respectively. For notational convenience, we also assume that nonnegativity constraints apply to all variables.

Pfizer Application 17.4 Revisited

Section 17.2 developed Pfizer pharmaceutical manufacturing lot size model

$$\begin{aligned}
 \min \quad & 66.21x_1 + \frac{2160}{x_1} + 426.8x_2 + \frac{8712}{x_2} && \text{(total cost)} \\
 & + 61.20x_3 + \frac{330}{x_3} + 268.1x_4 + \frac{2916}{x_4} && \\
 \text{s.t.} \quad & \frac{180}{x_1} + \frac{726}{x_2} + \frac{27.5}{x_3} + \frac{243}{x_4} \leq 221.5 && \text{(production time)} \\
 & x_1, \dots, x_4 \geq 0 &&
 \end{aligned} \tag{17.67}$$

Variables in this example are

$x_j \triangleq$ number of batches in each run or lot of product j

It is easy to see that this model is separable because its objective function can be written as

$$f(x_1, x_2, x_3, x_4) \triangleq f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4)$$

where

$$f_1(x_1) \triangleq 66.21x_1 + \frac{2160}{x_1}$$

$$f_2(x_2) \triangleq 426.8x_2 + \frac{8712}{x_2}$$

$$f_3(x_3) \triangleq 61.20x_3 + \frac{330}{x_3}$$

$$f_4(x_4) \triangleq 268.1x_4 + \frac{2916}{x_4}$$

and the main production time constraint decomposes similarly with

$$g_1(x_1) \triangleq \frac{180}{x_1}$$

$$g_2(x_2) \triangleq \frac{726}{x_2}$$

$$g_3(x_3) \triangleq \frac{27.5}{x_3}$$

$$g_4(x_4) \triangleq \frac{243}{x_4}$$

The remaining four (nonnegativity) constraints are linear and thus automatically separable.

Principle 17.59 Linear functions are always separable.

By definition, linear functions consist of a sum of terms $a_j x_j$ involving single decision variables.

EXAMPLE 17.29: RECOGNIZING SEPARABLE FUNCTIONS

Determine whether each of the following functions is separable.

(a) $f(w_1, w_2) \triangleq (w_1)^{3.5} + \ln(w_2)$

(b) $g_1(w_1, w_2) \triangleq 14w_1 - 26w_2$

(c) $g_2(w_1, w_2) \triangleq 14w_1 + w_1 w_2 - 26w_2$

Solution: We apply definition [17.6](#).

(a) This f is separable because it decomposes into the sum of $f_1(w_1) \triangleq (w_1)^{3.5}$ and $f_2(w_2) \triangleq \ln(w_2)$.

(b) This g_1 is separable because it is linear (principle [17.59](#)).

(c) This g_2 is not separable because the term $w_1 w_2$ involves both variables.

Piecewise Linear Approximation to Separable Functions

The main special convenience of separable programs is that they can sometimes be approximated closely by LPs (principle [17.8](#)). The transformation begins with piecewise linear approximation of the one-variable functions f_j and $g_{i,j}$.

Definition 17.60 **Piecewise linear approximation** of separable programs divides the domain of each decision variable x_j into a series of intervals k and interpolates linearly to approximate corresponding $f_j(x_j)$ and $g_{i,j}(x_j)$ as

$$f_j(x_j) \approx c_{j,0} + \sum_k c_{j,k} x_{j,k}$$

$$g_{i,j}(x_j) \approx a_{i,j,0} + \sum_k a_{i,j,k} x_{j,k}$$

New variables $x_{j,k}$ represent x_j within interval k , and coefficients $c_{j,k}$ and $a_{i,j,k}$ express interpolation intercepts and slopes.

Figure 17.11 illustrates for x_1 in our Pfizer model (17.67) over domain $[0, 8]$. Both objective function term $f_1(x_1)$ and main constraint function $g_1(x_1)$ have been approximated with three linear segments. One covers $x_1 \in [0, u_{1,1}] = [0, 1]$; a second treats $x_1 \in [u_{1,1}, u_{1,2}] = [1, 5.7]$; and the third handles $x_1 \in [u_{1,2}, u_{1,3}] = [5.7, 8]$. First breakpoint $u_{1,1} = 1$ was chosen near the “knee” of the two functions, $u_{1,2} = 5.7$ approximates the minimum of $f_1(x_1)$, and $u_{1,3} = 8$ is a practical upper bound on x_1 .

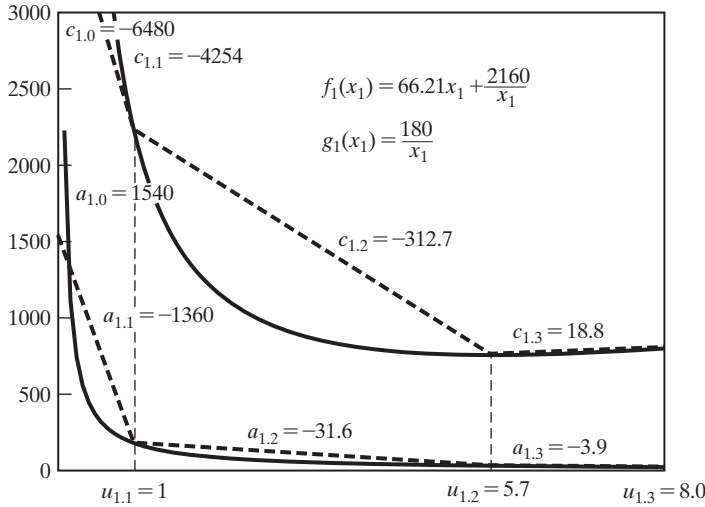


FIGURE 17.11 Piecewise Linear Approximation of Pfizer Application 17.4

Introducing new variables $x_{1,1}, x_{1,2}$, and $x_{1,3}$ with coefficients as in Figure 17.11, we have

$$f_1(x_1) \triangleq 66.21x_1 + \frac{2160}{x_1} \approx 6480 - 4254x_{1,1} - 312.7x_{1,2} + 18.8x_{1,3}$$

$$g_1(x_1) \triangleq \frac{180}{x_1} \approx 1540 - 1360x_{1,1} - 31.6x_{1,2} - 3.9x_{1,3}$$

Notice that the same interval limits must be used to approximate x_1 in the objective and all constraints. Upper bounds on the new variables are derived from interval limits

$$\begin{aligned} 0 &\leq x_{1,1} \leq u_{1,1} = 1 \\ 0 &\leq x_{1,2} \leq u_{1,2} - u_{1,1} = 5.7 - 1.0 = 4.7 \\ 0 &\leq x_{1,3} \leq u_{1,3} - u_{1,2} = 8.0 - 5.7 = 2.3 \end{aligned}$$

EXAMPLE 17.30: FORMING PIECEWISE LINEAR APPROXIMATIONS

Consider a separable nonlinear program with objective function and constraint component functions for nonnegative decision variable w_1 given by

$$\begin{aligned} f_1(w_1) &\triangleq (w_1)^2 - 4w_1 + 22 \\ g_{1,1}(w_1) &\triangleq \sqrt{w_1 + 9} \\ g_{2,1}(w_1) &\triangleq 14w_1 \end{aligned}$$

Form corresponding piecewise linear approximations using breakpoints $u_{1,1} = 2$ and $u_{1,2} = 5$.

Solution: We must estimate the interpolation coefficients of definition [17.60](#). Intercepts are

$$c_{1,0} = f_1(0) = 22, \quad a_{1,1,0} = g_{1,1}(0) = 3, \quad a_{2,1,0} = g_{2,1}(0) = 0$$

Slopes for the interval $[0, u_{1,1}] = [0, 2]$ are derived as

$$c_{1,1} = \frac{f_1(2) - f_1(0)}{2 - 0} = -2$$

$$a_{1,1,1} = \frac{g_{1,1}(2) - g_{1,1}(0)}{2 - 0} = 0.158$$

$$a_{2,1,1} = \frac{g_{2,1}(2) - g_{2,1}(0)}{2 - 0} = 14$$

Corresponding slopes for interval $[u_{1,1}, u_{2,1}] = [2, 5]$ are

$$c_{1,2} = \frac{f_1(5) - f_1(2)}{5 - 2} = 3$$

$$a_{1,1,2} = \frac{g_{1,1}(5) - g_{1,1}(2)}{5 - 2} = 0.142$$

$$a_{2,1,2} = \frac{g_{2,1}(5) - g_{2,1}(2)}{5 - 2} = 14$$

Thus piecewise linear approximations are

$$f_1(w_1) \approx 22 - 2w_{1,1} + 3w_{1,2}$$

$$g_{1,1}(w_1) \approx 3 + 0.158w_{1,1} + 0.142w_{1,2}$$

$$g_{2,1}(w_1) \approx 0 + 14w_{1,1} + 14w_{1,2}$$

Linear Program Representation of Separable Programs

Applying piecewise linear approximation [\[17.60\]](#) to every variable of separable program format (17.66) yields an LP approximation.

Definition 17.61 The **linear programming approximation** to a separable nonlinear program over nonnegative variables can be expressed as

$$\begin{aligned} \max \text{ or } \min \quad & \sum_j \left(c_{j,0} + \sum_k c_{j,k} x_{j,k} \right) \\ \text{s.t.} \quad & \sum_j \left(a_{i,j,0} + \sum_k a_{i,j,k} x_{j,k} \right) \geq b_i \quad \text{for all } i \in G \\ & \sum_j \left(a_{i,j,0} + \sum_k a_{i,j,k} x_{j,k} \right) \leq b_i \quad \text{for all } i \in L \\ & \sum_j \left(a_{i,j,0} + \sum_k a_{i,j,k} x_{j,k} \right) = b_i \quad \text{for all } i \in E \\ & 0 \leq x_{j,k} \leq u_{j,k} - u_{j,k-1} \quad \text{for all } j, k \end{aligned}$$

where $u_{j,k}$ are the interval breakpoints for variable x_j ($u_{j,0} \triangleq 0$), and the coefficients $c_{j,k}$ and $a_{i,j,k}$ express interpolation intercepts and slopes.

This model is a linear program solvable by methods of Chapters 5–7.

Full application of construction [\[17.61\]](#) to our Pfizer lot sizing model produces the linear approximation

$$\begin{aligned}
 \min \quad & 42,354 - 4254x_{1,1} - 312.7x_{1,2} + 18.8x_{1,3} \\
 & - 16,997x_{2,1} - 1509x_{2,2} + 184.8x_{2,3} \\
 & - 598.8x_{3,1} - 82.3x_{3,2} + 43.3x_{3,3} \\
 & - 5564x_{4,1} - 615.5x_{4,2} + 157.6x_{4,3} \\
 \text{s.t.} \quad & 7030 - 1360x_{1,1} - 31.6x_{1,2} - 3.9x_{1,3} \\
 & - 2452x_{2,1} - 161.3x_{2,2} - 20.2x_{2,3} \\
 & - 555x_{3,1} - 12x_{3,2} - 1.5x_{3,3} \\
 & - 1486x_{4,1} - 73.6x_{4,2} - 9.2x_{4,3} \leq 221.5 \\
 & 0 \leq x_{1,1} \leq 1, 0 \leq x_{1,2} \leq 4.7, 0 \leq x_{1,3} \leq 2.3 \\
 & 0 \leq x_{2,1} \leq 1, 0 \leq x_{2,2} \leq 3.5, 0 \leq x_{2,3} \leq 3.5 \\
 & 0 \leq x_{3,1} \leq 1, 0 \leq x_{3,2} \leq 1.3, 0 \leq x_{3,3} \leq 5.7 \\
 & 0 \leq x_{4,1} \leq 1, 0 \leq x_{4,2} \leq 2.3, 0 \leq x_{4,3} \leq 4.7
 \end{aligned} \tag{17.68}$$

An optimal solution is

$$\begin{aligned}
 x_{1,1}^* &= 1.0, \quad x_{1,2}^* = 4.7, \quad x_{1,3}^* = 2.3, \quad \text{or} \quad x_{1,1}^* + x_{1,2}^* + x_{1,3}^* = x_1^* = 8.0 \\
 x_{2,1}^* &= 1.0, \quad x_{2,2}^* = 3.5, \quad x_{2,3}^* = 2.4, \quad \text{or} \quad x_{2,1}^* + x_{2,2}^* + x_{2,3}^* = x_2^* = 6.9 \\
 x_{3,1}^* &= 1.0, \quad x_{3,2}^* = 1.3, \quad x_{3,3}^* = 0.0, \quad \text{or} \quad x_{3,1}^* + x_{3,2}^* + x_{3,3}^* = x_3^* = 2.3 \\
 x_{4,1}^* &= 1.0, \quad x_{4,2}^* = 2.3, \quad x_{4,3}^* = 0.0, \quad \text{or} \quad x_{4,1}^* + x_{4,2}^* + x_{4,3}^* = x_4^* = 3.3
 \end{aligned} \tag{17.69}$$

which corresponds fairly well to the nonlinear optimum of (17.7).

Correctness of the LP Approximation to Separable Programs

Does linear program [\[17.61\]](#) correctly model (17.66) (except for interpolation error)? Sometimes yes, sometimes no. To see the potential difficulty, suppose that we attempt to

$$\begin{aligned}
 \max \quad & f(y) \triangleq y^2 - 12y + 45 \\
 \text{s.t.} \quad & 0 \leq y \leq 9
 \end{aligned}$$

This f is the function in Figure 17.4 (Section 17.2). Obviously, the optimal $y^* = 0$, because $f(0) = 45, f(9) = 18$, and every y in between has a lower function value.

Forming linear program representation [\[17.61\]](#) for this simple example yields

$$\begin{aligned}
 \max \quad & 45 - 7y_1 + 2y_2 \\
 \text{s.t.} \quad & 0 \leq y_1 \leq 5 \\
 & 0 \leq y_2 \leq 4
 \end{aligned}$$

An optimal solution makes $y_1^* = 0, y_2^* = 4$, which implies that $y^* = y_1^* + y_2^* = 4$.

What went wrong? The approximation

$$f(y) \approx c_0 + \sum_k c_k y_k$$

corresponding to [17.60] is correct only if we assume that segment variables y_k satisfy a certain sequence at optimality. For $y \in [0, 5]$, we want first segment y_1 to represent y . If $y \in [5, 9]$, we want segment 1 to run to its upper limit and segment 2 to do the rest.

In general, each segment of a piecewise linear approximation must reach its upper bound before the next is available for use.

Principle 17.62 Linear program representation [17.61] gives a correct approximation to separable program (17.66) whenever optimal values for segment variables satisfy

$$x_{j,k+1}^* > 0 \quad \text{only if} \quad x_{j,k}^* = (u_{j,k} - u_{j,k-1}) \quad \text{for all } j \text{ and } k$$

with $u_{j,0} \triangleq 0$.

Pfizer application results (17.69) illustrate a case where conditions [17.62] are satisfied. For example, at $j = 2$, segment upper bounds in formulation (17.68) are 1.0, 3.5, and 3.5, while

$$x_{2,1}^* = 1.0, \quad x_{2,2}^* = 3.5, \quad x_{2,3}^* = 2.4$$

Each positive segment has the preceding one at its upper bound.

EXAMPLE 17.31: CHECKING PIECEWISE LINEAR APPROXIMATIONS

A piecewise linear approximation to a separable nonlinear program in nonnegative variables w_1 and w_2 uses breakpoints $u_{1,1} = 2$ and $u_{1,2} = 6$ for the first variable, together with $u_{2,1} = 7$ and $u_{2,2} = 20$ for the second. Determine whether each of the following LP approximation solutions provides a correct answer (except for interpolation error) to the original nonlinear program.

- (a) $w_{1,1}^* = 2, w_{1,2}^* = 3, w_{2,1}^* = 6, w_{2,2}^* = 0$
- (b) $w_{1,1}^* = 0, w_{1,2}^* = 3, w_{2,1}^* = 1, w_{2,2}^* = 13$

Solution: Following formulation [17.61], bounds on the segment variables will be

$$0 \leq w_{1,1} \leq 2 - 0 = 2, \quad 0 \leq w_{1,2} = 6 - 2 = 4$$

$$0 \leq w_{2,1} \leq 7 - 0 = 7, \quad 0 \leq w_{2,2} = 20 - 7 = 13$$

- (a) This optimal solution does give a correct approximation because it satisfies sequencing conditions [17.62]. Variable $w_{1,1}^*$ equals its upper bound, so $w_{1,2}^*$ can be positive.
- (b) This solution yields an incorrect approximation because it violates sequencing conditions [17.62]. Variable $w_{1,2}^* > 0$ and $w_{1,1}^*$ does not equal its upper bound.

Convex Separable Programs

Suppose now that the given separable program (17.61) is also a convex program. That is, we are minimizing a convex objective or maximizing a concave one, subject to concave \geq constraints, convex \leq ones, and linear equalities (definition [17.3]).

These requirements relate to the entire objective of constraint functions. Still, it is easy to see that all component functions for each variable in separable form (17.61) must have similar properties.

Principle 17.63 | Separable function

$$s(x_1, \dots, x_n) \triangleq \sum_{j=1}^n s_j(x_j)$$

is convex if and only if each component s_j is convex. It is concave if and only if each s_j is concave.

We already know from principle [16.29](#) that sums of convex (or concave) functions are convex (concave). But [17.63](#) asserts that the converse is also true for separable functions. To see why, we need only choose points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ with all components equal except the j th. Then a step λ from $\mathbf{x}^{(1)}$ toward $\mathbf{x}^{(2)}$ changes just component $s_j(x_j^{(1)} + \lambda(x_j^{(2)} - x_j^{(1)}))$ of s . Convex function definition [16.23](#) can hold for s only if it holds for s_j .

Given their many convenient properties, it should not surprise us that separable convex programs satisfy requirement [17.62](#) for good approximation by linear programming.

Principle 17.64 | Linear approximations [17.61](#) to a separable convex programs have an optimal solution satisfying sequencing conditions [17.62](#) if they have any optimum at all.

Again, Pfizer application model (17.68) illustrates. We have seen in Section 16.2 that the nonlinear version (17.67) is a convex program. Thus it was no accident that segment optima (17.69) satisfy sequencing condition [17.67](#).

To understand why separable convex programs have the required property for effective LP approximation, recall (principle [16.27](#)) that second derivatives of convex functions are nonnegative and those of concave functions are nonpositive. It follows that first derivatives, or slopes, are nondecreasing and nonincreasing, respectively.

For a minimizing convex objective, this implies that the least cost slope $c_{j,k}$ of each approximation occurs at $k = 1$, and for a maximizing concave objective, the first segment will also be the most preferred. In a similar way, coefficients in constraint rows also exhibit a preference for lower-numbered segments. For example, if constraint i is \leq , and thus convex, $a_{i,j,1}$ is the smallest slope in the approximation of $g_{i,j}$. Thus it does least damage to feasibility. If the constraint is \geq , and thus concave, $a_{i,j,1}$ is largest and advances feasibility most rapidly.

Combining these observations about objective and constraint approximations, we see that the first segment $x_{j,k}$ of each piecewise linear approximation gives the greatest objective function payoff with the least burden on constraints. An optimal LP solution must choose it first, making the second segment positive only when the first has reached its upper bound. The pattern continues through each interval k , so that property [17.62](#) is satisfied.

Difficulties with Nonconvex Separable Programs

When the given separable program is not convex, property [17.62] may not hold automatically. However, it can be enforced artificially.

Suppose that we apply upper-and lower-bounded simplex Algorithm 5D to linear approximation [17.61]. Each iteration chooses either a nonbasic lower-bounded variable to increase or a nonbasic upper-bounded variable to decrease.

General separable programming searches simply restrict the choices even further to maintain sequencing property [17.62]. A nonbasic lower-bounded segment variable cannot increase unless the preceding segment is already at its upper bound. Similarly, a nonbasic upper-bounded segment variable cannot decrease unless the succeeding segment has value 0.

Of course, these extra limitations may prevent the simplex from computing an optimal solution to linear program [17.61]. However, they do assure that property [17.67] is enforced. When no allowable pivots remain, we stop with a heuristic optimum.

An alternative approach producing globally optimal solutions can be derived using integer linear programming methods of Section 11.1. Binary variables $y_{j,k}$ are introduced that parallel each $x_{j,k}$, with

$$y_{j,k} \triangleq \begin{cases} 1 & \text{if } x_{j,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Then the switching constraints

$$(u_{j,k} - u_{j,k-1})y_{j,k+1} \leq x_{j,k} \leq (u_{j,k} - u_{j,k-1})y_{j,k} \quad \text{for all } j, k$$

enforce sequencing conditions [17.67] by pushing each segment to its upper bound if the next is positive.

17.10 POSYNOMIAL GEOMETRIC PROGRAMMING METHODS

As we have seen in Sections 17.1 and 17.2, many important applications of nonlinear programming arise in engineering design, where decision variables are physical dimensions, pressures, and so on. Such models are often highly nonlinear and have many locally optimal solutions. This section deals with special cases called **posynomial geometric programs**, which address the “variables to powers” form of many engineering design models and constitute the only broad class of nonconvex NLPs readily solved to global optimality.

Posynomial Geometric Program Form

In Section 17.2 we introduced (definition [17.10]) **posynomial** functions, which are positive-weighted sums of products of decision variables raised to arbitrary powers. A **posynomial geometric program (GP)** is an NLP minimizing a posynomial objective function over positive variables and \leq posynomial main constraints (definition [17.11]). The general form is

$$\min \sum_{k \in \mathcal{K}_0} d_k \prod_{j=1}^n (x_j)^{a_{kj}}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{k \in K_i} d_k \prod_{j=1}^n (x_j)^{a_{k,j}} \leq 1 & i = 1, \dots, m \\ & x_j > 0 & j = 1, \dots, n \end{aligned} \tag{17.70}$$

where nonoverlapping sets K_i index the posynomial terms in the objective and constraints, values d_k are the corresponding weights, and the $a_{k,j}$ are exponents of variables x_j in terms k .

Several details of the format will prove critical to its tractability:

- The objective must minimize a posynomial. Maximizations are not allowed.
- Coefficients d_k must all be positive.
- Constraints must enforce \leq requirements on a posynomial. Any $=$ and \geq forms destroy the structure.
- Decision variables must be limited to positive values. We will want to take their logarithms.

Cofferdam Application Revisited

We illustrate GP methods with Section 17.2’s model (17.11) to optimize a cofferdam design:

$$\begin{aligned} \min \quad & 168x_1x_2 + 3648x_1 + 3648\frac{x_1x_2}{x_3} + \frac{1.46 \times 10^7}{x_4} & (\text{cost}) \\ \text{s.t.} \quad & \frac{1.25x_4}{x_1} + \frac{41.625}{x_1} \leq 1 & (\text{empirical}) \\ & \frac{1.0425x_1}{x_2} \leq 1 & (\text{slipping}) \\ & 0.00035x_1x_3 \leq 1 & (\text{tension}) \\ & x_1, x_2, x_3, x_4 > 0 \end{aligned} \tag{17.71}$$

In the notation of (17.70), $K_0 = \{1, 2, 3, 4\}$, $K_1 = \{5, 6\}$, $K_2 = \{7\}$, and $K_3 = \{8\}$. Corresponding coefficients are

$$\begin{aligned} d_1 &= 168, & d_2 &= 3648, & d_3 &= 3648, & d_4 &= 1.46 \times 10^7 \\ d_5 &= 1.25, & d_6 &= 41.625, & d_7 &= 1.0425, & d_8 &= .00035 \\ a_{1,1} &= 1, & a_{1,2} &= 1, & a_{1,3} &= 0, & a_{1,4} &= 0 \\ a_{2,1} &= 1, & a_{2,2} &= 0, & a_{2,3} &= 0, & a_{2,4} &= 0 \\ a_{3,1} &= 1, & a_{3,2} &= 1, & a_{3,3} &= -1, & a_{3,4} &= 0 \\ a_{4,1} &= 0, & a_{4,2} &= 0, & a_{4,3} &= 0, & a_{4,4} &= -1 \\ a_{5,1} &= -1, & a_{5,2} &= 0, & a_{5,3} &= 0, & a_{5,4} &= 1 \\ a_{6,1} &= -1, & a_{6,2} &= 0, & a_{6,3} &= 0, & a_{6,4} &= 0 \\ a_{7,1} &= 1, & a_{7,2} &= -1, & a_{7,3} &= 0, & a_{7,4} &= 0 \\ a_{8,1} &= 1, & a_{8,2} &= 0, & a_{8,3} &= 1, & a_{8,4} &= 0 \end{aligned}$$

EXAMPLE 17.32: PLACING GEOMETRIC PROGRAMS IN STANDARD FORM

Identify the constants and index sets of standard form (17.70) for the following posynomial geometric program:

$$\begin{aligned} \min \quad & 3 \frac{w_1^{43}}{w_2} + 14w_2w_3 \\ \text{s.t.} \quad & w_1\sqrt{w_3} + w_2\sqrt{w_3} \leq 20 \\ & \frac{w_1}{w_2} \leq 1 \\ & w_1, w_2, w_3 > 0 \end{aligned}$$

Solution: Begin by dividing through the main constraint by 20 to obtain standard right-hand side 1. Then the objective function has terms $k \in K_0 \triangleq \{1, 2\}$, the first constraint involves $k \in K_1 \triangleq \{3, 4\}$, and the second has only the one $k \in K_2 \triangleq \{5\}$. The corresponding standard-form coefficients are

$$d_1 = 3, \quad d_2 = 14, \quad d_3 = 0.05, \quad d_4 = 0.05, \quad d_5 = 1$$

and

$$\begin{aligned} a_{1,1} &= 0.43, & a_{1,2} &= -1, & a_{1,3} &= 0 \\ a_{2,1} &= 0, & a_{2,2} &= 1, & a_{2,3} &= 1 \\ a_{3,1} &= 1, & a_{3,2} &= 0, & a_{3,3} &= 0.5 \\ a_{4,1} &= 0, & a_{4,2} &= 1, & a_{4,3} &= 0.5 \\ a_{5,1} &= 1, & a_{5,2} &= -1, & a_{5,3} &= 0 \end{aligned}$$

Logarithmic Change of Variables in GPs

Posynomial functions need not be convex (see application (17.14)), and thus geometric programs (17.70) are often not convex programs. However, a change of variables can make them convex. In particular (principle [17.12](#)), we consider substituting

$$z_j \triangleq \ln(x_j) \tag{17.72}$$

or

$$x_j \triangleq e^{z_j} \tag{17.73}$$

Under transformation (17.73), terms k of posynomials in (17.70) simplify as

$$d_k \prod_j (x_j)^{a_{k,j}} = d_k \prod_j (e^{z_j})^{a_{k,j}} = d_k \prod_j e^{a_{k,j}z_j} = d_k e^{\sum_j a_{k,j}z_j} = d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}}$$

where $\mathbf{a}^{(k)} \triangleq (a_{k,1}, \dots, a_{k,n})$ and $\mathbf{z} \triangleq (z_1, \dots, z_n)$. Then original geometric program form (17.70) becomes

$$\begin{aligned} \min \quad & f(\mathbf{z}) \triangleq \sum_{k \in K_0} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} \\ \text{s.t.} \quad & g_i(\mathbf{z}) \triangleq \sum_{k \in K_i} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} \leq 1 \quad i = 1, \dots, m \\ & z_j \text{ URS} \quad j = 1, \dots, n \end{aligned} \tag{17.74}$$

For example, cofferdam model (17.71) transforms to

$$\begin{aligned} \min \quad & 168e^{\mathbf{a}^{(1)} \cdot \mathbf{z}} + 3648e^{\mathbf{a}^{(2)} \cdot \mathbf{z}} + 3648e^{\mathbf{a}^{(3)} \cdot \mathbf{z}} + (1.46 \times 10^7)e^{\mathbf{a}^{(4)} \cdot \mathbf{z}} \\ \text{s.t.} \quad & 1.25e^{\mathbf{a}^{(5)} \cdot \mathbf{z}} + 41.625e^{\mathbf{a}^{(6)} \cdot \mathbf{z}} \leq 1 \\ & 1.0425e^{\mathbf{a}^{(7)} \cdot \mathbf{z}} \leq 1 \\ & 0.00035e^{\mathbf{a}^{(8)} \cdot \mathbf{z}} \leq 1 \\ & \mathbf{z} \text{ URS} \end{aligned} \tag{17.75}$$

with

$$\begin{aligned} \mathbf{a}^{(1)} &= (1, 1, 0, 0), & \mathbf{a}^{(2)} &= (1, 0, 0, 0) \\ \mathbf{a}^{(3)} &= (1, 1, -1, 0), & \mathbf{a}^{(4)} &= (0, 0, 0, -1) \\ \mathbf{a}^{(5)} &= (-1, 0, 0, 1), & \mathbf{a}^{(6)} &= (-1, 0, 0, 0) \\ \mathbf{a}^{(7)} &= (1, -1, 0, 0), & \mathbf{a}^{(8)} &= (1, 0, 1, 0) \end{aligned}$$

EXAMPLE 17.33: CHANGING VARIABLES IN GEOMETRIC PROGRAMS

Return to the posynomial geometric program of Example 17.32. Change variables via (17.73) to produce a convex program in format (17.74).

Solution: Using the coefficients of Example 17.32, the transformed model is

$$\begin{aligned} \min \quad & 3e^{0.43z_1 - 1z_2} + 14e^{1z_2 + 1z_3} \\ \text{s.t.} \quad & 0.05e^{1z_1 + 0.5z_3} + 0.05e^{1z_2 + 0.5z_3} \leq 1 \\ & 1e^{1z_1 - 1z_2} \leq 1 \\ & z_1, z_2, z_3 \text{ URS} \end{aligned}$$

Convex Transformed GP Model

The power of this simple change of variables (17.73) is to convert a posynomial geometric program to a convex program (definition [17.3](#)).

Principle 17.65 | The transformed model obtained from a geometric program by substituting $x_j = e^{z_j}$ is a convex program in new variables z_j .

To see why the transformed model is convex in \mathbf{z} , observe that the objective and constraint functions are positive-weighted sums of terms

$$p_k(\mathbf{z}) \triangleq e^{\mathbf{a}^{(k)} \cdot \mathbf{z}}$$

Linear exponent $\mathbf{a}^{(k)} \cdot \mathbf{z}$ is convex in \mathbf{z} (principle [16.28](#)), and $h(y) \triangleq e^y$ is nondecreasing and convex. Composition principle [16.31](#) then implies that each p_k is convex, so that their transformed sum must be, too. It follows that format (17.74) minimizes a convex objective, subject to convex \leq constraints, which makes it a convex program.

Notice how this analysis depends on details of the posynomial GP format. Coefficients $d_{j,k}$ must be positive for weighted sums of convex terms to be guaranteed convex. Also, a minimizing objective and \leq nonlinear constraints are essential if a model over convex functions is to be a convex program.

Direct Solution of the Transformed Primal GP

Principle [17.65](#) provides a direct avenue to computing global optimal solutions to posynomial geometric programs (17.65). We need only substitute $x_j = e^{z_j}$, solve the resulting convex program in \mathbf{z} by methods of Section 17.6, and transform back by

$$x_j^* \leftarrow e^{z_j^*} \quad \text{for all } j \quad (17.76)$$

For example, application of reduced gradient Algorithm 17C to transformed model (17.75) produces optimal solution

$$z_1^* = 4.138, \quad z_2^* = 4.179, \quad z_3^* = 3.820, \quad z_4^* = 2.823$$

Then inverse transformation (17.76) gives the following optimal solution in original variables:

$$\begin{aligned} x_1^* &= e^{4.138} = 62.65, & x_2^* &= e^{4.179} = 65.32, \\ x_3^* &= e^{3.820} = 45.60, & x_4^* &= e^{2.823} = 16.82 \end{aligned}$$

Dual of a Geometric Program

Sometimes even more efficient methods than solving convex program (17.74) can be used to optimize geometric programs. The process begins with still another transformation of the given model termed its dual.

In addition to the usual Lagrange multiplier for each constraint, geometric programming duals introduce variables for every term k of the objective and constraint posynomials:

$$\begin{aligned} v_i &\triangleq \text{Lagrange multiplier for constraint } i \\ \delta_k &\triangleq \text{dual variable for posynomial term } k \end{aligned}$$

Definition 17.66 The **GP dual** corresponding to posynomial geometric program (17.70) can be expressed as

$$\begin{aligned} \max \quad & \sum_{\text{all } k} \delta_k \ln \left(\frac{d_k}{\delta_k} \right) - \sum_{j=1}^m v_j \ln(-v_j) \\ \text{s.t.} \quad & \sum_{\text{all } k} \mathbf{a}^{(k)} \delta_k = 0 \\ & \sum_{k \in K_0} \delta_k = 1 \end{aligned}$$

(Continued)

$$\begin{aligned} \sum_{k \in K_i} \delta_k &= -\nu_i & i = 1, \dots, m \\ \delta_k &\geq 0 & \text{for all } k \\ \nu_i &\leq 0 & i = 1, \dots, m \end{aligned}$$

where ν_i is the Lagrange multiplier on main primal constraint i and δ_k is the dual variable for polynomial term k .

Cofferdam model (17.71) illustrates. Dual form 17.66 is

$$\begin{aligned} \max \quad & \delta_1 \ln\left(\frac{168}{\delta_1}\right) + \delta_2 \ln\left(\frac{3648}{\delta_2}\right) + \delta_3 \ln\left(\frac{3648}{\delta_3}\right) \\ & + \delta_4 \ln\left(\frac{1.46 \times 10^7}{\delta_4}\right) + \delta_5 \ln\left(\frac{1.25}{\delta_5}\right) + \delta_6 \ln\left(\frac{41.625}{\delta_6}\right) \\ & + \delta_7 \ln\left(\frac{1.0425}{\delta_7}\right) + \delta_8 \ln\left(\frac{0.00035}{\delta_8}\right) \\ \text{s.t.} \quad & -\nu_1 \ln(-\nu_1) - \nu_2 \ln(-\nu_2) - \nu_3 \ln(-\nu_3) \\ & +\delta_1 + \delta_2 + \delta_3 - \delta_5 - \delta_6 + \delta_7 + \delta_8 = 0 \\ & +\delta_1 + \delta_3 - \delta_7 = 0 \\ & -\delta_3 + \delta_8 = 0 \\ & -\delta_4 + \delta_5 = 0 \\ & +\delta_1 + \delta_2 + \delta_3 + \delta_4 = 1 \\ & +\delta_5 + \delta_6 = -\nu_1 \\ & +\delta_7 = -\nu_2 \\ & +\delta_8 = -\nu_3 \\ & \delta_1, \dots, \delta_8 \geq 0 \\ & \nu_1, \nu_2, \nu_3 \leq 0 \end{aligned} \tag{17.77}$$

It first 4 constraints weight δ_k with exponents $a_{j,k}$ for primal variables j . The fifth normalizes the δ total of variables relating to the objective function. The remaining main constraints simply recover (negatives of) Lagrange multipliers for the 3 primal constraints as sums of associated δ_k . An optimal solution is

$$\begin{aligned} \nu_1^* &= -1.225, & \nu_2^* &= -0.481, & \nu_3^* &= -0.155, \\ \delta_1^* &= 0.326, & \delta_2^* &= 0.108, & \delta_3^* &= 0.155, & \delta_4^* &= 0.411 \\ \delta_5^* &= 0.411, & \delta_6^* &= 0.814, & \delta_7^* &= 0.481, & \delta_8^* &= 0.155 \end{aligned} \tag{17.78}$$

with optimal value 14.563.

EXAMPLE 17.34: FORMULATING GEOMETRIC PROGRAM DUALS

Form the dual of the posynomial geometric program in Example 17.33.

Solution: Following format [17.66](#), we introduce Lagrange multipliers ν_1 and ν_2 for the main constraints, and variables $\delta_1, \dots, \delta_5$ for the five posynomial terms. Then the dual becomes

$$\begin{aligned} \max \quad & \delta_1 \ln\left(\frac{3}{\delta_1}\right) + \delta_2 \ln\left(\frac{14}{\delta_2}\right) + \delta_3 \ln\left(\frac{0.05}{\delta_3}\right) + \delta_4 \ln\left(\frac{0.05}{\delta_4}\right) + \delta_5 \ln\left(\frac{1}{\delta_5}\right) \\ & - \nu_1 \ln(-\nu_1) - \nu_2 \ln(-\nu_2) \\ \text{s.t.} \quad & 0.43\delta_1 + 1\delta_3 + 1\delta_5 = 0 \\ & -1\delta_1 + 1\delta_2 + 1\delta_4 - 1\delta_5 = 0 \\ & 1\delta_2 + .5\delta_3 + .5\delta_4 = 0 \\ & 1\delta_1 + 1\delta_2 = 1 \\ & 1\delta_3 + 1\delta_4 = -\nu_1 \\ & 1\delta_5 = -\nu_2 \\ & \delta_1, \dots, \delta_5 \geq 0 \\ & \nu_1, \nu_2 \leq 0 \end{aligned}$$

Degrees of Difficulty and Solving the GP Dual

Dual problem [17.66](#) is a separable program (definition [17.7](#)) with linear constraints. Furthermore, its objective function can be shown to be concave over feasible (δ, \mathbf{v}) , even though it is not concave over all choices of the decision variables.

Principle 17.67 | Posynomial geometric program dual [17.66](#) is a separable convex program over linear constraints.

Thus either the separable programming methods of Section 17.9 or the reduced gradient algorithms of Section 17.6 can be employed to compute a global optimum.

Sometimes the task is even easier. Noting that the last main system of constraints entirely determines the ν_i in terms of the δ_k , the degree of difficulty of a model depends on the number of truly independent variables δ_k in other main constraints.

Definition 17.68 | The **degree of difficulty** of a geometric program is

$$(\text{number of posynomial terms } k) - (\text{number of variables } j) - 1$$

The first two sets of main constraints in dual [17.66](#) have one variable for each k and $(n + 1)$ constraints for the n primal decision variables. Thus the degree of difficulty bounds the number of δ -variables that must be fixed in value to determine the rest uniquely. Some models even have degree of difficulty zero, meaning that the

dual can be optimized by solving a system of linear equations. Cofferdam application dual (17.77) has degree of difficulty

$$\text{terms} - \text{variables} - 1 = 8 - 4 - 1 = 3$$

EXAMPLE 17.35: DETERMINING GP DEGREES OF DIFFICULTY

Determine the degrees of difficulty in posynomial geometric program of Examples 17.32 to 17.34.

Solution: The model has 5 posynomial terms and 3 variables. Thus its degree of difficulty is $5 - 3 - 1 = 1$.

Recovering a Primal GP Solution

We have seen that the dual problem [17.66] may be convenient to solve, but primal [17.11] is the model of true interest. How can we retrieve a primal optimum \mathbf{x}^* ?

An elegant dual relationship makes recovery straightforward when optimal Lagrange multipliers are known for the dual.

Principle 17.69 Suppose that \mathbf{z}^* are the optimal Lagrange multipliers on constraints $\sum_k \mathbf{a}^{(k)} \delta_k = \mathbf{0}$ in geometric programming dual [17.66]. Then

$$x_j^* \leftarrow e^{-z_j^*}$$

yields a global optimum in the corresponding primal.

For example, optimal Lagrange multipliers for the first 4 constraints of cofferdam model dual (17.77) are

$$z_1^* = -4.138, \quad z_2^* = -4.179, \quad z_3^* = -3.820, \quad z_4^* = -2.823$$

Application of transformation [17.69] recovers the same primal optimum that we have seen before:

$$\begin{aligned} x_1^* &= e^{4.138} = 62.65, & x_2^* &= e^{4.179} = 65.32 \\ x_3^* &= e^{3.820} = 45.60, & x_4^* &= e^{2.823} = 16.82 \end{aligned}$$

Derivation of the GP Dual

Why is problem [17.66] termed a dual, and why can primal optima be recovered by principle [17.69]? Begin by taking logarithms of both sides in constraints and the objective function of transformed model (17.74).

$$\begin{aligned} \min \quad & \ln(f(\mathbf{z})) \triangleq \ln \left(\sum_{k \in K_0} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} \right) \\ \text{s.t.} \quad & \ln(g_i(\mathbf{z})) \triangleq \ln \left(\sum_{k \in K_i} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} \right) \leq 0 \quad i = 1, \dots, m \\ & z_j \text{ URS} \quad \quad \quad j = 1, \dots, n \end{aligned} \tag{17.79}$$

Minimizing $\ln(f)$ is equivalent to minimizing f , and $g_i(\mathbf{z}) > 0$, so that logarithms always exist.

Noting the convexity of transformed model (17.75), this logarithmically retransformed form (17.79) is also a convex program. Karush–Kuhn–Tucker conditions will be sufficient for an optimal solution (principle [17.28](#)).

With Lagrange multipliers, ν_i , the main rows of KKT conditions are

$$\frac{1}{f(\mathbf{z})} \sum_{k \in K_0} d_k a_{k,j} e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} - \sum_{i=1}^m \frac{\nu_i}{g_i(\mathbf{z})} \sum_{k \in K_i} d_k a_{k,j} e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} = 0 \quad \text{for all } j \quad (17.80)$$

Now substituting

$$\delta_k \triangleq \frac{1}{f(\mathbf{z})} (d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}}) \quad k \in K_0 \quad (17.81)$$

$$\delta_k \triangleq \frac{-\nu_i}{g_i(\mathbf{z})} (d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}}) \quad k \in K_i, \quad i = 1, \dots, m$$

equations (17.80) become

$$\sum_{\text{all } k} a_{k,j} \delta_k = 0 \quad \text{for all } j = 1, \dots, n$$

These are exactly the first main constraints of dual [17.66](#). Sign restrictions on the ν_i and δ_k also correspond.

To make new variables δ_k perform according to their definitions (17.81), we must also enforce

$$f(\mathbf{z}) = \sum_{k \in K_0} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}}$$

or, dividing by $f(\mathbf{z})$,

$$1 = \sum_{k \in K_0} \delta_k \quad (17.82)$$

Similarly,

$$g_i(\mathbf{z}) = \sum_{k \in K_i} d_k e^{\mathbf{a}^{(k)} \cdot \mathbf{z}} \quad \text{for all } i = 1, \dots, m$$

becomes upon multiplication by $-\nu_i/g_i(\mathbf{z})$

$$-\nu_i = \sum_{k \in K_i} \delta_k \quad \text{for all } i = 1, \dots, m \quad (17.83)$$

Expressions (17.82) and (17.83) complete the constraints of dual [17.66](#).

Only the complementary slackness part of KKT conditions now remain to be formulated. Instead of explicitly including such conditions, dual formulation [17.66](#) maximizes objective function

$$\sum_{\text{all } k} \delta_k \ln \left(\frac{d_k}{\delta_k} \right) - \sum_i \nu_i \ln(-\nu_i) \quad (17.84)$$

over (δ, \mathbf{v}) , fulfilling the other constraints. A long but tedious derivation can show that this maximizes the Lagrangian of (17.79) over stationary points \mathbf{z} , which has the same effect as enforcing complementary slackness.

Signomial Extension of GPs

Often, models that cannot be formulated as a posynomial geometric program do fit a less restrictive signomial form.

Definition 17.70 Function $s(\mathbf{x})$ is a **signomial** if it can be expressed as

$$s(x_1, \dots, x_n) \triangleq \sum_k d_k \left(\prod_{j=1}^n (x_j)^{a_{k,j}} \right)$$

for given d_k and $a_{j,k}$ of arbitrary sign.

Note that term weights d_k are not required to be positive.

Obviously, the easy convexity of NLPs over transformed posynomials (principle 17.65) is lost when weights may be negative. Still, considerable tractability is preserved. The reader is referred to more advanced books on nonlinear programming for details.

EXERCISES

17-1 A lidless, rectangular box is to be manufactured from 30- by 40-inch cardboard stock sheets by cutting squares from the four corners, folding up ends and sides, and joining with heavy tape. The designer wishes to choose box dimensions that maximize volume.

- ✓ (a) Formulate this design problem as a constrained NLP.
- ✓ (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-2 A partially buried, rectangular office building is to be constructed with a volume of at least 50,000 cubic meters. To minimize energy for heating and cooling, the exterior roof and side-wall surface exposed above ground should not exceed 2250 square meters. Within these limits, the designer wishes to choose dimensions that minimize the volume excavated for the buried part of the building.

- (a) Formulate this design problem as a constrained NLP.
- ✓ (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-3 A company maintains inventories of its 5 products, replenishing the stock of an item whenever it reaches zero by manufacturing a fixed lot size of new units. The following table shows the setup cost for manufacturing, the unit volume, the unit annual inventory holding cost, and the estimated annual demand for each item.

	Product				
	1	2	3	4	5
Setup	300	120	440	190	80
Volume	33	10	12	15	26
Holding	87	95	27	36	135
Demand	800	2000	250	900	1350

Managers wish to choose lot sizes for each item that minimize total average annual setup and holding costs while assuring that the maximum combined stored volume will not exceed the 4000 cubic meters available. Assume that lots arrive the instant they are ordered.

- ✓ (a) Formulate this operations problem as a constrained NLP. (*Hint*: What is the average on hand inventory of item j as a function of the lot size for j ?)

- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-4 A print shop plans to maintain 5 different presses, replacing each every few years on a regular cycle. The following table shows the replacement cost (in thousands of dollars) of each press, and the estimated annual income (in thousands of dollars) that each can generate when new. However, as the presses grow older, their productivity declines; final values in the table show the (simple, not compound) percent income loss each year of life.

	Press				
	1	2	3	4	5
Replace	110	450	150	675	320
Income	90	110	55	220	250
Decline	5%	20%	30%	20%	40%

The owner wishes to choose a replacement (cycle) time for each press that minimizes total replacement and lost income costs within the \$250,000 she can average annually for purchasing new presses.

- (a) Formulate this replacement problem as a constrained NLP. (*Hint*: What is the average income loss on press j as a function of the replacement time for j ?)
- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-5 A machinist will remove excess metal from a rotary (round) machine part by passing the cutting tool of a lathe along 42 inches of the part length. For a lathe turning at N revolutions per minute and advancing the tool at a feed rate of f inches per revolution, classic empirical relationships project the effective life of a cutting tool (in minutes) at

$$\text{tool life} = \left(\frac{5}{Nf^{0.60}} \right)^{6.667}$$

Each time a tool wears out the operator must install a new one and spend 0.1 hour realigning the machine. Engineers wish to choose the machining plan that minimizes total cost at \$52 per hour for machinist time and \$87 each for new tools. Speed N must be in the interval [200, 600] and feed rate f in the interval [0.001, 0.005].

- (a) Formulate this machining problem as a constrained NLP.
- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-6 A warehousing firm services orders for its 5 products from an automatic storage and retrieval (ASAR) system, refilling storage from backup areas whenever the ASAR stock of any item reaches zero. The following table shows the weekly demand and the unit volume (cubic feet) for each product.

	Product				
	1	2	3	4	5
Demand	100	25	30	50	200
Volume	2	5	3	7	5

Managers wish to decide how many of each item to accommodate within the 1000 cubic feet of available storage to minimize the total number of refilling operations per week.

- (a) Formulate this operations problem as a constrained NLP.
- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-7 A solid waste company must locate 2 disposal sites to service the demand (tons per day) of the 5 communities detailed in the following table.

	Community				
	1	2	3	4	5
Demand	60	90	35	85	70
E-W coordinate	0	4	30	20	16
N-S coordinate	0	30	8	17	15

Each site will be able to handle up to 200 tons per day, and planners want to select the site locations to minimize to total ton-miles of hauling from community (mile) coordinates shown in the table to disposal sites. Assume that hauling distance is proportional to straight-line distance.

- (a) Formulate this location-allocation problem as a constrained NLP.
- (b) Use class optimization software to start from reasonable locations and compute at least a local optimum.

17-8 A light manufacturing firm is planning a new factory in a rural part of the western United States. A total of 100 employees are to be hired from the 5 surrounding communities. The following table shows the number of (equally) qualified workers available in each community and community location coordinates (in miles).

	Community				
	1	2	3	4	5
Available	70	15	20	40	30
E-W coordinate	0	10	6	1	2
N-S coordinate	0	1	8	9	3

Planners want to choose a factory site that minimizes total employee travel distance. Assume that travel distance is proportional to straight-line distance from community to factory site.

- (a) Formulate this location-allocation problem as a constrained NLP.
- (b) Use class optimization software to start from reasonable locations and compute at least a local optimum.

17-9 An investor has decided to divide his \$1.5 million portfolio among government bonds, interest-sensitive stocks, and technology stocks because some of these categories tend to increase return in periods when the others decrease. Specifically, his analysis of recent experience produced the following average returns and covariances among categories:

	Bonds	Interest-Sensitive Stocks	Technology Stocks
Mean return	5.81%	10.97%	13.02%
Covariance			
Bonds	1.09	-1.12	-3.15
Interest-sensitive stocks	-1.12	1.52	4.38
Technology stocks	-3.15	4.38	12.95

The investor wants to find the least variance way to divide his portfolio among the three categories while maintaining a 10% average return.

- (a) Formulate this portfolio problem as a constrained NLP.

- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-10 A farmer wants to allocate between 10 and 60% of his available acreage to each of corn, soybeans, and sunflowers. With markets varying wildly from year to year, he has done some research on past performance to guide his decisions. The following table shows the average return per acre and the covariances among categories that he has computed.

	Corn	Soybeans	Sunflow
Dollar return	77.38	88.38	107.50
Covariance			
Corn	1.09	-1.12	-3.15
Soybeans	-1.12	1.52	4.38
Sunflow	-3.15	4.38	12.95

The farmer wants the least risk plan that will average at least \$90 per acre.

- (a) Formulate this portfolio problem as a constrained NLP.
- (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-11 A new premium whiskey will be produced by blending up to 5 different distilling products, and the quality of the results will be measured by 3 performance indices. The following table shows the value of each index for the 5 ingredients, along with lower and upper limits for the index of the blend and costs per unit volume.

Index	Ingredient				
	1	2	3	4	5
1	12.6	15.8	17.2	10.1	11.7
2	31.4	30.2	29.6	40.4	28.9
3	115	202	184	143	169
Cost	125	154	116	189	132

Index	Blend	
	Low	High
1	12	16
2	31	36
3	121	164

The index of the blend in the first two cases will be just a volume-weighted sum of those of the chosen ingredients. However, the third index is logarithmic (i.e., the natural logarithm of the blend value will be the logarithm of the volume-weighted ingredient sum). Production planners want to choose a blend that meets upper and lower index requirements at minimum cost.

- ✓ (a) Formulate this blending problem as a constrained NLP.
- ✓ (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-12 A chemical manufacturer needs to produce 1250 barrels of a special industrial cleaning fluid by blending 5 available ingredients. The quality of the result is measured by 3 quantitative indices. The following table show the index values for each ingredient, along with the minimum and maximum required in the blend and the cost per barrel of ingredients.

Index	Ingredient				
	1	2	3	4	5
1	50.4	45.2	33.1	29.9	44.9
2	13.9	19.2	18.6	25.5	10.9
3	89.2	75.4	99.8	84.3	68.8
Cost	531	339	128	414	307

Index	Blend	
	Minimum	Maximum
1	33	43
2	17	20
3	81	99

The index of the blend in the first case will be just a volume-weighted average of those of the chosen ingredients. However, the square of the second blend index will be the square of the volume-weighted ingredient average, and the logarithm of the third blend index will be the logarithm of the volume-weighted ingredient average. Production planners want to choose a blend that meets upper and lower index requirements at minimum cost.

- (a) Formulate this blending problem as a constrained NLP.
- ✓ (b) Use class optimization software to start from a feasible solution and compute at least a local optimum.

17-13 A laser printer manufacturer can make models $i = 1, \dots, 6$ at any of plants $j = 1, \dots, 4$. The fraction of plant j capacity required per unit of printer i has been estimated for each combination at f_{ij} . The laser printer market is very competitive, so the price that can be charged for any model is a decreasing nonlinear (demand) function p_i of the total number of model i printers that are sold. Assuming these demand functions are known, formulate an NLP to determine a maximum total revenue production plan.

17-14 A new automatic storage and retrieval (ASAR)⁶ area is being added to an existing warehouse on land already owned by the company. It will have $n \geq 1$ aisles, each with pallet storage cells on both sides and a stacker crane moving in the middle which can carry a pallet to/from any location in the aisle. Storage must be provided for a total of at least p pallet cells of width w , depth d , and height h feet. All racks will be $m \geq 1$ cells high, and k cells from one end of the aisle to the other on each side. The total building height should not be more than t feet, including a clearance of u feet between the top of the racks and the ceiling. Aisle width will be 150% of pallet depth to allow for clear passage of pallets carried by the cranes, and their length should accommodate the k storage cells plus one extra pallet width to provide for an input/output station at the end. The ASAR area and its input/output stations will be enclosed with a roof and walls on three sides, being open only on the end (perpendicular to the aisles) where it adjoins the existing warehouse. Engineers want to find a minimum total cost design for the new facility using $c_1 \triangleq$ unit cost of cranes, $c_2 \triangleq$ unit cost of steel racks per pallet storage cell, together with $c_3, c_4,$ and c_5 being the construction cost per square of foundation/floors, ceilings, and sidewalls, respectively. Formulate an NLP model to choose an optimal design in terms of decision variables $n, m, k,$ and

⁶Based on J. Ashayeri, L. Gelders, and L. Van Wassenhove (1985), "A Microcomputer-Based Optimisation Model for Design of Automated Warehouses," *International Journal of Production Research*, 23, 825–839.

building exterior dimensions x (perpendicular to the aisles) y (parallel to the aisles), and z (height). Take all other symbols as constant.

✓**17-15** Assume that Syntex Laboratories⁷ is re-examining the distribution of its sales force across major pharmaceutical products $j = 1, \dots, 7$. Present force levels e_j are expected to produce s_j units of product sales per month at a profit margin of p_j per unit. However, extensive discussion and surveying has quantified impacts of changing the effort dedicated to different products. Nonlinear functions $r_j(x_j)$ predict the ratio of future to current product sales quantities as a function of $x_j \triangleq$ ratio of future to current sales effort. Formulate an NLP model in terms of decision variables x_j to determine a maximum total profit realignment of salesforce distribution keeping the total force unchanged and the force devoted to each product within $\pm 50\%$ of the current levels.

17-16 A major oil company⁸ manufactures petroleum lubricants at sites $j = 1, \dots, 10$ using a critical additive purchased from suppliers $i = 1, \dots, 15$ all over the world. Manufacturing site j requires d_j metric tons per month of the additive, and suppliers i can provide up to s_i metric tons. The transportation cost for shipping additive from supplier i to manufacturing site j is a constant $t_{i,j}$ per ton. However, the purchase cost from the supplier varies with volume. Starting with a base price c_i , supplier i reduces the unit price for all purchases from i each month by a fraction $\alpha_i > 0$ for each multiple of quantity q_i ordered. For example, the reduction might be fraction $0.5\% = 0.005$ off the base price for each 1000 metric tons ordered per month. Formulate an NLP model to find a minimum total cost procurement and shipping plan using the decision variables ($i = 1, \dots, 15; j = 1, \dots, 10$)

$x_{i,j} \triangleq$ metric tons shipped from supplier i to site j
 $y_i \triangleq$ total metric tons purchased from supplier i

17-17 A stirred tank reactor⁹ is a tank equipped with a large stirring device that is used in the

chemical and biochemical industry to produce chemical reactions. A series of 5 such tanks will be used to lower the concentration of toxic chemical from c_0 at input for tank 1, to no more than \bar{c} on exit from tank 5. A flow rate of q liters per minute will be maintained through the tank sequence, but the effect of each tank depends on its volume v_i . In particular, the units of toxic chemical removed per minute will be approximately

$$\gamma \frac{\text{output concentration}}{1 + \text{output concentration}}$$

times the tank volume. Stirred tank cost can also be estimated from the volume at $\alpha(\text{volume})^\beta$. Formulate an NLP model to find a minimum total cost sequence of tanks using the decision variables ($i = 1, \dots, 5$)

$c_i \triangleq$ output concentration of tank i
 $v_i \triangleq$ size of tank i

Assume that all other symbols are constants.

17-18 Each day q_i tons of freight arrive by sea¹⁰ in Japan bound for in-country regions $i = 1, \dots, 50$. These goods may arrive at any of the major ports $j = 1, \dots, 17$, but the internal transportation cost per ton $c_{i,j}$ varies by port and destination. The government plans a capital investment program in port facilities to pick daily tonnage processing capacities at each port j that minimize these internal transportation costs plus associated port maintenance costs, plus delay costs from freight passing through each port. Port j maintenance costs can be expressed $a_j(\text{capacity } j)^{b_j}$, where a_j and b_j are known constants. Delay cost at j can be estimated by $d[(\text{capacity } j) - (\text{traffic through } j)]$, where d is the delay cost per ton per day. Formulate an NLP model to optimize the ports using the decision variables ($i = 1, \dots, 50; j = 1, \dots, 17$)

$x_{i,j} \triangleq$ tons shipped through port j for i
 $x_j \triangleq$ total tons shipped through port j
 $y_j \triangleq$ capacity of port j

⁷Based on L. M. Lodish, E. Curtis, M. Ness, and M. K. Simpson (1988), "Sales Force Sizing and Deployment Using a Decision Calculus Model at Syntex Laboratories," *Interfaces*, 18:1, 5–20.

⁸Based on P. Ghandforoush and J. C. Loo (1992), "A Non-linear Procurement Model with Quantity Discounts," *Journal of the Operational Research Society*, 43, 1087–1093.

⁹Based on L. Ong (1988), "Heuristic Approach for Optimizing Continuous Stirred Tank Reactors in Series Using Michaelis–Menten Kinetics," *Engineering Optimization*, 14, 93–99.

¹⁰Based on M. Noritake and S. Kimura (1990), "Optimum Allocation and Size of Seaports," *Journal of Waterway, Port, Coastal and Ocean Engineering*, 116, 287–299.

17-19 Three urban neighborhoods are mutually connected by freeways admitting traffic in both directions. Net output $b_{i,k}$ (per hour) at each neighborhood k of vehicles originating at i can be estimated from known patterns ($b_{i,k} = -\sum_{k \neq i} b_{i,k}$). The delay vehicles experience on any arc (i, j) is an increasing nonlinear function $d_{i,j}$ of the total flow on that arc reflecting the number of lanes and other characteristics of the road link. Formulate an NLP to compute a “system optimal” traffic flow (i.e., one that minimizes the total delay experienced in carrying the required traffic), using the decision variables $(i, j, k = 1, \dots, 3)$

$$x_{i,j,k} \triangleq \text{flow on arc } (i, j) \text{ bound for node } k$$

(Observe that this may not be the same as a “user optimal” flow where each driver tries to minimize his or her own delay.)

17-20 The commander of a battlefront¹¹ must plan how to employ his f frontline and r reserve firepower to minimize the advance achieved over days $t = 1, \dots, 14$ by an attack of opposing forces with firepower a . Intelligence and battle simulations predict that each surviving unit of firepower in the attacking force will kill p units of defender firepower per day, and each unit of defender firepower committed to the battle will kill q units of attacker per day. Also, the kilometer advance on day t can be estimated in terms of the ratio of forces fighting that day as

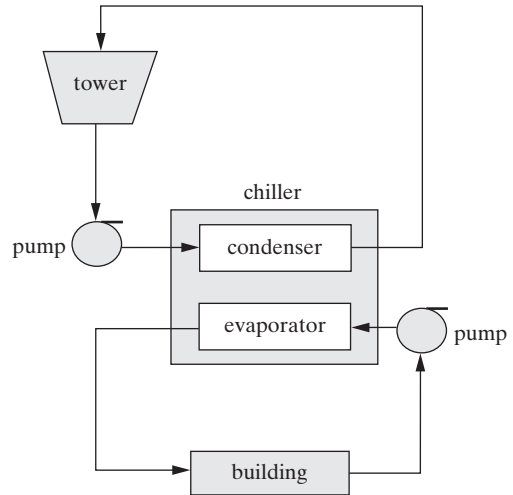
$$\exp \left[-4 \left(\frac{\text{defender firepower}}{\text{attacker firepower}} \right)^2 \right]$$

Reserves will not be in the battle on day 1, but they may be committed as desired over the ensuing days. Once committed, reserves cannot be withdrawn from the battle. Formulate an NLP model to choose an optimal plan for the defender using the decision variables $(t = 1, \dots, 14)$

$$\begin{aligned} x_t &\triangleq \text{attacker firepower fighting on day } t \\ y_t &\triangleq \text{defender firepower fighting on day } t \\ z_t &\triangleq \text{defender reserve firepower newly} \\ &\quad \text{committed on day } t \end{aligned}$$

Assume that forces are lost only to enemy fire.

17-21 Chilled-water building cooling systems¹² operate as indicated in the following sketch.



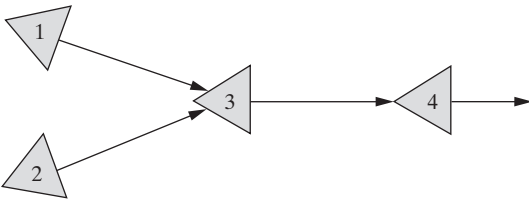
Water flows at a rate of F_1 gallons per minute around the lower loop, entering the chiller at temperature $T_{1,1}$ and being cooled to temperature $T_{1,2}$ before passing through the building. An upper loop flowing at rate F_2 gallons per minute absorbs heat within the chiller in a separate water stream, and passes it through an outdoor cooling tower to reduce its temperature from $T_{2,1}$ to $T_{2,2}$. Tower output temperature $T_{2,2}$ is a nonlinear function f_1 of $T_{2,1}, F_2$, the tower fan speed S , and the ambient air temperature T_0 . The lower loop must absorb a load H of heat within the building (measured in water temperature difference times flow). Similarly, the heat exchanged between the two loops should balance within the chiller after adjusting for extra heat generated by the unit and its pumps, which is a constant multiple k of the electrical energy they consume. That energy is, in turn, a nonlinear function f_2 of all four temperatures and both flow rates. Energy consumed in the cooling tower is another nonlinear function f_3 of fan speed. On any given day, temperature T_0 , heat load H , constant k , and all 3 functions will be known, but other system variables can be

¹¹Based on K. Y. K. Ng and M. N. Lam (1995), “Force Deployment in a Conventional Theatre-Level Military Engagement,” *Journal of the Operational Research Society*, 46, 1063–1072.

¹²Based on R. T. Olson and J. S. Liebman (1990), “Optimization of a Chilled Water Plant Using Sequential Quadratic Programming,” *Engineering Optimization*, 15, 171–191.

controlled within ranges $[\underline{T}_{i,j}, \bar{T}_{i,j}]$, $[\underline{E}_i, \bar{E}_i]$, and $[\underline{S}, \bar{S}]$, respectively. Formulate an NLP model to decide how to operate such a system at minimum total electrical consumption.

17-22 The figure below shows a system of reservoirs and hydroelectric dams of the sort operated by large utilities such as California's PG&E.¹³



Each node is a reservoir with a power plant releasing water on the downstream side. The water then requires one month to reach the reservoir of the next dam on the river system. Acre-feet inflows $b_{i,t}$ from streams feeding each reservoir i (other than the indicated rivers) can be estimated for coming months $t = 1, \dots, 3$, and reservoirs begin month 1 with storage $s_{i,0}$. Lower and upper bounds \underline{s}_i and \bar{s}_i restrict the number of acre-feet of water that should be stored in each reservoir i at all times, and similar bounds \underline{f}_i and \bar{f}_i limit the release flow through the dams. The hydropower obtained from any dam i in any month can be estimated by nonlinear function $h_i(s, f)$ reflecting both the flow f through the dam that month and the water pressure, which is a consequence of the amount of water s stored in the corresponding reservoir at the end of the month. Managers would like to operate the system to maximize the total power produced.

- (a) Sketch a time-expanded flow network for this problem with arcs $f_{i,t}$ reflecting monthly release flows at i , and other arcs $s_{i,t}$ reflecting ending storage there at month t . Label arcs with lower and upper bounds, and nodes with net inflows. Assume that all releases $f_{i,0} = 0$.
- (b) Formulate an NLP to maximize the total power produced in your network of part (a).

17-23 Determine whether each of the following NLP's is a convex program.

- ✓ (a) max $\ln(x_1) + 3x_2$
 s.t. $x_1 \geq 1$
 $2x_1 + 3x_2 = 1$
 $(x_1)^2 + (x_2)^2 \leq 9$
- (b) min $x_1 + x_2$
 s.t. $x_1, x_2 \leq 9$
 $-5 \leq x_1 \leq 5$
 $-5 \leq x_2 \leq 5$
- ✓ (c) max $x_1 + 6/x_1 + 5(x_2)^2$
 s.t. $4x_1 + 6x_2 \leq 35$
 $x_1 \geq 5, x_2 \geq 0$
- (d) min $14x_1 + 9x_2 - 7x_3$
 s.t. $6x_1 + 2x_2 = 20$
 $3x_2 + 11x_3 \leq 25$
 $x_1, x_2, x_3 \geq 0$
- ✓ (e) min $e^{x_1+x_2} - 28x_2$
 s.t. $(x_1 - 3)^2 + (x_2 - 5)^2 \leq 4$
 $14x_1 - 6x_2 = 12$
 $-2(x_1)^2 + 2x_1x_2 - (x_2)^2 \geq 0$
- (f) max $62x_1 + 123x_2$
 s.t. $\ln(x_1) + \ln(x_2) = 4$
 $7x_1 + 2x_2 = 900$
 $x_1, x_2 \geq 1$

✓ **17-24** Determine which of the NLPs in Exercise 17-23 are separable programs.

17-25 Determine whether each of the following NLPs is a quadratic program, and if so, identify the **c** and **Q** of matrix objective function form $\mathbf{c} \cdot \mathbf{x} + \mathbf{xQx}$.

- ✓ (a) min $x_1x_2 + 134/x_3 + \ln(x_1)$
 s.t. $x_1 + 4x_2 - x_3 \leq 7$
 $14x_1 + 2x_3 = 16$
 $x_1, x_2, x_3 \geq 0$
- (b) min $6(x_1)^2 + 34x_1x_2 + 5(x_2)^2$
 $-12x_1 + 19x_2$

¹³Based on Y. Ikura, G. Gross, and G. S. Hall (1986), "PG&E's State-of-the-Art Scheduling Tool for Hydro Systems," *Interfaces* 16:1, 65–82.

$$\begin{aligned} \text{s.t. } & 7x_1 + 3x_2 \geq 15 \\ & 93x_1 + 27x_2 + 11x_3 \leq 300 \\ & x_3 \geq 0 \end{aligned}$$

✓ (c) $\max \quad 2x_1x_2 + (x_2)^2 + 9x_3$
 s.t. $x_1 + x_2 + x_3 \geq 6$
 $x_j \leq 5, j = 1, \dots, 3$

(d) $\max \quad (x_1)^2 + (x_2)^2$
 s.t. $(x_1 - 10)^2 + (x_2 - 4)^2 \leq 9$
 $x_1, x_2 \geq 0$

17-26 Determine whether each of the following is a posynomial.

- ✓ (a) $23x_1 - 34x_2 + 60x_3$
 (b) $54x_1 + 89x_2 + 52x_3$
 ✓ (c) $7x_1x_2/(x_3)^{2.3} + 4\sqrt{x_1}$
 (d) $44x_1/\ln(x_2) + e^{-x_3}$

17-27 Demonstrate that each of the following NLPs is a posynomial geometric program by placing the model in standard form and detailing the sets K_i , and associated coefficients d_k and $a_{k,j}$.

- ✓ (a) $\min \quad 13x_1x_2/x_3 + 9\sqrt{x_1x_3}$
 s.t. $3x_1 + 8x_2 \leq x_3$
 $20/(x_3)^4 \leq 4$
 $x_1, x_2, x_3 > 0$
 (b) $\min \quad 40/x_1 + x_2/\sqrt{x_3}$
 s.t. $x_1x_2 \leq (x_3)^2$
 $18x_1 + 14x_2 \leq 2$
 $x_1, x_2, x_3 > 0$

17-28 Consider the nonlinear program

$$\begin{aligned} \min \quad & 8(x_1 - 2)^2 + 2(x_2 - 1)^2 \\ \text{s.t. } \quad & 32x_1 + 12x_2 = 126 \end{aligned}$$

- ✓ (a) Form the Lagrangian function for this model.
 ✓ (b) Write stationary conditions for the Lagrangian.
 ✓ (c) Solve your stationary conditions for x_1 and x_2 , and explain why your answers are optimal in the original model.
 (d) Explain why a constraint $x_1 \leq 2$ would be active if added to the original model.
 ✓ (e) Use the Lagrangian approach to compute an optimal solution to the model with the added constraint of part (d).

17-29 Do Exercise 17-28 for the NLP

$$\begin{aligned} \max \quad & 300 - 5(x_1 - 20)^2 - 4(x_2 - 6)^2 \\ \text{s.t. } \quad & x_1 + x_2 = 8 \end{aligned}$$

and part (d) extra constraint $x_2 \geq 0$.

17-30 State the Karush–Kuhn–Tucker optimality conditions for each of the following mathematical programs.

✓ (a) $\min \quad 14(x_1 - 9)^2 + 3(x_2 - 5)^2 + (x_3 - 11)^2$
 s.t. $2x_1 + 18x_2 - x_3 = 19$
 $6x_1 + 8x_2 + 3x_3 \leq 20$
 $x_1, x_2 \geq 0$

(b) $\max \quad 6x_1 + 40x_2 + 5x_3$
 s.t. $x_1 \sin(x_2) + 9x_3 \geq 2$
 $e^{18x_1 + 3x_2} + 14x_3 \leq 50$
 $x_2, x_3 \geq 0$

✓ (c) $\min \quad 100 - (x_1 - 3)^2 - (x_2)^4 + 19x_3$
 s.t. $5(x_1 - 1)^2 + 30(x_2 - 2)^2 \geq 35$
 $60x_2 + 39x_3 = 159$
 $x_1, x_2, x_3 \geq 0$

(d) $\max \quad 7 \ln(x_1) + 4 \ln(x_2) + 11 \ln(x_3)$
 s.t. $(x_1 + 2)^2 - x_1x_2 + (x_2 - 7)^2 \geq 80$
 $5x_1 + 7x_3 \geq 22$
 $x_1, x_2, x_3 \geq 3$

✓ **17-31** For each mathematical program in Exercise 17-30, determine whether principle 17.26 assures that a KKT point is a global optimum.

17-32 Consider the NLP

$$\begin{aligned} \min \quad & 15(x_1)^2 + 4(x_2)^2 \\ \text{s.t. } \quad & 3x_1 + 2x_2 = 8 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- ✓ (a) State the KKT optimality conditions for this model.
 (b) Verify that at solution $\mathbf{x} = (0, 4)$ there exists an improving feasible direction $\Delta \mathbf{x} = (2, -3)$.
 (c) Confirm that KKT conditions have no solution for the nonoptimal \mathbf{x} of part (b).
 (d) Explain why every local optimum of the model must be a KKT point.
 ✓ (e) Show that global optimal solution $\mathbf{x}^* = (1, \frac{5}{2})$ is a KKT point.

17-33 Do Exercise 17-32 for NLP

$$\begin{aligned} \max \quad & 2 \ln(x_1) + 8 \ln(x_2) \\ \text{s.t.} \quad & 4x_1 + x_2 = 8 \\ & x_1, x_2 \geq 1 \end{aligned}$$

with nonoptimal point $\mathbf{x} = (1, 4)$, improving feasible direction $\Delta \mathbf{x} = (-1, 4)$, and global optimum $\mathbf{x}^* = (1, 4)$.

✔ **17-34** Use absolute value (unsquared) penalty functions to reduce each NLP of Exercise 17-30 to an unconstrained penalty model.

✔ **17-35** Do Exercise 17-34 using squared penalty functions.

17-36 Consider the NLP

$$\begin{aligned} \min \quad & 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2 \\ \text{s.t.} \quad & (x_1)^2 + (x_2)^2 \leq 4 \\ & 0 \leq x_1 \leq 2, x_2 \geq 0 \end{aligned}$$

with optimal solution $\mathbf{x}^* = (1.088, 1.678)$.

✔ (a) Use unsquared penalty functions to reduce this problem to an unconstrained penalty model.

✔ (b) Explain why local minima of the unconstrained model in part (a) must be global minima for all $\mu \geq 0$.

✔ (c) Determine whether the penalty objective of part (a) is differentiable. Explain.

✔ (d) Determine whether there will be a penalty multiplier μ large enough that the unconstrained optimum in part (a) is optimal in the original model. Explain.

✔ (e) Suppose that we are solving the given constrained NLP by the sequential unconstrained penalty Algorithm 17A. Explain why it is reasonable to begin with multiplier $\mu = 0.5$ and increase it by a factor $\beta = 2$ after each unconstrained optimization.

✔ (f) Use class optimization software to apply Algorithm 17A, starting at $\mathbf{x} = (3, 5)$ and managing the penalty multiplier as in part (e).

✔ **17-37** Do Exercise 17-36 using squared penalty functions. Stop the search in part (f) when the total constraint violation is ≤ 0.2 .

17-38 Do Exercise 17-36 for the NLP

$$\begin{aligned} \max \quad & 100 - 8(x_1)^2 - 3(x_2 - 3)^2 \\ \text{s.t.} \quad & x_2 \geq 2/x_1 \\ & 0 \leq x_1 \leq 2 \\ & 0 \leq x_2 \leq 2 \end{aligned}$$

Start at $\mathbf{x}^{(0)} = (2, 2)$ with multiplier $\mu = 0.5$, and increase by the factor $\beta = 4$.

17-39 Do Exercise 17-38 using squared penalty functions. Stop the search in part (f) when total constraint violation ≤ 0.2 .

✔ **17-40** Determine whether barrier methods can be applied to each of the NLPs in Exercise 17-23, and if so, use log barrier functions to reduce the constrained optimization model to an unconstrained barrier model.

✔ **17-41** Do Exercise 17-40 using reciprocal barrier functions.

17-42 Consider solving the NLP of Exercise 17-36 by barrier methods.

✔ (a) Use logarithmic barrier functions to reduce this problem to an unconstrained barrier model.

✔ (b) Explain why local minima of the unconstrained model in part (a) for all $\mu \geq 0$ must be global minima.

✔ (c) Determine whether the barrier objective of part (a) is differentiable. Explain.

✔ (d) Determine whether there will be a barrier multiplier $\mu > 0$ small enough that the unconstrained optimum in part (a) is optimal in the original model. Explain.

✔ (e) Suppose that we are solving the given constrained NLP by the sequential unconstrained barrier Algorithm 17B. Explain why it is reasonable to begin with multiplier $\mu = 2$ and decrease it by a factor $\beta = \frac{1}{4}$ after each unconstrained optimization.

✔ (f) Use class optimization software to apply Algorithm 17B, starting at $\mathbf{x}^{(0)} = (3, 5)$ and managing the barrier multiplier as in part (e). Proceed while $\mu \geq \frac{1}{32}$.

✔ **17-43** Do Exercise 17-42 using reciprocal barrier functions.


17-44 Do Exercise 17-42 for the NLP of Exercise 17-38. Start at $\mathbf{x}^{(0)} = (1.8, 1.8)$ with multiplier $\mu = 8$, and decrease with factor $\beta = \frac{1}{4}$.

17-45 Do Exercise 17-44 using reciprocal barrier functions.

17-46 Consider the nonlinear program

$$\begin{aligned} \min \quad & (x_1 - 8)^2 + 2(x_2 - 4)^2 \\ \text{s.t.} \quad & 2x_1 + 8x_2 \leq 16 \\ & x_1 \leq 7 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- ✔ (a) Introduce slack variables x_3 and x_4 to place the model in standard form for reduced gradient Algorithm 17C.
- ✔ (b) Show that x_2 and x_4 form a basic set of variables for your standard form.
- ✔ (c) Assuming the basis of part (b), classify variables as basic, nonbasic, or superbasic at initial solution $\mathbf{x}^{(0)} = (0, 1, 8, 7)$.
- ✔ (d) Compute the reduced gradient corresponding to the basis and $\mathbf{x}^{(0)}$ of part (c).
- ✔ (e) Construct the move direction that would be pursued by Algorithm 17C at the basis and $\mathbf{x}^{(0)}$ of part (c).
- ✔ (f) Compute the maximum feasible step λ in the direction of part (e). Then, assuming (correctly) that the direction of part (e) remains improving all the way to the maximum λ , compute the resulting new solution $\mathbf{x}^{(1)}$.
- ✔ (g) Explain why a basis change would be required by Algorithm 17C at the $\mathbf{x}^{(1)}$ of part (f), and choose an appropriate new basis.


 **17-47** Return to the standard form NLP of Exercise 17-46(a).

- ✔ (a) Apply reduced gradient Algorithm 17C to compute an optimal solution starting from the $\mathbf{x}^{(0)} = (0, 1, 8, 7)$.
- (b) Graph your progress in a plot of the feasible (x_1, x_2) .

17-48 Do Exercise 17-46 for nonlinear program

$$\begin{aligned} \max \quad & 500 - 3(x_1 + 1)^2 + 2x_1x_2 - (x_2 - 10)^2 \\ \text{s.t.} \quad & x_1 - x_2 \leq 1 \\ & x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

using basis $\{x_1, x_4\}$ and standard-form starting solution $\mathbf{x}^{(0)} = (2, 1, 0, 4)$.

 **17-49** Do Exercise 17-47 on the standard-form NLP of Exercise 17-48(a).

17-50 Consider the equality-constrained quadratic program


$$\begin{aligned} \min \quad & 6(x_1)^2 + 2(x_2)^2 - 6x_1x_2 + 4(x_3)^2 \\ & + 5x_1 + 15x_2 - 16x_3 \\ \text{s.t.} \quad & x_1 + 3x_2 - 2x_3 = 2 \\ & 3x_1 - x_2 + x_3 = 3 \end{aligned}$$

- ✔ (a) Identify the \mathbf{Q} , \mathbf{c} , \mathbf{A} , and \mathbf{b} of (symmetric) quadratic program standard form.


- ✔ (b) State Karush–Kuhn–Tucker optimality conditions for the model as a system of linear equalities.
- ✔ (c) Solve your system of part (b) for the unique KKT point of the model.

17-51 Do Exercise 17-50 for the equality-constrained quadratic program

$$\begin{aligned} \max \quad & -(x_1)^2 - 8(x_2)^2 - 2(x_3)^2 + 10x_2x_3 \\ & + 14x_1 - 8x_2 + 20x_3 \\ \text{s.t.} \quad & x_1 + 4x_3 = 4 \\ & -x_2 + 3x_3 = 1 \end{aligned}$$

 **17-52** Return to the NLP of Exercise 17-46, and consider solving by active set Algorithm 17D starting from solution $\mathbf{x}^{(0)} = (0, 1)$.

- ✔ (a) Demonstrate that the model is a quadratic program by deriving the c_0 , \mathbf{c} , \mathbf{Q} , $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(4)}$, b_1, \dots, b_4 , G , L , and E of general symmetric form [17.50](#).
- ✔ (b) State and solve as a system of linear equalities the active set optimality conditions [17.53](#) corresponding to initial solution $\mathbf{x}^{(0)}$.
- ✔ (c) Determine the step λ that would be applied to the direction resulting from part (b), and compute the new point $\mathbf{x}^{(1)}$.
- ✔ (d) Verify by forming and solving the optimality conditions [17.53](#) corresponding to $\mathbf{x}^{(1)}$ that no further progress can be made if all inequalities active at $\mathbf{x}^{(1)}$ of part (c) are included in the active set S .
- ✔ (e) Use the results of part (d) to show which active constraint should be dropped from S .
- ✔ (f) Begin from part (e) and complete the solution of this quadratic program.
- (g) Graph your progress in a plot of the feasible (x_1, x_2) .
- (h) Compare the evolution of active set Algorithm 17D in part (g) with corresponding reduced gradient Algorithm 17C computations in Exercise 17-47.

 **17-53** Do Exercise 17-52 for the NLP of Exercise 17-48 starting from solution $\mathbf{x}^{(0)} = (2, 1)$.

17-54 Form linear programming approximations [17.59](#) to each of the following separable programs using breakpoints $u_{1,0} = 0$, $u_{1,1} = 1$, $u_{1,2} = 3$, $u_{2,0} = 0$, $u_{2,1} = 2$, $u_{2,2} = 4$.

- ✓ (a) $\min x_1/(4 - x_1) + (x_2 - 1)^2$
 s.t. $2x_1 + x_2 \geq 2$
 $4(x_1 + 1)^3 - 9(x_2)^2 \leq 25$
 $0 \leq x_1 \leq 3$
 $0 \leq x_2 \leq 4$
- (b) $\max 500 - (x_1 - 1)^2 - 25/(x_2 + 1)$
 s.t. $\sqrt{x_1} - (x_2 + 1)^2 \geq -3$
 $6x_1 + 2x_2 \leq 10$
 $0 \leq x_1 \leq 3$
 $0 \leq x_2 \leq 4$

17-55 Consider the trivial separable program

$$\begin{aligned} \min \quad & 2(x - 3)^2 \\ \text{s.t.} \quad & 0 \leq x \leq 6 \end{aligned}$$

- (a) Verify that the model is a convex program.
- ✓ (b) Verify by inspection that an optimal solution occurs at $x^* = 3$.
- ✓ (c) Form a linear programming approximation [\[17.61\]](#) using $u_0 = 0, u_1 = 2, u_2 = 6$.
- ✓ (d) Solve your LP approximation of part (c) by inspection, and determine whether correctness condition [\[17.62\]](#) is satisfied by the approximate optimum.
- (e) Discuss how convexity of part (a) relates to correct sequencing [\[17.62\]](#) in part (d).
- (f) Verify by inspection that $x^* = 0$ and $x^* = 6$ are alternative optima in the original NLP when the objective is maximized instead of minimized.
- ✓ (g) Repeat part (d), this time maximizing the objective function.
- (h) Comment on the errors introduced in objective function and other values when sequencing condition [\[17.62\]](#) is violated in part (g).

17-56 Consider the standard-form posynomial geometric program

$$\begin{aligned} \min \quad & 3/\sqrt{x_1} + x_1x_2 + 10/(x_3)^3 \\ \text{s.t.} \quad & 0.5x_1x_2/(x_3)^2 \leq 1 \\ & 0.167x_1 + 0.25(x_1)^{0.4}x_2 + 0.0833x_3 \leq 1 \\ & x_1, x_2, x_3 > 0 \end{aligned}$$

- ✓ (a) Change variables to convert this geometric program into a convex program.

- ✓ (b) Use class optimization software to solve your convex program of part (a) and transform optimal variable values back to obtain an optimal solution for the original NLP.
- ✓ (c) Form the geometric programming dual of the original NLP.
- ✓ (d) Determine the degree of difficulty of the original NLP.
- ✓ (e) Use class optimization software to solve the dual of part (c) and retrieve an optimal primal solution from the corresponding Lagrange multipliers.

17-57 Do Exercise 17-56 for the posynomial geometric program

$$\begin{aligned} \min \quad & 10/(x_1x_2x_3)^2 \\ \text{s.t.} \quad & 12(x_1)^2x_2 + 4x_3 \leq 1 \\ & 0.1x_2\sqrt{x_1} + x_2x_3 \leq 1 \\ & (x_1x_2)^{0.333} \leq 1 \\ & x_1, x_2, x_3 > 0 \end{aligned}$$

17-58 A water distribution system¹⁴ is a network with (positive = forward or negative = reverse) flows $x_{i,j}$, in pipes between nodes $i, j = 0, \dots, m$ representing storage tanks and pipe intersections. Pressures at the nodes i can be measured in hydraulic “head,” which is the height to which water will rise in an opened vertical pipe installed at the node, relative to the “ground node” 0. Heads have assigned values s_i for storage tank nodes i , and net outflows r_i are established for all nodes ($\sum_{i=0}^m r_i = 0$). The ground node 0 is connected to each storage node i by an arc $(0, i)$, and to no others. To determine how the system will perform at steady state, engineers need to find flows $f_{i,j}$ and heads h_i that (i) maintain net flow balance at every node, (ii) achieve assigned heads s_i at storage nodes, and (iii) satisfy nonlinear head-to-flow equations

$$h_j - h_i = \phi_{i,j}(f_{i,j}) \quad \text{nonground } (i, j)$$

where functions $\phi_{i,j}(x_{i,j})$ are known relations between head difference and the flow on particular arcs (i, j) that reflect length, size, pumping, grade, and other characteristics.

- (a) Formulate a related NLP over unrestricted flows $x_{i,j}$ having only flow balance

¹⁴Based on M. Collins, L. Cooper, R. Helgason, J. Kennington, and L. LeBlanc (1978), “Solving the Pipe Network Analysis Problem Using Optimization Techniques,” *Management Science*, 24, 747–760.

constraints at all nodes and a minimizing objective function summing terms

$$f_{i,j}(x_{i,j}) \begin{cases} s_j x_{0,j} & \text{arcs } (0, j) \\ \int_0^{x_{i,j}} \phi_{i,j}(z) dz & \text{other } (i, j) \end{cases}$$

- (b) Explain why your NLP of part (a) is a separable program. (With mild assumptions on the $\phi_{i,j}$ it can also be shown to be convex.)
- (c) State Karush–Kuhn–Tucker conditions for the primal model of part (a) and explain why they must be satisfied by a locally optimal \mathbf{x}^* .
- (d) Interpret conditions of part (c) to show that a solution to steady-state equation system (i)–(iii) above can be obtained from locally optimal flows \mathbf{x}^* in part (a) and corresponding KKT multipliers \mathbf{v}^* .

17-59 Return to the NLP of Exercise 17-36, and consider solving it by Sequential Quadratic Programming Algorithm 17E.

REFERENCES

Bazarra, Mokhtar, Hanif D. Sherali, and C. M. Shetty (2006). *Nonlinear Programming - Theory and Algorithms*, Wiley Interscience, Hoboken, New Jersey.

Griva, Igor, Stephen G. Nash, and Ariela Sofer (2009). *Linear and Nonlinear Optimization*, SIAM, Philadelphia, Pennsylvania.

- ✓ (a) Using dual variables v_1 , roll constraints into the objective function to formulate the Lagrangian.
- ✓ (b) Formulate the second-order the Lagrangian as a function of move direction $\Delta \mathbf{x}$ as in principle [17.59].
- ✓ (c) Formulate the quadratic program [17.60] that must be solved at each iteration of SQP, and justify each element.
- ✓ (d) Specialize your formulation of (c) to state the quadratic subproblem that would result at iteration 1 if $\mathbf{x}^{(0)} = (1, 1)$ and all multipliers $v_1 = 0$.

17-60 Do Exercise 17-59 on the NLP of Exercise 17-38, again using $\mathbf{x}^{(0)} = (1, 1)$ and all multipliers $v_1 = 0$ in part (d).

Luenberger, David G. and Yinyu Ye (2008), *Linear and Nonlinear Programming*, Springer, New York, New York.

This page intentionally left blank

Group Projects

This textbook focuses broadly on developing two kinds of skills in students. First is **modeling** the enormous variety of applied settings where optimization methods can yield practically useful insights. Then, **analysis** using the variety of algorithms and mathematical insights available to obtain and understand best possible solutions to fully formulated models. The two are closely related because choices made in formulation can dramatically impact the tractability of models to available solution methods as well as the validity of results obtained.

This addendum proposes projects that can be accomplished by small teams of students in order to deepen their experience with the use of optimization methods in applied engineering and management settings within the time available in busy courses. A first category focuses on settings already referenced in the book's numbered Applications and Exercises. The second, more wide-ranging category of project topics, places the burden and flexibility on students to find their own published report of a application setting. Then they must distill it into a usable starting point parallel to those within the text before further investigation can proceed.

Instructors may choose to assign projects focused on particular types of optimization models (LP, ILP, NLP, etc.) as those arise in the course timeline. Alternatively, they may prefer to assign a term project with students having freedom to take on any category of models treated in the course.

1. PROJECTS BASED ON REFERENCES IN THIS BOOK

Numbered Applications in the chapters of this book tell simplified stories of how optimization methods can be employed in a variety of applied domains to meet the needs of decision makers therein. Most have explicit footnote references to a published paper about the underlying investigation. Exercises at the ends of chapters with similar footnote references introduce a host of additional domain scenarios where appropriate optimization modeling and analysis can prove valuable.

Assignment

Choose one of those simplified environments depicted in an Application or footnoted Exercise as the focus of your project. Next obtain a copy of the referenced paper (if there is one), and also research related sources online and in other published papers (see the journal list below) to enrich your understanding of the real problem domain and the opportunity to usefully employ optimization techniques there.

Then prepare a project report describing how optimization methods and analysis can be applied more deeply and completely to addressing the needs of real

decision makers confronted with the challenge sketched in the book development. Specifically,

(a) Determine and justify whether, and if so how, the simplified story and any constant parameter values invented in the text need to be enhanced to validly address the needs of decision makers in the described problem environment. (Justify all answers and reference sources.)

- Are important domain elements over simplified or left out of the text discussion that should be treated more completely? If so, how can remedial adjustments be made to the model without doing too much damage to tractability?
- What scales (number of locations, employees, products, customers, processes, time periods, etc.) are needed for a realistic instance of the applied environment?
- What are appropriate choices for values of model parameters (costs, profits, capacities, yields, demands, etc.)?
- If you have chosen a multi-objective case, make it more manageable for the project by showing how it can be reformulated as a goal program or one with a single-weighted-sum of objectives. Justify your choices of goal targets or objective weights in doing so.

(b) Formulate and justify a new model consistent with your recommendations of part (a) and having all scale and other parameter values explicit as data. Be sure it is feasible and bounded in objective value. If a truly realistic scale exceeds the capacity of available software, choose and justify reduced-scale parameter values to make the solution possible.

(c) Solve your updated model of part (b) with class optimization software or general-purpose tools like spreadsheets, to obtain an exact or approximate optimal solution. Then, fully explain the results obtained.

(d) Taking account of the results obtained in part (c), evaluate whether a revised model and/or a new choice of parameters might give superior results for decision makers. If so, modify those inputs, re-solve, and compare with first results to highlight any gains you see in the new results.

(e) Document all your findings and analysis in a written project report, and prepare a PowerPoint or similar presentation suitable for delivery to the full class.

2. PROJECTS BASED ON A NEW REFERENCE CHOSEN BY THE STUDENT TEAM

The list below that follows enumerates engineering and management journals often describing real cases of optimization modeling and analysis.

Assignment

Use available online and library resources to peruse those and related published sources and choose an article on which to base your project. The article should not come from any text or similar book. It should be clear that the authors of the paper

were thinking about a real applied challenge rather than merely contriving a vehicle for demonstrating mathematical methods and properties.

If the article develops more than one model, choose one to pursue. Then study the paper and model, and find related material online about the application domain and related research, to prepare a project report describing how optimization modeling and analysis can best contribute to addressing the needs of decision makers confronted with the challenge discussed. (Justify all answers and reference sources.)

(a) Begin by distilling material in your chosen paper into a suitable project starting point:

- Briefly describe the “story” based upon the chosen model and paper in a fashion similar to those presented throughout the text. What is being decided? What constraints must be considered? What objective(s) determine preferred solutions? Where do parameter values come from?
- Formulate a preliminary model of your story in standard mathematical format, clearly defining and explaining all indexes, decision variables, symbolic parameters, constraints, and objective functions.
- Then proceed from that preliminary model as if it had been taken from an Application or Exercise in the text.

(b) Defend and refine your simplified model of part (a) to assess how it can validly address the needs of decision makers in the described problem environment. (Justify all answers and reference sources.)

- Are important domain elements over simplified or left out of your story and model that should be treated more completely? If so, how can remedial adjustments be made to the model without doing too much damage to tractability?
- What scales (number of locations, employees, products, customers, processes, time periods, etc.) are needed for a realistic instance of the problem?
- What are appropriate choices for values of the model parameters (costs, profits, capacities, yields, demands, etc.)?
- If you have chosen a multi-objective case, make it more manageable for the project by showing how it can be reformulated as a goal program or one with a single-weighted-sum of objectives. Justify your choices of goal targets or objective weights in doing so.

(c) Formulate and justify a full model consistent with your recommendations of part (b) and having all scale and other parameter values explicit as data. Be sure it is feasible and bounded in objective value. If a truly realistic scale exceeds the capacity of available software, choose and justify reduced-scale parameter values to make solution possible.

(d) Solve your updated model of part (c) with class optimization software or general-purpose tools like spreadsheets, to obtain an exact or approximate optimal solution. Then, fully explain the results obtained.

(e) Taking account of the results obtained in part (d), evaluate whether a revised model and/or a new choice of parameters might give superior results for decision

makers. If so, modify those inputs, re-solve, and compare with first results to highlight any gains you see in the new outcomes.

(f) Document all your findings and analysis in a written project report, and prepare a PowerPoint or similar presentation suitable for delivery to the full class.

Journal Sources for Optimization Applications

Descriptions of optimization applications can be found in many scientific journals. The following is a partial list in rough order of choice as a source for the above project.

European Journal of Operational Research

Operational Research (formerly Operational Research Quarterly)

IIE (Institute of Industrial Engineers) Transactions

IIE (Institute of Industrial Engineers) Transactions on Healthcare Systems Engineering

Interfaces

Operations Research

Computers and Operations Research

International Journal of Production Research

Computers and Industrial Engineering

Transportation Science

Healthcare Management Science

Management Science

Naval Research Logistics

Omega

Decision Sciences

Opsearch (Indian Journal of Operations Research)

INFOR (formerly Canadian Journal of Operational Research)

Marketing Science

Manufacturing and Service Operations Management

Selected Answers

Chapter 1

1-1 (a) s **(b)** d, p and b **(c)** $\min(d/s)^2$ **(d)** $ps \leq b$, s nonnegative and integer

1-2 (a) feasible and optimal **(b)** neither because infeasible **(c)** feasible but not optimal

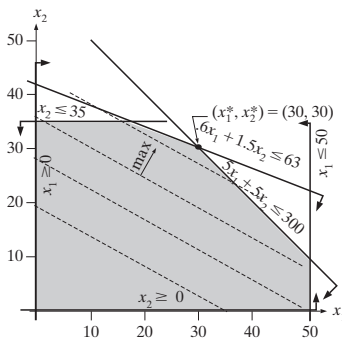
1-5 (a) exact numerical optimization **(c)** closed-form optimization

1-8 (b) 16.4 hours; 166.1 days; 110.6 years; 6.5 million years

1-9 (a) random variable **(c)** deterministic **(e)** deterministic **(g)** random variable **(i)** deterministic

Chapter 2

2-1 (a) $\max 200x_1 + 350x_2$, s.t. $5x_1 + 5x_2 \leq 300$, $0.6x_1 + 1.5x_2 \leq 63$, $x_1 \leq 50$, $x_2 \leq 35$, $x_1 \geq 0$, $x_2 \geq 0$ **(b)** x_1^* = basic = 30, x_2^* = deluxe = 30 **(c)**



(d) all optimal from $\mathbf{x} = (30, 30)$ to $\mathbf{x} = (17.5, 35)$

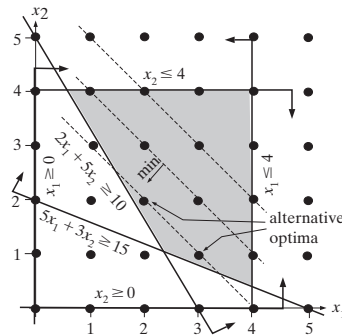
2-2 (b) x_1^* = domestic = \$5 million, x_2^* = foreign = \$7 million

2-3 (b) x_1^* = Squawking Eagle = 40 thousand, x_2^* = Crooked Creek = 10 thousand

2-4 (b) x_1^* = beef = 25g, x_2^* = chicken = 100g

2-5 (b) v^* = 7000, c^* = 0

2-6 (a) $\min x_1 + x_2$, s.t. $5x_1 + 3x_2 \geq 15$, $2x_1 + 5x_2 \geq 10$, $0 \leq x_1 \leq 4$, $0 \leq x_2 \leq 4$, x_1, x_2 integer **(b)** partial patterns make no physical sense **(c)** Either $x_1^* = x_2^* = 2$, or $x_1^* = 3, x_2^* = 1$ **(d)**



2-7 (a) $\min 16x_1 + 16x_2$, s.t. $x_1x_2 = 500$, $x_1 \geq 2x_2$, $x_2 \leq 15$, $x_1 \geq 0$, $x_2 \geq 0$

(b) x_1^* = length = $33\frac{1}{3}$ feet, x_2^* = width = 15 feet **(d)** $x_1 \leq 25$ leaves no feasible

2-8 (b) x_1^* = diameter = 78.16 feet, x_2^* = floors = 31.26

2-9 (b) $\min x_2$ **(c)** $\min x_1 + x_2$ **(d)** $\max x_2$ **(e)** $x_2 \leq 1/2$

2-11 (a) $\min \sum_{i=3}^4 i \sum_{j=1}^2 y_{i,j}$ **(c)** $\max \sum_{i=1}^p \alpha_i y_{i,4}$

(e) $\sum_{j=1}^4 y_{i,j} = s_i$, $i = 1, \dots, 3$

2-12 (a) $\sum_{i=1}^{17} x_{i,j,t} \leq 200$, $j = 1, \dots, 5$;

$t = \dots, 7$; 35 constraints **(b)** $\sum_{j=1}^5 \sum_{t=1}^7 x_{5,j,t}$

≤ 4000 ; 1 constraint **(c)** $\sum_{j=1}^5 x_{i,j,t} \geq 100$,

$i = 1, \dots, 17$; $t = 1, \dots, 7$; 119 constraints

2-13 model; param m; param n;

param p; set products := 1..m;

```

set lines := 1..n; set weeks :=
1..p; var x{i in products, j in
lines, t in weeks} >= 0; subject to
linecap {j in lines, t in weeks}:
sum {i in products} x[i, j, t] <= 200;
prod5lim: sum {j in lines,
t in weeks} x[5, j, t] <= 4000;
minprodn{i in products, t in weeks}:
sum {j in lines} x[i, j, t] >= 100;
data; param m := 17; param n := 5;
param p := 7;

```

2-16 (a) $f(y_1, y_2, y_3) \triangleq (y_1)^2 y_2 / y_3$,
 $g_1(y_1, y_2, y_3) \triangleq y_1 + y_2 + y_3, b_1 = 13$,
 $g_2(y_1, y_2, y_3) \triangleq 2y_1 - y_2 + 9y_3, b_2 = 0$,
 $g_3(y_1, y_2, y_3) \triangleq y_1, b_3 = 0, g_4(y_1, y_2, y_3) \triangleq y_3$,
 $b_4 = 0$

2-17 (a) linear **(c)** nonlinear **(e)** nonlinear
(g) nonlinear

2-18 (a) LP **(c)** NLP

2-19 (a) continuous **(c)** discrete

2-20 (a) $\sum_{j=1}^8 x_j = 3$ **(c)** $x_3 + x_8 \leq 1$

2-21 (a) max $85x_1 + 70x_2 + 62x_3 + 93x_4$,
s.t. $700x_1 + 400x_2 + 300x_3 + 600x_4 \leq 1000$,
 $x_j = 0$ or $1, j = 1, \dots, 4$ **(b)** fund 2 and 4,
i.e., $x_1^* = x_3^* = 0, x_2^* = x_4^* = 1$

2-22 (b) build 3 and 4

2-23 (a) ILP **(c)** INLP **(e)** INLP **(g)** LP

2-24 (a) model (b) **(c)** model (d)

2-25 (a) Alternative optima from $x_1^* = 8, x_2^* = 0$
to $x_1^* = 8, x_2^* = 12$ **(b)** Unique optimum $x_1^* = 0$,
 $x_2^* = 4$ **(c)** Helping one can hurt the other.

2-27 (b) nonzeros: $x_5^* = 1000, x_{12}^* = 15000$

2-28 (i) $x_1^* = x_2^* = x_3^* = 1100, x_4^* = x_6^* = 1500$,
 $x_5^* = 1400, x_7^* = 400, x_8^* = x_{10}^* = 0, x_9^* = 1900$

2-29 (h) $x_1^* = x_3^* = x_6^* = x_7^* = 1$, others = 0

2-30 (h) nonzeros: $x_{1,1}^* = 81, x_{1,2}^* = 93$,
 $x_{1,3}^* = 166, x_{1,5}^* = 90, x_{1,6}^* = 88, x_{1,7}^* = 145$,
 $x_{2,2}^* = 301, x_{3,1}^* = 166, x_{3,4}^* = 105, x_{4,3}^* = 99$

2-33 (g) $x_2^* = x_3^* = x_4^* = 1$, others = 0

2-35 (h) nonzero values: $x_{4,2}^* = 115, x_{4,3}^* = 165$,
 $x_{5,1}^* = 85, x_{5,3}^* = 225$

2-37 (i) $x_{10}^* = 50, x_{15}^* = 25, x_{20}^* = 5, y_1^* = 5000$,
 $y_2 = 8500$

2-39 (i) nonzeros: $x_{2,2}^* = x_{2,4}^* = x_{3,1}^* = x_{3,3}^* =$
 $x_{5,5}^* = 1, y_2^* = y_3^* = y_5^* = 1$

2-40 (a) max $\sum_{j=1}^8 r_j x_j$, subject to, $\sum_{j=1}^8 x_j \leq 4$,
 $x_1 + x_2 + x_3 \geq 2, x_4 + x_5 + x_6 + x_7 + x_8 \geq 1$,

$x_2 + x_3 + x_4 + x_8 \geq 2, x_1 \dots x_8 = 0$ or 1
(b) model; param n ; set games :=
1..n; #ratings param r{j in games};
#home? param h{j in games}; #state?
param s{j in games}; #cover? var
x{j in games} binary; maximize
totrat: sum{j in games} r[j]*x[j];
subject to capacity: sum{j in games}
x[j] <= 4; home: sum{j in games}
h[j]*x[j] >= 2; away: sum{j in games}
(1 - h[j])*x[j] >= 1; state:
sum{j in games} s[j]*x[j] >= 2; data;
param n := 8; param r := 1 3.0 2
3.7 3 2.6 4 1.8 5 1.5 6 1.3 7 1.6 8 2.0;
param h := 1 1 2 1 3 1 4 0 5 0 6 0
7 0 8 0; param s := 1 0 2 1 3 1 4 1
5 0 6 0 7 0 8 1; **(c)** The model is an ILP
because all constraints and the objective are
linear, but decision variables are binary.

2-43 max $199x_1 + 229x_2 + 188x_3 + 205x_4 -$
 $180y_1 - 224y_2 - 497y_3$, subject to,
 $23x_3 + 41x_4 \leq 2877y_1, 14x_1 + 29x_2 \leq 2333y_2$,
 $11x_3 + 27x_4 \leq 3011y_3, x_1 + x_2 + x_3 + x_4 \geq 205$,
 $y_1 + y_2 + y_3 \leq 2, x_1, \dots, x_4 \geq 0, y_1, \dots, y_3 = 0$
or 1

Chapter 3

3-1 (a) feasible and local max; infeasible; feasi-
ble; feasible, local, and global max

3-2 (a) $\mathbf{y}^{(1)} = (8, -2, 5), \mathbf{y}^{(2)} = (3, 8, 10)$,
 $\mathbf{y}^{(3)} = (3, 11, 10)$

3-3 (a) $\Delta \mathbf{w}^{(1)} = (4, -2, 6), \Delta \mathbf{w}^{(2)} = (0, -2, 8)$,
 $\Delta \mathbf{w}^{(3)} = (-1, 0, 3)$

3-4 (a) nonimproving **(c)** improving
(e) nonimproving

3-5 (a) feasible **(c)** feasible **(e)** feasible

3-6 (a) $\lambda = 3$; not unbounded **(c)** $\lambda = +\infty$;
unbounded

3-7 (a) improving **(c)** need more information
(e) improving

3-8 (a) $\Delta \mathbf{w} = (3, -2, 0, 1)$ **(c)** $\Delta \mathbf{w} = (-8, 3)$

3-9 (a) [ii], [iv]

3-10 (a) feasible **(c)** infeasible

3-11 (a) $2\Delta w_1 + 3\Delta w_3 = 0, 1\Delta w_1 + 1\Delta w_2 +$
 $2\Delta w_3 = 0, \Delta w_1 \geq 0$ **(c)** $1\Delta w_1 + 1\Delta w_2 = 0$,
 $2\Delta w_1 - 1\Delta w_2 \geq 0$

3-12 (a) $-1\Delta y_1 + 5\Delta y_2 < 0$ **(b)** substitution
(c) $y_1 \geq 0, -y_1 + y_2 \leq 3$. **(d)** $1\Delta y_1 + 0\Delta y_2 \geq 0$,

$-1\Delta y_1 + 1\Delta y_2 \leq 0$. **(e)** feasible by substitution.
 $\lambda = 1$.

3-14 (a) $(4, 7) \cdot (2, 0) > 0$, $(4, 7) \cdot (-2, 4) > 0$
(b) $\Delta \mathbf{z}^{(1)}$ for $\lambda = 2$ to $\mathbf{z}^{(1)} = (4, 0)$, $\Delta \mathbf{z}^{(2)}$ for
 $\lambda = 3/4$ to $\mathbf{z}^{(2)} = (5/2, 3)$, $\Delta \mathbf{z}^{(1)}$ for $\lambda = 1/4$ to
 $\mathbf{z}^{(3)} = (3, 3)$

3-16 (a) $(3, 1, 0) + \lambda(-3, 3, 9)$, $\lambda \in [0, 1]$;
 $\lambda = 1/3$ for $\mathbf{z}^{(3)}$; no λ gives $\mathbf{z}^{(4)}$

3-17 (a) not convex, $\mathbf{x}^{(1)} = (0, 3)$, $\mathbf{x}^{(2)} = (3, 0)$
(c) convex **(e)** not convex, $\mathbf{x}^{(1)} = (0, 0, 0, 4)$,
 $\mathbf{x}^{(2)} = (0, 0, 0, 5)$

3-18 (a) $\min w_4 + w_5$, s.t. $40w_1 + 30w_2 + 10w_3 + w_4 = 150$, $w_1 - w_2 \leq 0$,
 $4w_2 + w_3 + w_5 \geq 10$, $w_1, w_2, w_3, w_4, w_5 \geq 0$;
 $w_4 = 150$, $w_5 = 10$ **(c)** $\min w_3 + w_4 + w_5$,
s.t. $(w_1 - 3)^2 + (w_2 - 3)^2 - w_3 \leq 4$,
 $2w_1 + 2w_2 + w_4 = 5$, $w_1 + w_5 \geq 3$, w_3, w_4 ,
 $w_5 \geq 0$; $w_3 = 14$, $w_4 = 5$, $w_5 = 3$

3-20 (a) stop and conclude model is infeasible
(c) proceed with Phase II from initial solution
 $\mathbf{y} = (1, 3, 1)$

3-21 (a) $\max 22w_1 - w_2 + 15w_3 - M(w_4 + w_5)$,
s.t. $40w_1 + 30w_2 + 10w_3 + w_4 = 150$,
 $w_1 - w_2 \leq 0$, $4w_2 + w_3 + w_5 \geq 10$, w_1, w_2, w_3 ,
 $w_4, w_5 \geq 0$; $w_4 = 150$, $w_5 = 10$ **(c)** $\min 2w_1 + 3w_2 + M(w_3 + w_4 + w_5)$, s.t. $(w_1 - 3)^2 + (w_2 - 3)^2 - w_3 \leq 4$,
 $2w_1 + 2w_2 + w_4 = 5$,
 $w_1 + w_5 \geq 3$, $w_3, w_4, w_5 \geq 0$; $w_3 = 14$, $w_4 = 5$,
 $w_5 = 3$

3-22 (a) stop and conclude model is infeasible if M is big enough, else increase M and repeat
(c) stop and conclude $\mathbf{y} = (1, 3, 1)$ is a local optimum for the original model

Chapter 4

4-1 (a) $x_j \triangleq$ cases shipped to region j , max
 $1.60x_1 + 1.40x_2 + 1.90x_3 + 1.20x_4$,
s.t. $\sum_{j=1}^4 x_j = 1200$, $310 \leq x_1 \leq 434$,
 $245 \leq x_2 \leq 343$, $255 \leq x_3 \leq 357$,
 $190 \leq x_4 \leq 266$ **(b)** $x_1^* = \text{NE} = 408$,
 $x_2^* = \text{SE} = 245$, $x_3^* = \text{MW} = 357$,
 $x_4^* = \text{W} = 190$

4-2 (b) nonzero values are $x_{1,1}^* = 70$,
 $x_{1,2}^* = 10$, $x_{2,2}^* = 40$, $x_{2,3}^* = 5$, $x_{2,4}^* = 35$, $x_{3,3}^* = 80$

4-3 (a) $x_j \triangleq$ fraction of ingredient j ;
 $\min 200x_1 + 150x_2 + 100x_3 + 75x_4$, s.t.
 $\sum_{j=1}^4 x_j = 1$, $60x_1 + 80x_2 + 55x_3 + 40x_4 \geq 60$,
 $50x_1 + 70x_2 + 40x_3 + 100x_4 \leq 60$,

$90x_1 + 30x_2 + 60x_3 + 80x_4 \geq 60$ **(b)** the
3 main inequalities **(c)** $x_1^* = \text{oats} = .157$,
 $x_2^* = \text{corn} = .271$, $x_3^* = \text{alfalfa} = .401$,
 $x_4^* = \text{hulls} = .171$

4-4 (c) $x_1^* = .176$, $x_2^* = .353$, $x_3^* = .000$,
 $x_4^* = .471$

4-5 (a) $45 \sum_{i=1}^m x_{i,j} \leq \sum_{i=1}^m a_{i,11} x_{i,j} \leq 48 \sum_{i=1}^m x_{i,j}$,
 $j = 1, \dots, n$ **(c)** $\sum_{i=1}^m a_{i,15} x_{i,15} \geq 116 \sum_{i=1}^m x_{i,15}$

(e) $7x_{1,j} \leq 3 \sum_{i=2}^m x_{i,j}$, $j = 6, \dots, 11$

(g) $3 \sum_{i=3}^6 \sum_{j=1}^n x_{i,j} \geq \sum_{i=1}^m \sum_{j=1}^n x_{i,j}$

4-7 (a) $x_j \triangleq$ the number of cuts with pattern j ;
 $\min .34x_1 + .22x_2 + .27x_3$, s.t. $2x_1 + 1x_3 \geq 37$,
 $5x_2 + 3x_3 \geq 211$, $x_j \geq 0$, $j = 1, \dots, 4$
(b) $x_1^* = 0$, $x_2^* = 20$, $x_3^* = 37$

4-8 (b) $x_1^* = \text{santas} = 147.4$, $x_2^* = \text{trees} = 0.0$,
 $x_3^* = \text{houses} = 21.1$

4-9 (a) $\max 30x_1 + 45x_2$, s.t. $.30x_1 + .30x_2 + .10x_3 + .15x_4 + .50x_5 \leq 80$, $1.5x_3 + 2.5x_4 \leq 500$,
 $x_3 = 4x_1$, $x_4 = 4x_2$, $x_5 = x_1 + x_2$, $x_j \geq 0$,
 $j = 1, \dots, 5$ **(b)** the 3 equalities **(c)** $x_1^* = 27.8$,
 $x_2^* = 33.3$, $x_3^* = 111.1$, $x_4^* = 133.3$, $x_5^* = 61.1$

4-10 (c) $x_1^* = 0.0$, $x_2^* = 666.7$, $x_3^* = 0.0$,
 $x_4^* = 2000$, $x_5^* = 12,000$

4-12 (a) $\sum_{p=1}^n x_{i,p} = \sum_{p=1}^n d_{i,p} + \sum_{k=1}^m \sum_{p=1}^n a_{i,k} x_{k,p}$, $i = 1, \dots, m$

4-13 (a) $x_1 \triangleq$ number with 5 days starting Sunday, \dots , $x_7 \triangleq$ number with 5 days starting Saturday; $\min \sum_{j=1}^7 x_j$, s.t.

$x_1 + x_4 + x_5 + x_6 + x_7 \geq 8$,
 $x_1 + x_2 + x_5 + x_6 + x_7 \geq 6$,
 $x_1 + x_2 + x_3 + x_6 + x_7 \geq 6$,
 $x_1 + x_2 + x_3 + x_4 + x_7 \geq 6$,
 $x_1 + x_2 + x_3 + x_4 + x_5 \geq 6$,
 $x_2 + x_3 + x_4 + x_5 + x_6 \geq 10$,
 $x_3 + x_4 + x_5 + x_6 + x_7 \geq 10$,
 $x_j \geq 0$, $j = 1, \dots, 7$ **(b)** all main constraints
(c) $x_1^* = 0.0$, $x_2^* = .67$, $x_3^* = 2.0$, $x_4^* = 2.67$,
 $x_5^* = 2.0$, $x_6^* = 2.67$, $x_7^* = .67$

4-14 (c) $x_1^* = 5$, $x_2^* = 0$, $x_3^* = 0$, $x_4^* = 0$,
 $x_5^* = 3$, $x_6^* = 3$

4-15 (a) $x_{j,t} \triangleq$ investment in option j , year t ;
 $\max 1.05x_{1,4} + 1.12x_{2,3} + 1.21x_{3,1}$, s.t.

$10 = x_{1,1} + x_{2,1} + x_{3,1}$,
 $1.05x_{1,1} + 10 = x_{1,2} + x_{2,2}$,
 $1.05x_{1,2} + 1.12x_{2,1} + 10 = x_{1,3} + x_{2,3}$,
 $1.05x_{1,3} + 1.12x_{2,2} = x_{1,4}$,
all variables nonnegative **(b)** all main

constraints (c) 4 years (d) nonzero values are $x_{2,1}^* = x_{2,2}^* = 10, x_{2,3}^* = x_{1,4}^* = 21.2$

4-16 (a) $x_j \triangleq$ purchase in month $j, h_j \triangleq$ hold in month j . **(b)** LP optima are likely to be relatively large in value, meaning ignoring any fractions in them has little impact, but LP is much more tractable than ILP. **(c)** min $12x_1 + 14x_2 + 14x_3 + 14x_4 + 1.2h_1 + 1.2h_2 + 1.3h_3 + 1.4h_4$, subject to, $x_1 = 100 + h_1, h_1 + x_2 = 130 + h_2, h_2 + x_3 = 95 + h_3, h_3 + x_4 = 300, x_1, \dots, x_4 \geq 0, h_1, \dots, h_3 \geq 0$ **(d)** nonzeros $x_1^* = 230, x_3^* = 95, x_4^* = 300, h_1^* = 130$, value = 84,460.

4-20 (d) $x_1^* = x_3^* = x_4^* = 1200, x_2^* = 650, z_1^* = 0, z_2^* = 150, z_3^* = 1250, z_4^* = 1600$

4-21 (a) $\sum_{i=1}^m a_{i,k}x_{i,t} \leq b_k, k = 1, \dots, q, t = 1, \dots, n$ **(c)** $x_{i,1} = d_{i,1} + z_{i,1}, i = 1, \dots, m$

4-22 (a) $w_j^+ \triangleq$ over-estimation on point $j, w_j^- \triangleq$ under-estimation on point j ; min $\sum_{j=1}^4 (w_j^+ + w_j^-)$, s.t. $\beta_0 + 2\beta_1 = 1 + w_1^+ - w_1^-, \beta_0 + 3\beta_1 = 3 + w_2^+ - w_2^-, \beta_0 + 5\beta_1 = 3 + w_3^+ - w_3^-, \beta_0 + 7\beta_1 = 5 + w_4^+ - w_4^-$, all variables nonnegative **(b)** $\beta_0^* = .000, \beta_1^* = .714$, nonzero over $w_1^{+*} = .429, w_3^{+*} = .571$, nonzero under $w_2^{-*} = .857$ **(c)** .429, .857, .571, .000

4-23 (b) $x^* = 10$, nonzero runs $w_3^{+*} = 15, w_1^{-*} = 5$

4-24 (a) $z \triangleq$ largest deviation, $w_j^+ \triangleq$ over-estimation on point $j, w_j^- \triangleq$ under-estimation on point j ; min z , s.t. $\beta_0 + 2\beta_1 = 1 + w_1^+ - w_1^-, \beta_0 + 3\beta_1 = 3 + w_2^+ - w_2^-, \beta_0 + 5\beta_1 = 3 + w_3^+ - w_3^-, \beta_0 + 7\beta_1 = 5 + w_4^+ - w_4^-$, $z \geq w_j^+, j = 1, \dots, 4, z \geq w_j^-, j = 1, \dots, 4$, all variables nonnegative **(b)** $\beta_0^* = .333, \beta_1^* = .667, z^* = .667$, nonzero over $w_1^{+*} = w_3^{+*} = .667$, nonzero under $w_2^{-*} = .667$ **(c)** .667, .667, .667, .000

4-26 (a) min $\sum_{j=1}^5 c_j x_j$, s.t. $\sum_{j=1}^5 a_{i,j} x_j \geq r_i, i = 1, \dots, 7; 0 \leq x_j \leq u_j, j = 1, \dots, 5$; where the $a_{i,j}$ are the yield fractions in the table, c_j the costs, and u_j the availabilities; the r_i are the given requirements. **(b)** $x_1^* = 18.75, x_2^* = 125.00, x_3^* = 150.00, x_4^* = 650.00, x_5^* = 0.00$

4-27 (b) Nonzeroes $x_{1,1}^* = x_{1,2}^* = x_{1,3}^* = 833.3, x_{1,4}^* = 500.0, x_{2,1}^* = x_{2,2}^* = x_{2,3}^* = 10000.0, x_{2,4}^* = 6000.0, w_{3,1}^* = x_{3,2}^* = x_{3,3}^* = 833.3,$

$x_{3,4}^* = 500.0, x_{4,1}^* = x_{4,2}^* = x_{4,3}^* = 833.3, x_{4,4}^* = 500.0, x_{5,1}^* = x_{5,2}^* = x_{5,3}^* = 833.3, x_{5,4}^* = 500.0, h_{5,1}^* = 533.3, h_{5,2}^* = 166.7, h_{5,4}^* = 200.0$

4-28 (b) $s_1^* = s_2^* = 0, s_3^* = 8, s_4^* = 12, s_5^* = s_6^* = 24, s_7^* = 32, t_1^* = 12, t_2^* = t_7^* = 8, t_3^* = t_5^* = t_6^* = 16, t_4^* = 20$

4-47 (a) Stage 1 is acquiring voting machines. Stage 2 possible turnouts. Transferring machines from the warehouse for unmet demand.

(b) $q^{(s)} \triangleq$ the probability of scenario $s, v_p^{(s)} \triangleq$ voters (in hundreds) at precinct p under scenario s : min $\sum_{s=1}^3 q^{(s)} \sum_{p=1}^4 w_p^{(s)}$, subject to, $5 \sum_{p=1}^5 x_p + .5 \sum_{p=1}^4 \sum_{s=1}^3 y_p^{(s)} \leq 150, s = 1, \dots, 3, \sum_{p=1}^4 y_p^{(s)} \leq x_5, s = 1, \dots, 3, \sum_{p=1}^4 \sum_{s=1}^3 x_p + y_p^{(s)} + w_p^{(s)} = v_p^{(s)}, x_1, \dots, x_5 \geq 0, y_p^{(s)} \geq 0$ and $w_p^{(s)} \geq 0$ for $p = 1, \dots, 4, s = 1, \dots, 3$. **(c)** nonzeros $x_1^* = 5, x_2^* = 4, x_3^* = 2, x_4^* = 2, x_5^* = 15.45, y_1^{(2)*} = 1, y_2^{(2)*} = 3, y_3^{(2)*} = 4, y_3^{(3)*} = 2.45, y_4^{(2)*} = 6, y_4^{(3)*} = 13, w_1^{(3)*} = 2, w_2^{(3)*} = 4, w_3^{(3)*} = 5.55$, value = 4.62. Fulfills scenario $s = 1$ with direct machine shipments, other voter demands from warehouse inventory. Some demands go unfulfilled.

Chapter 5

5-1 (b) boundary and extreme, infeasible, interior, boundary not extreme, infeasible **(c)** $w_2 \leq 3$ active, no active, $w_2 \leq 3$ active, no active **(d)** optimal or unique, neither, neither, optimal not unique, neither **(e)** $w^{(1)}$ basic feasible with actives $w_2 \leq 3$ and $-w_1 + w_2 \leq 1$; $w^{(2)}$ basic infeasible with actives $w_1 \geq 0$ and $w_2 \leq 3$; all others not basic.

5-3 (a) $A = \begin{pmatrix} 1 & -4 & 1 & 1 & 0 \\ 9 & 0 & 6 & 0 & 0 \\ -5 & 9 & 0 & 0 & -1 \end{pmatrix}$,

$b = (12, 15, 3), c = (4, 2, -33, 0, 0)$

(c) $A = \begin{pmatrix} 2 & -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$,

$b = (0, 3, 3, 3), c = (15, 41, -11, 0, 0, 0, 0)$

(e) $A = \begin{pmatrix} 1 & -1 & 1 & 5 & 1 \\ 0 & 3 & -3 & -9 & 0 \end{pmatrix}, b = (10, -6),$

$c = (2, 1, -1, 4, 0)$

5-4 (b) $\mathbf{A} = \begin{pmatrix} -1 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 \end{pmatrix}, \mathbf{b} = (2, 10)$

(c) yes, no, yes, yes, no, no

(d) $\mathbf{y} = (2, 4, 0, 0)$ feasible,

$\mathbf{y} = (0, 0, 2, 10)$ feasible, $\mathbf{y} = (-2, 0, 0, 20)$

infeasible **(e)** $\mathbf{y} = (2, 4), \mathbf{y} = (0, 0),$

$\mathbf{y} = (-2, 0)$

5-6 (a) $5\Delta w_1 + 1\Delta w_2 - 1\Delta w_3 = 0,$

$3\Delta w_1 - 4\Delta w_2 + 8\Delta w_3 = 0, \Delta w_2 \geq 0$

5-7 (a) $\mathbf{x} = (1, 0, 3, 0)$ **(b)** $\Delta \mathbf{x} = (-3, 1, -1, 0)$

for $x_2, \Delta \mathbf{x} = (1, 0, -5, 1)$ for x_4 **(c)** each has

$\mathbf{A} \Delta \mathbf{x} = \mathbf{0}, \Delta x_2 \geq 0$ and $\Delta x_4 \geq 0$ **(d)** no, yes

(e) $\lambda = 1/3, \{x_2, x_3\}, \lambda = 3/5, \{x_1, x_4\}$

5-9 (a) $z_1^* = 4, z_2^* = 2$

(b) $\mathbf{A} = \begin{pmatrix} -2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{b} = (2, 6, 4),$

$\mathbf{c} = (3, 1, 0, 0, 0)$ **(c)** bases $\{z_3, z_4, z_5\}$, then

either $\{z_1, z_3, z_4\}, \{z_1, z_2, z_3\}$ or $\{z_2, z_4, z_5\},$

$\{z_1, z_2, z_5\}, \{z_1, z_2, z_3\}$ **(d)** $\mathbf{z}^{(0)} = (0, 0),$

then either $\mathbf{z}^{(1)} = (4, 0), \mathbf{z}^{(2)} = (4, 2)$ or

$\mathbf{z}^{(1)} = (0, 2), \mathbf{z}^{(2)} = (10/3, 8/3), \mathbf{z}^{(3)} = (4, 2)$

5-12 (a) $\mathbf{x}^{(0)} = (2, 0, 8, 5, 0)$, basic feasible.

(b)

	x_1	x_2	x_3	x_4	x_5	
min \mathbf{c}	0	1	0	1	1	\mathbf{b}
\mathbf{A}	-2	1	0	2	0	6
	0	0	0	1	1	5
	0	1	1	-1	0	3
$t = 0$	B	N	B	B	N	
$\mathbf{x}^{(0)}$	2	0	8	5	0	5
$\Delta \mathbf{x}, x_2$.5	1	-1	0	0	$\bar{c}_2 = 1$
$\Delta \mathbf{x}, x_5$	1	0	-1	-1	1	$\bar{c}_5 = 0$

$\mathbf{x}^{(0)}$ is optimal because all nonbasic $\bar{c}_j \geq 0$ in a min problem.

5-15 (a) yes **(c)** no **(e)** no

5-16 (a) $z = 10 - 29x_2 + 10x_3,$

$x_1 = 1 - (3x_2 - 1x_4), x_2 = 3 - (1x_2 + 5x_4)$

5-17 (a) no **(c)** yes

5-18 (b) $\mathbf{A} = \begin{pmatrix} -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}, \mathbf{b} = (4, 10),$

$\mathbf{c} = (4, 5, 0, 0)$ **(c)** $\mathbf{y}^{(0)} = (0, 0, 4, 10),$

then either $\mathbf{y}^{(1)} = (10, 0, 14, 0)$ and

$\Delta \mathbf{y} = (1, 1, 0, 0)$ for y_2 improves without limit,

or $\mathbf{y}^{(1)} = (0, 4, 0, 14)$ and $\Delta \mathbf{y} = (1, 1, 0, 0)$ for

y_1 improves without limit

5-20 (a) min $w_6 + w_7$, s.t. $w_1 + w_2 + w_4 = 18,$

$-2w_1 + w_3 - w_6 = -2, 3w_2 + 5w_3 - w_5 +$

$w_7 = 15, w_1, \dots, w_7 \geq 0; \{w_4, w_6, w_7\}$

5-21 (a) max $2w_1 + w_2 + 9w_3 - M(w_6 + w_7)$

s.t. same constraints and starting basis.

5-22 (b) $\mathbf{A} = \begin{pmatrix} -2 & 1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}, \mathbf{b} = (2, 1),$

min $\mathbf{c} = (0, 0, 0, 0, 1)$ **(c)** $\mathbf{y}^{(0)} = (0, 0, 0, 1, 2),$

$\mathbf{y}^{(1)} = (0, 1, 0, 0, 1)$, optimal value positive

5-24 (a) $(4!)/(2!2!) = 6$

(c) $(2340!)/(1150!1190!)$

5-25 (a) yes **(c)** no

5-26 (b) $\mathbf{A} = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix},$

$\mathbf{b} = (2, 10, 4)$, **(c)** yes, no, yes, yes, no, no

(d) $\{y_1, y_2, y_3\}, \{y_1, y_2, y_4\}$, and $\{y_1, y_2, y_5\}$

(e) Some basic components = 0. From basis

$\{y_1, y_2, y_3\}$, introducing y_4 gives direction

$\Delta \mathbf{y} = (-1/5, 0, -1/5, 1, 0)$, which will make $\lambda = 0.$

5-28 (a) alternative optima $\mathbf{x} = (3, 6)$ through

$\mathbf{x} = (6, 3)$ **(b)** $\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ -2 & 1 & 0 & 1 & 0 \\ 1 & -2 & 0 & 0 & 1 \end{pmatrix},$

$\mathbf{b} = (9, 0, 0)$, $\mathbf{c} = (1, 1, 0, 0, 0)$ **(c)** bases

$\{x_3, x_4, x_5\}$, then either $\{x_1, x_3, x_4\}$ and

$\{x_1, x_2, x_4\}$, or $\{x_2, x_3, x_5\}$ and $\{x_1, x_2, x_5\}$

(d) $\mathbf{x}^{(0)} = \mathbf{x}^{(1)} = (0, 0)$, then either

$\mathbf{x}^{(2)} = (3, 6)$ or $\mathbf{x}^{(2)} = (6, 3)$

5-30 (a) $\mathbf{B}^{-1} = \begin{pmatrix} 0 & .500 \\ .250 & .125 \end{pmatrix}$

(b) $v = (0, .5)$ **(c)** no for x_2 , yes for x_4

(d) $E \begin{pmatrix} 1 & .2 \\ 0 & .2 \end{pmatrix}, \mathbf{B}^{-1} \begin{pmatrix} .050 & .525 \\ .050 & .025 \end{pmatrix}$

5-32 (a)

	x_1	x_2	x_3	x_4	x_5	
min c	5	4	3	2	16	b
A	2	0	1	0	6	8
	0	1	1	2	3	12
$t = 0$	1st	2nd	N	N	N	
$\mathbf{x}^{(0)}$	4	12	0	0	0	68
	$\mathbf{B}^{-1} = \begin{pmatrix} 0.5 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{v} = \begin{pmatrix} 2.5 \\ 4 \end{pmatrix}$					
$\bar{\mathbf{c}}$	0	0	-3.5	-6	$\boxed{-11}$	
$\Delta \mathbf{x}, x_5$	-3	-3	0	0	1	
	$\boxed{\frac{4}{3}}$	$\frac{12}{3}$	-	-	-	$\lambda = 4/3$
$t = 1$	N	2nd	N	N	1st	
$\mathbf{x}^{(1)}$	0	9	0	0	4/3	57 1/3

(b) (c) See above. (d) New $\mathbf{B} = \begin{pmatrix} 6 & 0 \\ 3 & 1 \end{pmatrix}$,

$\mathbf{E} = \begin{pmatrix} .33 & 0 \\ -1 & 1 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} 1/6 & 0 \\ -1/2 & 1 \end{pmatrix}$ and

$\mathbf{v} = (2/3, 4)$

5-33 (a)

	x_1	x_2	x_3	x_4	x_5	
min c	0	1	0	1	1	b
A	-2	1	0	2	0	6
	0	0	0	1	1	5
	0	1	1	-1	0	3
$t = 0$	1st	N	2nd	3rd	N	
$\mathbf{x}^{(0)}$	2	0	8	5	0	5
	$\mathbf{B}^{-1} = \begin{pmatrix} -1/2 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \mathbf{v} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$					
$\bar{\mathbf{c}}$	0	1	0	0	0	

$\mathbf{x}^{(0)}$ is optimal because all nonbasic $\bar{c}_j \geq 0$ in a min problem.

5-34 (a) nonimproving, $\lambda = 2, (L, B, B, L, B)$

(c) improving, $\lambda = 5, (L, B, U, B, B)$

5-35 (a) $z_1^* = 2, z_2^* = 1$

(b) $\mathbf{A} = \begin{pmatrix} 1 & 1 & -1 & 0 \\ 3 & 2 & 0 & -1 \end{pmatrix}, \mathbf{b} = (3, 8),$

$\mathbf{c} = (5, 6, 0, 0)$ (c) basis statuses $(U, U, 1st, 2nd)$, then either $(L, U, 1st, 2nd), (L, 2nd, 1st, L)$ and $(1st, 2nd, L, L)$, or $(U, L, 1st, 2nd), (1st, L, L, 2nd)$ and $(1st, 2nd, L, L)$ (d) $\mathbf{z}^{(0)} = (6, 5)$, then either

$\mathbf{z}^{(1)} = (0, 5), \mathbf{z}^{(2)} = (0, 4)$ and $\mathbf{z}^{(3)} = (2, 1)$, or $\mathbf{z}^{(1)} = (6, 0), \mathbf{z}^{(2)} = (3, 0)$ and $\mathbf{z}^{(3)} = (2, 1)$

Chapter 6

6-1 (b) \$ profit, thousand beta zappers, thousand freeze phasers, extrusion hours, trimming hours, assembly hours (c) make thousand beta zappers, make thousand freeze phasers (d) input 1: 5 hours extrusion, 1 hour trimming, 12 hours assembly; output 1: thousand beta zappers, \$2500 profit; input 2: 9 hours extrusion, 2 hours trimming, 15 hours assembly; output 2: thousand freeze phasers, \$1600 profit

6-3 (b) \$ cost, professional-equivalent hours production, Proof hours supervision, grad maximum hours (c) ugrad hour programming, grad hour programming, professional hour programming (d) input 1: .2 hours Proof supervision, \$4 cost; output 1: .2 professional-equivalent hours programming; input 2: .3 hours Proof supervision, 1 hour grad maximum, \$10 cost; output 2: .3 professional-equivalent hours programming; input 3: .05 hours Proof supervision, \$25 cost; output 3: 1 professional-equivalent hour programming

6-5 (a) tighten, decrease, more steep (c) relax, decrease, less steep (e) relax, increase, less steep (g) tighten, increase, more steep

6-6 (a) tighten, decrease (c) tighten, increase

6-7 (a) increase, more steep (c) increase, less steep

6-8 (a) $v_1 \triangleq$ \$ change in optimal profit per thousand increase in beta zapper demand, $v_2 \triangleq$ \$ change in optimal profit per thousand increase in freeze phaser demand, $v_3 \triangleq$ \$ change in optimal profit per hour increase in extrusion capacity, $v_4 \triangleq$ \$ change in optimal profit per hour increase in trimming capacity, $v_5 \triangleq$ \$ change in optimal profit per hour increase in assembly capacity (b) $v_1, v_2 \leq 0; v_3, v_4, v_5 \geq 0$; RHS increase tightens \geq and relaxes \leq (c) $v_1 + 5v_3 + v_4 + 12v_5 \geq 2500, v_2 + 9v_3 + 2v_4 + 15v_5 \geq 1600$, implicit cost of activities should equal or exceed objective function return (d) min $10v_1 + 15v_2 + 320v_3 + 300v_4 + 480v_5$, minimize total implicit cost (f) $x_1 = 10$ or $v_1 = 0, x_2 = 15$ or $v_2 = 0, 5x_1 + 9x_2 = 320$ or $v_3 = 0, x_1 + 2x_2 = 300$ or $v_4 = 0, 12x_1 + 15x_2 = 480$ or $v_5 = 0$, either a primal constraint is active or small RHS change as no objective function impact

(g) $v_1 + 5v_3 + v_4 + 12v_5 = 2500$ or $x_1 = 0$, $v_2 + 9v_3 + 2v_4 + 15v_5 = 1600$ or $x_2 = 0$, a primal variable should be used only if its implicit cost matches its objective function return

6-10 (a) $v_1 \triangleq$ \$ change in optimal cost per professional-equivalent hour increase in required production, $v_2 \triangleq$ \$ change in optimal cost per hour increase in Proof supervision, $v_3 \triangleq$ \$ change in optimal cost per hour increase in grad availability **(b)** $v_1 \geq 0$; $v_2, v_3 \leq 0$; RHS increase tightens \geq and relaxes \leq

(c) $.2v_1 + .2v_2 \leq 4$, $.3v_1 + .1v_2 + v_3 \leq 10$, $v_1 + .05v_2 \leq 25$, implicit value of activities should not exceed objective function cost

(d) $\max 1000v_1 + 164v_2 + 500v_3$, maximize total implicit value **(f)** $.2x_1 + .3x_2 + x_3 = 1000$ or $v_1 = 0$, $.2x_1 + .1x_2 + .05x_3 = 164$ or $v_2 = 0$, $x_2 = 500$ or $v_3 = 0$, either a primal constraint is active or small RHS change as no objective function impact **(g)** $.2v_1 + .2v_2 = 4$ or $x_1 = 0$, $.3v_1 + .1v_2 + v_3 = 10$ or $x_2 = 0$, $v_1 + .05v_2 = 25$ or $x_3 = 0$, a primal variable should be used only if its implicit value matches its objective function cost

6-12 (a) $\max 40v_1 + 10v_2$, s.t. $2v_1 + 4v_2 \leq 17$, $3v_1 + 4v_2 \leq 29$, $2v_1 + 3v_3 \leq 0$, $3v_1 + v_2 - v_3 \leq 1$, $v_1 \leq 0$, $v_2 \geq 0$, v_3 URS

(c) $\min 10v_1 + 19v_2 + 5v_3$, s.t. $2v_1 + v_3 \geq 30$, $-3v_1 + 4v_2 + v_3 = 0$, $-v_2 + v_3 \leq -2$, $9v_1 = 10$, $v_1 \geq 0$, $v_2 \leq 0$, v_3 URS **(e)** $\min 10v_1 + 11v_3$, s.t. $3v_1 + v_2 = 0$, $2v_1 + v_3 \geq 2$, $-v_1 + 3v_3 \geq 9$, $-v_2 + v_3 = 0$, $v_1 \leq 0$, $v_2 \geq 0$, v_3 URS **(g)** $\max 40v_1 + 18v_2 + 11v_3$, s.t. $15v_1 + 12v_2 \leq 0$, $15v_1 - 90v_2 \leq 32$, $15v_1 \leq 50$, $14v_2 + v_3 \leq 0$, $v_1 \leq -19$, v_1 URS, $v_2 \geq 0$, $v_3 \leq 0$

6-13 (a) $2x_1 + 3x_2 + 2x_3 + 3x_4 = 40$ or $v_1 = 0$, $4x_1 + 4x_2 + x_4 = 10$ or $v_2 = 0$, $2v_1 + 4v_2 = 17$ or $x_1 = 0$, $3v_1 + 4v_2 = 29$ or $x_2 = 0$, $2v_1 + 3v_2 = 0$ or $x_3 = 0$, $3v_1 + v_2 - v_3 = 1$ or $x_4 = 0$ **(c)** $2x_1 - 3x_2 + 9x_4 = 10$ or $v_1 = 0$, $4x_2 - x_3 = 19$ or $v_2 = 0$, $2v_1 + v_3 = 30$ or $x_1 = 0$, $-v_2 + v_3 = -2$ or $x_3 = 0$

(e) $3w + 2x_1 - x_2 = 10$ or $v_1 = 0$, $w - y = 0$ or $v_2 = 0$, $2v_1 + v_3 = 2$ or $x_1 = 0$, $-v_1 + 3v_3 = 9$ or $x_2 = 0$ **(g)** $12x_1 - 90x_2 + 14x_4 = 18$ or $v_2 = 0$, $x_4 = 11$ or $v_3 = 0$, $15v_1 + 12v_2 = 0$ or $x_1 = 0$, $15v_1 - 90v_2 = 32$ or $x_2 = 0$, $15v_1 = 50$ or $x_3 = 0$, $14v_2 + v_3 = 0$ or $x_4 = 0$, $v_1 = -19$ or $x_5 = 0$

6-14 (a) $\min 14v_1 + 14v_2$, s.t. $2v_1 + 5v_2 \geq 14$, $5v_1 + 2v_2 \geq 7$, $v_1, v_2 \geq 0$; $\mathbf{x}^* = (2, 2)$, $\mathbf{v}^* = (1/3, 8/3)$ **(c)** $\max 24v_1 + 11v_2$, s.t. $2v_1 + 3v_2 \leq 8$, $9v_1 + v_2 \leq 11$, $v_1, v_2 \geq 0$; $\mathbf{x}^* = (3, 2)$, $\mathbf{v}^* = (1, 2)$

6-15 (a) $v_1 = 3$, $v_2 = 1/3$ **(c)** $v_1 = 1/3$, $v_2 = 4$

6-16 (a) $\min 4v_1 + 12v_2$, s.t. $2v_1 \geq 4$, $v_1 + 3v_2 \geq 1$, $v_1 \leq 0$, $v_2 \geq 0$; unbounded, infeasible **(c)** $\max 2v_1 + 5v_2$, s.t. $v_1 \geq 10$, $v_1 - v_2 \geq 3$, $v_1, v_2 \geq 0$, infeasible, unbounded **(e)** $\max 2v_1 + 5v_2$, s.t. $-v_1 + v_2 \leq -3$, $2v_1 - 2v_2 \leq 4$, $v_1, v_2 \geq 0$; infeasible, infeasible

6-17 (a) dual optimal value ≤ 70

6-18 (a) For min primal, $v_1 \leq 0 \triangleq$ dual on \leq row 1, $v_2 \geq 0 \triangleq$ dual on \geq row 2, v_3 URS \triangleq dual on $=$ row 3. First dual \leq constraint corresponds to primal $x_1 \geq 0$, second to $x_2 \geq 0$. Checking dual feasibility for $\mathbf{v} = (0, 0, 2)$, $15 \cdot 2 \leq 30$, $-4 \cdot 2 = -8 < 2$. Dual value $10 \cdot 2 = 20$ lower bounds primal optimal value.

6-22 (a) no; 51.25 **(b)** \$0; \$208.33 **(c)** increase \$20,833; increase at least \$39,375 and at most \$41,667 **(d)** increase \$31,875; decrease at least \$25,925 and at most \$31,875 **(e)** 87.5 hours **(f)** \$624.99

6-24 (a) $v_1^* = \$25.882$ **(b)** \$1294.10, at least \$2415.64 and at most \$2588.20 **(c)** yes, increase at least \$258.81; increase at least \$376.45 **(d)** \$2.235 **(e)** \$4917.65; at least \$17,376.01 and at most \$19,670.58 **(f)** yes **(g)** no **(h)** \$20.12

6-29 (a) $\min 19v_1 + 10v_2$, subject to $-3v_1 + 4v_2 + 6v_3 \geq 0$, $2v_2 = 13$, $1v_1 + 7v_2 + 8v_3 \geq -8$, $v_1 \geq 0$, v_2 URS, $v_3 \leq 0$. **(b)** $\bar{\mathbf{z}}$ must satisfy all constraints of the given primal, $\bar{\mathbf{v}}$ must satisfy all constraint of the dual in part (a), and complementary slackness conditions for all main inequalities must be satisfied, i.e.

$$(19 + 3\bar{z}_1 - \bar{z}_3)\bar{v}_1 = 0, (0 - 6\bar{z}_1 - 8\bar{z}_3)\bar{v}_3 = 0, (0 + 3\bar{v}_1 - 4\bar{v}_2 - 6\bar{v}_3)\bar{z}_1 = 0, (-8 - 1\bar{v}_1 - 7\bar{v}_2 - 8\bar{v}_3)\bar{z}_3 = 0$$

6-31 (a) $\mathbf{B}^{-1} = \begin{pmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{pmatrix}$, $\mathbf{N} = \begin{pmatrix} 2 & 0 & 4 \\ 0 & 2 & 1 \end{pmatrix}$, $\mathbf{c}^B = (5, -10)$, $\mathbf{c}^N = (0, 0, 0)$, $\mathbf{b} = (2, 8)$

(b) $\bar{\mathbf{x}}^B = \begin{pmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 2 \\ 8 \end{pmatrix} = (5, 3)$,

$\bar{\mathbf{x}}^N = (0, 0, 0)$, value $= -5$, feasible $5 - 3 = 2$, $5 + 3 = 8$, all ≥ 0 . **(c)** $\min (2, 8) \cdot \mathbf{v}$,

subject to $\mathbf{v} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \geq (5, -10)$,

$\mathbf{v} \begin{pmatrix} 2 & 0 & 4 \\ 0 & 2 & 1 \end{pmatrix} \geq (0, 0, 0)$, \mathbf{v} URS.

(d) $\bar{\mathbf{v}} = (5, -10) \begin{pmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{pmatrix} = (15/12, -5/2)$

value = -5 same as primal. (e) Part (d) assures matching solution values, but not complementary slackness needed for optimality.

6-34 (a) min $2x_1 + 3x_2 + 4x_3$, subject to,

$x_1 + 2x_2 + x_3 - x_4 = 3$,

$2x_1 - x_2 + 3x_3 - x_5 = 4, x_1, \dots, x_5 \geq 0$.

(b) max $3v_1 + 2v_2$, subject to, $v_1 + 2v_2 \leq 2$,

$2v_1 - v_2 \leq 3, v_1 + 3v_2 \leq 4, v_1 \geq 0, v_2 \geq 0$.

(c) $x_1 = x_2 = x_3 = 0, x_4 = -3, x_5 = -4$.

Infeasible because some components negative.

(d) All $c_j \geq 0$, implies feasible (b). Only on x_4 and x_5 and $\bar{c}_4 = \bar{c}_5 = 0$. (e) $t = 0$:

$\mathbf{B} = \mathbf{B}^{-1} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \mathbf{x}^B = (-3, -4)$,

$\bar{\mathbf{c}} = (2, 3, 4, 0, 0), t = 1: \mathbf{r} = (1, 0)$,

$\Delta \bar{\mathbf{c}} = (-1, -2, -1, 1, 0), p = 2, \lambda = 3/2$,

$\mathbf{v} = (3/2, 0), \bar{\mathbf{c}} = (1/2, 0, 5/2, 3/2, 0)$,

$\mathbf{B} = \begin{pmatrix} 2 & 0 \\ -1 & -1 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} 1/2 & 0 \\ -1/2 & -1 \end{pmatrix}$,

$\mathbf{x}^B = (3/2, -11/2). t = 2: \mathbf{r} = (1/2, 1)$,

$\Delta \bar{\mathbf{c}} = (-5/2, 0, -7/22, 1/2, 0), p = 1$,

$\lambda = 1/5, \mathbf{v} = (8/5, 1/5), \mathbf{B} = \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix}$,

$\mathbf{B}^{-1} = \begin{pmatrix} 1/5 & 2/5 \\ 2/5 & -1/5 \end{pmatrix}, \mathbf{x}^B = (11/5, 2/5)$.

Stop primal feasible, value 28/5.

6-35 (a) max $6v_1 + 3v_2$, subject to, $2v_1 + v_2 \leq 3$,

$-v_1 + v_2 \leq 4, v_1 + 2v_2 \leq 6, 6v_1 + v_2 \leq 7$,

$-5v_1 + 2v_2 \leq 1, v_1 \geq 0, v_2 \geq 0$.

(b) All $c_j \geq 0$ implies feasible in (a).

(c) Restricted primals min $q_1 + q_2$, subject to

$2x_1 - x_2 + x_3 + 6x_4 - 5x_5 - x_6 + q_1 = 6$,

$x_1 + x_2 + 2x_3 + x_4 + 2x_5 - x_7 + q_2 = 3$,

all variables $\geq 0. t = 0: \mathbf{v}^{(0)} = (0, 0)$,

$\bar{\mathbf{c}} = (3, 4, 6, 7, 1, 0, 0), \mathbf{B} = \mathbf{B}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,

$\mathbf{q}^B = (6, 3), \bar{\mathbf{w}} = (1, 1)$, restricted optimal.

$\lambda = 1 \mathbf{v}^{(1)} = (1, 1), \Delta \mathbf{x} = (3, 0, 0, 0, 0, 0, 0)$,

$\Delta \mathbf{q} = (-2, -1). x_1$ enters, q_2 leaves.

$t = 1: \mathbf{B} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}, \mathbf{B}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}$,

$\bar{\mathbf{c}} = (0, 4, 3, 0, 4, 1, 1), \bar{\mathbf{x}} = (3, 0, 0, 0, 0, 0, 0)$,
 $\bar{\mathbf{q}} = (0, 0)$, primal feasible and optimal,
 value = 9.

Chapter 7

7-1 (a) $w_1^* = 3/2, w_2^* = 2$ (b) $\Delta \mathbf{w} = (2, 3)$

(d) check constraints strictly satisfied

(e) $\lambda_{max} = 5/17$ (f) $\mathbf{w}^{(1)} = (27/17, 32/17)$

7-3 (a) no (c) no (e) yes

7-4 (a) no (c) yes (e) no

7-5 (a) $2\Delta w_1 + 3\Delta w_2 - 3\Delta w_3 = 0$,

$4\Delta w_1 - 1\Delta w_2 + 1\Delta w_3 = 0$

7-6 (a) $\Delta \mathbf{x} = (-.8, -1.6, 4)$,

$\begin{pmatrix} 1 & 2 & 1 \\ -2 & 1 & 0 \end{pmatrix} \Delta \mathbf{x} = \mathbf{0}$

7-7 (a) $\Delta \mathbf{z} = (-14, -3, -5)$

(b) $\mathbf{P} = \begin{pmatrix} 1/6 & -1/6 & 1/3 \\ -1/6 & 1/6 & -1/3 \\ 1/3 & -1/3 & 2/3 \end{pmatrix}$

(c) $\Delta \mathbf{z} = (-7/2, 7/2, -7)$

(d) $\begin{pmatrix} 2 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix} \Delta \mathbf{z} = \mathbf{0}$,

$(14, 3, 5) \cdot (-7/2, 7/2, -7) < 0$

7-9 (a) $(1/2, 1/5, 1, 1/9)$ (c) $(3/2, 1, 1, 2/3)$

7-10 (a) $(2, 5, 1, 9)$ (c) $(6, 25, 1, 54)$

7-11 (a) $\mathbf{x}^* = (0, 0, 4)$ (b) $\mathbf{y}^{(3)} = (1, 1, 1)$,

$\mathbf{y}^* = (0, 0, 4)$ (c) min $4y_1 + 3y_2 + 5y_3$,

s.t. $4y_1 + 5y_2 + 3y_3 = 12, y_1, y_2, y_3 \geq 0$

7-13 (a) $\Delta \mathbf{x} = (-.64, 1.6, -2.24)$,

$\Delta \mathbf{y} = (-.32, 1.6, -2.24)$ (b) $(2, 3, 5) \cdot \Delta \mathbf{x} < 0$,

$(2, 5, 3) \cdot \Delta \mathbf{x} = 0$ (c) $\lambda = .36084$

(d) $\mathbf{x}^{(4)} = (1.769, 1.577, 0.192)$,

$\mathbf{y}^{(4)} = (0.884, 1.577, 0.192)$

7-15 (a) check feasible and strictly positive

(b) min $40y_1 + 3y_2$, s.t. $4y_1 - 3y_2 + 2y_3 = 3$,

$3y_2 - y_3 = 2, y_1, y_2, y_3 \geq 0$

(c) $\Delta \mathbf{x} = (-7.669, 7.669, 7.669)$

(d) $(10, 1, 0) \cdot \Delta \mathbf{x} < 0, \begin{pmatrix} 1 & -1 & 2 \\ 0 & 1 & -1 \end{pmatrix} \Delta \mathbf{x} = \mathbf{0}$

(e) $\lambda = .12037, \mathbf{x}^{(1)} = (3.077, 3.923, 1.923)$

7-17 (a) yes, unbounded

(c) no, $\mathbf{x}^{(12)} = (3.894, 0.106, 9)$

7-19 (a) $\mathbf{w}^* = (4, 0, 0)$ (b) max $13w_1 - 2w_2$

+ $w_3 + \mu(\ln(w_1) + \ln(w_2) + \ln(w_3))$,

s.t. $3w_1 + 6w_2 + 4w_3 = 12, w_1, w_2, w_3 \geq 0$

(c) 17.7 and 16.45, 3.088 and -78.10 , moderate bonus in middle vs. major penalty near boundary (d) $\mathbf{w}^* = (1.497, 0.619, 0.949)$, $\mathbf{w}^* = (2.799, 0.285, 0.474)$, $\mathbf{w}^* = (3.850, 0.350, 0.600)$

7-21 (a) yes (c) no

7-22 (a) curve II

7-23 (a) check feasible and strictly positive

(b) $\min 4x_1 - x_2 + 2x_3 - 10(\ln(x_1) + \ln(x_2) + \ln(x_3))$, s.t. $4x_1 - 3x_2 + 2x_3 = 13$, $3x_2 - x_3 = 1$, $x_1, x_2, x_3 \geq 0$

(c) $\Delta \mathbf{x} = (-4.6415, 6.1887, 18.566)$

(d) $(0.667, -11, -3) \cdot \Delta \mathbf{x} < 0$,

$\begin{pmatrix} 4 & -3 & 2 \\ 0 & 3 & -1 \end{pmatrix} \Delta \mathbf{x} = \mathbf{0}$ (e) $\lambda_{max} = .64634$

(f) decrease then increase

7-25 (a) check feasible and strictly positive

(b) $\Delta \mathbf{x} = (-16.899, 16.899, 16.899)$

(c) $(10, 1, 0) \cdot \Delta \mathbf{x} < 0$, $\begin{pmatrix} 1 & -1 & 2 \\ 0 & 1 & -1 \end{pmatrix} \Delta \mathbf{x} = \mathbf{0}$

(d) $\lambda_{max} = .2367$ (e) $\lambda = .1$,

$\mathbf{x}^{(1)} = (2.310, 4.690, 2.690)$

7-27 (a) $\max 8v_1 + 12v_2$, subject to,

$2v_1 + w_1 = 5$, $v_2 + w_2 = 4$, $v_1 + v_2 + w_3 = 3$,

$2v_2 + w_4 = 2$, $6v_1 + 3v_2 + w_5 = 16$,

$w_1, \dots, w_5 \geq 0$. (b) $\mathbf{w} = (4, 7/2, 2, 1, 23/2)$.

Check that all are strictly positive and satisfy all main constraints. (c) $x_j \cdot w_j = 0$,

$j = 1, \dots, 5$. (d) primal = 39.33, dual = 10,

complementary slackness = 29.33.

(e) $\mathbf{A}\Delta \mathbf{x} = \mathbf{0}$, $\mathbf{A}^T \Delta \mathbf{v} + \Delta \mathbf{w} = \mathbf{0}$,

$2\Delta w_1 + 4\Delta x_1 = -3$, $3\Delta w_2 + 3.5\Delta x_2 = -5.5$,

$2\Delta w_3 + 2\Delta x_3 = 1$, $3\Delta w_4 + 1\Delta x_4 = 2$,

$0.33\Delta w_5 + 11.5\Delta x_5 = 1.17$. (f) $t = 1$:

$\mathbf{x}^{(1)} \leftarrow (1.2811, 0.0030, 2.6470, 3.9773, 0.4651)$,

primal value = 29.7547, $\mathbf{v}^{(1)} \leftarrow (1.4794, 0.5315)$,

$\mathbf{w}^{(1)} \leftarrow (2.0411, 3.4685, 0.9891, 0.9370, 5.5289)$,

dual value = 18.2133. $t = 2$:

$\mathbf{x}^{(2)} \leftarrow (0.9309, 1.8314, 3.1943, 2.7512, 0.4906)$,

primal value = 34.9151, $\mathbf{v}^{(2)} \leftarrow (1.7927, 0.9995)$,

$\mathbf{w}^{(2)} \leftarrow (1.4146, 3.0005, 0.2078, 0.0009, 2.2451)$,

dual value = 26.3361. $t = 3$:

$\mathbf{x}^{(3)} \leftarrow (0.9590, 0.0018, 3.2666, 3.6619, 0.4692)$,

primal value = 29.4332, $\mathbf{v}^{(3)} \leftarrow (1.8455, 0.7283)$,

$\mathbf{w}^{(3)} \leftarrow (1.3089, 3.2717, 0.4262, 0.5434, 2.7419)$,

dual value = 23.5036.

7-29 (a) (LP) is a problem or model form with size and symbolic parameters unspecified. An instance, like the given one, is a specific case with parameter values and size made explicit.

(b) $\$4/\$11\$5\$8\$16/0\$ - \$1\$2\$1/2\$1\$0\$3$

/1\$7/ Length = no. chars used = 35.

(c) The number of digits of integer value $\pm q$ is $\pm \lceil \log(|q|) + 1 \rceil$, not its magnitude. (d) Here $n = 4$ and $m = 2$. The count of the $n c_j, m b_i$, and $n \cdot m a_{i,j}$ grows in proportion to $n + m + nm$.

Lengths of coefficients values $\pm q$ grow in proportion to $\pm \lceil \log(|q|) + 1 \rceil$

7-31 (a)

	x_1	x_2	x_3	x_4	x_5	x_6	
max c	0	1	0	0	0	0	b
s.t.	+1		-1				1/4
	+1			+1			1
	+1/4	-1			+1		0
	+1/4	+1				+1	1
	B	B	N	B	N	B	

(b) Basic as shown, yield $\mathbf{x}^{(1)} = (1/4, 1/16, 0, 3/4, 0, 7/8)$ value = $1/16$. (c) Introducing nonbasic slack x_3 gives feasible $\Delta \mathbf{x}^{(1)} = (1, 1/4, 1, -1, 0, -1/2)$ with improving reduced cost $1/4$. With $\lambda = 3/4$, $\mathbf{x}^{(2)} = (1, 1/4, 3/4, 0, 0, 1/2)$

Chapter 8

8-1 (b) (166, 85.2) to (0, 184.8) (c) first: $\mathbf{x}^* = (24, 166, 50)$; second: $\mathbf{x}^* = (24, 0, 216)$

8-3 (b) (2.4, 19.8) to (2.8, 12.5) to (3.2, 12.1) to (4.2, 11.3) (c) first: $\mathbf{x}^* = (0, 1, 0, 0.1)$; second: $\mathbf{x}^* = (1, 0, 1, 1, 0)$

8-4 (b) approximately (9.05, 4202) to (12, 2760) to (20, 1809) to (30, 1365) to (43.29, 1110) (c) first: $\mathbf{x}^* = (6.65, 5.56, 4.46, 3.33)$; second: $\mathbf{x}^* = (1, 1, 1, 17)$

8-5 (a) first: $\mathbf{x}^* = (0, 6)$, second: $\mathbf{x}^* = (4, 1)$ (b) no, no, yes, yes, no, yes (c) (9, 4) to (27, 2) to (30, 0)

8-7 (a) first: first objective, s.t. original constraints, points (1, 9/2) through (3, 3/2) alternative optima; second: second objective, s.t. original constraints and $6x_1 + 4x_2 \geq 26$, $\mathbf{x}^* = (1, 9/2)$

(b) first: second objective, s.t. original constraints, $\mathbf{x}^* = (0, 5)$; second: first objective, s.t. original constraints and $x_2 \geq 5$, $\mathbf{x}^* = (0, 5)$

8-9 (a) $\min 15x_1 + 110x_2 + 92x_3 + 123x_4$

8-10 (a) $\mathbf{x}^* = (1, 9/2)$ (b) $\mathbf{x}^* = (0, 5)$

8-12 (a) $\min d_1 + d_2$, s.t. $3x_1 + 5x_2 - x_3 - d_1 \leq 20$, $11x_2 + 23x_3 + d_2 \geq 100$, $d_1, d_2 \geq 0$, and all original constraints

(c) $\min d_1 + d_2 + d_3$, s.t. $40x_1 + 23x_2 + d_1 \geq 700$, $20x_1 - 20x_2 - d_2 \leq 25$, $5x_2 + x_3 - d_3 \leq 65$,

$d_1, d_2, d_3 \geq 0$, and all original constraints

(e) $\min d_1 + d_2$, s.t. $22x_1 + 8x_2 + 13x_3 - d_1 \leq 20$, $3x_1 + 6x_2 + 4x_3 + d_2 \geq 12$, $d_1, d_2 \geq 0$, and all original constraints

8-13 (b) $\min d_1 + d_2$, s.t. $x_1 + d_1 \geq 3$, $2x_1 + 2x_2 + d_2 \geq 14$, $2x_1 + x_2 \leq 9$, $x_1 \leq 4$, $x_2 \leq 7$, $x_1, x_2, d_1, d_2 \geq 0$ **(c)** least total distance to the two contours

8-15 (a) $\min d_1$, s.t. $x_1 + d_1 \geq 3$, $2x_1 + 2x_2 + d_2 \geq 14$, $2x_1 + x_2 \leq 9$, $x_1 \leq 4$, $x_2 \leq 7$, $x_1, x_2, d_1, d_2 \geq 0$; any feasible solution with $x_1 \geq 3$ is alternative optimal; $\min d_2$, s.t. $d_1 \leq 0$ and other constraints of the first LP; $\mathbf{x}^* = (3, 3)$ **(b)** $\min 100d_1 + d_2$, s.t. same constraints as first LP in (a) **(c)** yes; yes

8-17 (a) first: first objective s.t. original constraints, $\mathbf{x}^* = (24, 166, 50)$; second: second objective s.t. original constraints and $x_2 \geq 166$, $\mathbf{x}^* = (24, 166, 50)$ **(b)** $\max .50x_1 + 2.20x_2 + .80x_3$, s.t. original constraints, $\mathbf{x}^* = (24, 166, 50)$

(c) $\min d_1 + d_2$, s.t. $x_2 + d_1 \geq 100$, $.50x_1 + .20x_2 + .80x_3 + d_2 \geq 144$, $d_1, d_2 \geq 0$, and all original constraints **(e)** $\mathbf{x}^* = (24, 100, 116)$, in between the earlier **(f)** first: $\min d_1$, s.t. constraints of (c), any feasible solution with $x_2 \geq 100$ is optimal; second: $\min d_2$, s.t. $d_1 \leq 0$ and constraints of (c), $\mathbf{x}^* = (24, 100, 116)$

(g) $\min 10000d_1 + d_2$

8-19 (a) first: first objective s.t. original constraints, $\mathbf{x}^* = (0, 1, 0, 0, 1)$; second: second objective s.t. original constraints and $1.0x_1 + 0.4x_2 + 1.4x_3 + 1.8x_4 + 2.0x_5 \leq 2.4$, $\mathbf{x}^* = (0, 1, 0, 0, 1)$ **(b)** $\min 4.5x_1 + 6.2x_2 + 7.4x_3 + 7.8x_4 + 18.4x_5$, s.t. original constraints, $\mathbf{x}^* = (1, 1, 1, 0, 0)$ **(c)** $\min d_1 + d_2$, s.t. $1.0x_1 + 0.4x_2 + 1.4x_3 + 1.8x_4 + 2.0x_5 - d_1 \leq 3.0$, $2.5x_1 + 5.4x_2 + 4.6x_3 + 4.2x_4 + 14.4x_5 - d_2 \leq 15$, $d_1, d_2 \geq 0$, and all original constraints **(d)** part (c) optimal value $\neq 0$ **(e)** $\mathbf{x}^* = (0, 1, 1, 1, 0)$, new intermediate **(f)** first: $\min d_1$, s.t. constraints of (c), $\mathbf{x}^* = (1, 0, 0, 0, 1)$; second: $\min d_2$, s.t. $d_1 \leq 0$ and constraints of (c), same \mathbf{x}^* **(g)** $\min 10000d_1 + d_2$

8-21 (a) $x_1 \triangleq$ singles, $x_2 \triangleq$ doubles, $x_3 \triangleq$ luxuries; $\min 2d_1^- + d_2^-$, s.t. $40x_1 + 60x_2 + 120x_3 \leq 10000$; $.7x_1 + .4x_2 + .9x_3 + d_1^- \geq 100$; $.3x_1 + .6x_2 + .1x_3 + d_2^- \geq 120$; all variables nonnegative

(b) $x_1^* = 76.92, x_2^* = 115.38, x_3^* = 0.00$, $d_1^{*-} = 0.00, d_2^{*-} = 27.69$

8-22 (b) nonzeros: $x_1^* = 4.4, x_2^* = 10.0$, $x_3^* = 12.3, x_4^* = 16.5, r_1^* = 0.6, r_4^* = 1.5$

8-28 (b) $v^* = 285, f^* = .007, d^* = .040$, $g_1^{+*} = g_1^{-*} = 0, g_2^* = .637$

8-32 (c) $x_1^* = 53.33, x_2^* = 500, x_3^* = 166.67$, $y_1^* = 100, y_2^* = 1150, y_3 = 250$, all $d_k^* = 0$ except $d_1^* = 466.67$

Chapter 9

9-1 (a) nodes: 1,2,3,4,5; arcs: (3,2), (3,4), (3,5); edges: (1,2), (1,3), (1,4), (2,5), (4,5) **(b)** yes, no, yes, no **(c)** replace each edge with 2 opposed arcs of the same length as the edge

9-3 (a) 1 - 3 - 5 - 2, length 6, 1 - 3, length 3, 1 - 4, length 7, 1 - 3 - 5, 4 **(b)** 1-3 best for 1 to 3, 1-3-5 best for 1 to 5, 3-5 best for 3 to 5, 3-5-2 best for 3 to 2, 5-2 best for 5 to 2 **(c)** $\nu[1] = 0$, $\nu[2] = 6, x_{1,3}[2] = x_{3,5}[2] = x_{2,5}[2] = 1$, $\nu[3] = 3, x_{1,3}[3] = 1, \nu[4] = 7, x_{1,4}[4] = 1$, $\nu[5] = 4, x_{1,3}[5] = x_{3,5}[5] = 1$ **(d)** $\nu[1] = 0$, $\nu[2] = \min\{\nu[1] + 10, \nu[3] + 5, \nu[5] + 2\}$, $\nu[3] = \min\{\nu[1] + 3\}$, $\nu[4] = \min\{\nu[1] + 7, \nu[3] + 9, \nu[5] + 4\}$, $\nu[5] = \min\{\nu[2] + 2, \nu[3] + 1, \nu[4] + 4\}$ **(f)** positive lengths preclude negative cycles

9-5 (a) 1 - 3 - 2, length 4, 1 - 3, length 3, 1 - 3 - 2 - 4, length 10, 2 - 1, length 8, 2 - 4 - 3, length 10, 2 - 4, length 6, 3 - 1, length 3, 3 - 2, length 1, 3 - 2 - 4, length 7, 4 - 3 - 1, length 7, 4 - 3 - 2, length 5, 4 - 3 length 3 **(b)** 1-3 best for 1 to 3, 1-3-2 best for 1 to 2, 3-2 best for 3 to 2, 3-2-4 best for 3 to 4, 2-4 best for 2 to 4 **(c)** $\nu[k][k] = 0, k = 1, \dots, 4$, $\nu[1][2] = 4, x_{1,3}[1][2] = x_{3,2}[1][2] = 1$, $\nu[1][3] = 3, x_{1,3}[1][3] = 1, \nu[1][4] = 10$, $x_{1,3}[1][4] = x_{3,2}[1][4] = x_{2,4}[1][4] = 1$, $\nu[2][1] = 8, x_{1,2}[2][1] = 1, \nu[2][3] = 10$, $x_{2,4}[2][3] = x_{4,3}[2][3] = 1, \nu[2][4] = 6$, $x_{2,4}[2][4] = 1, \nu[3][1] = 3, x_{1,3}[3][1] = 1$, $\nu[3][2] = 1, x_{3,2}[3][2] = 1, \nu[3][4] = 7$, $x_{3,2}[3][4] = x_{2,4}[3][4] = 1, \nu[4][1] = 7$, $x_{4,3}[4][1] = x_{1,3}[4][1] = 1, \nu[4][2] = 5$, $x_{4,3}[4][2] = x_{3,2}[4][2] = 1, \nu[4][3] = 3$, $x_{4,3}[4][3] = 1$ **(d)** $\nu[k][k] = 0, k = 1, \dots, 4$, $\nu[1][2] = \min\{8, \nu[1][3] + \nu[3][2], \nu[1][4] + \nu[4][1]\}$, $\nu[1][3] = \min\{3, \nu[1][2] + \nu[2][3], \nu[1][4] + \nu[4][2]\}$,

$$\begin{aligned} \nu[1][4] &= \min \{ \nu[1][2] + \nu[2][4], \nu[1][3] + \nu[3][4] \}, \\ \nu[2][1] &= \min \{ 8, \nu[2][3] + \nu[3][1], \nu[2][4] + \nu[4][1] \}, \\ \nu[2][3] &= \min \{ \nu[2][1] + \nu[1][3], \nu[2][4] + \nu[4][3] \}, \\ \nu[2][4] &= \min \{ 6, \nu[2][1] + \nu[1][4], \nu[2][3] + \nu[3][4] \}, \\ \nu[3][1] &= \min \{ 3, \nu[3][2] + \nu[2][1], \nu[3][4] + \nu[4][1] \}, \\ \nu[3][2] &= \min \{ 1, \nu[3][1] + \nu[1][2], \nu[3][4] + \nu[4][2] \}, \\ \nu[3][4] &= \min \{ \nu[3][1] + \nu[1][4], \nu[3][2] + \nu[2][4] \}, \\ \nu[4][1] &= \min \{ \nu[4][2] + \nu[2][1], \nu[4][3] + \nu[3][1] \}, \\ \nu[4][2] &= \min \{ 6, \nu[4][1] + \nu[1][2], \nu[4][3] + \nu[3][2] \}, \\ \nu[4][3] &= \min \{ 4, \nu[4][1] + \nu[1][3], \nu[4][2] + \nu[2][3] \} \end{aligned}$$

(f) positive lengths preclude negative dicycles

9-7 (a) 1 - 2, length -5, 1 - 3, length 10, 1 - 3 - 4, length 12 **(b)** 1 - 2 - 4 - 1, 1 - 3 - 4 - 1 **(c)** no, yes **(d)** functional equations no longer sufficient

9-9 (a) one to all, no negative dicycles **(b)** see 9-3(a) **(c)** see 9-3(a) **(d)** 1 - 3 - 2, length 8, 1 - 3, length 3, 1 - 4, length 7, 1 - 3 - 5, length 4 **(e)** 4, labels correct after $t = 4$

9-11 (a) 1 - 3 - 4 - 1

9-12 (a) all to all, no negative dicycles **(b)** see 9-5(a) **(c)** see 9-5(a) **(d)** 1 - 2, length 8, 1 - 3, length 3, 1 - 2 - 4, length 14, 2 - 1, length 8, 2 - 1 - 3, length 11, 2 - 4, length 6, 3 - 1, length 3, 3 - 2, length 1, 3 - 2 - 4, length 7, 4 - 2 - 1, length 14, 4 - 2, length 6, 4 - 3 length 3

9-14 (a) 1 - 3 - 4 - 1

9-15 (a) one to all, lengths nonnegative **(b)** see 9-3(a) **(c)** see 9-3(a) **(d)** 1 - 3 - 2, length 8, 1 - 3, length 3, 1 - 4, length 7, 1 - 3 - 5, length 4

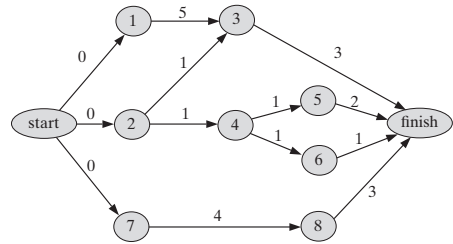
9-17 (a) Bellman-Ford, Floyd-Warshall, Dijkstra (best), 1 - 3 - 2, length 5, 1 - 3, length 2, no path to 4 **(c)** none apply

9-18 (a) acyclic; one numbering: $a = 1, b = 2, c = 4, d = 3, e = 6, f = 5$ **(c)** not acyclic; dicycle a-d-e-c-b-a

9-19 (a) acyclic digraph **(b)** $\nu[1] = 0, \nu[2] = \infty, \nu[3] = 2, \nu[4] = 12, \nu[5] = -1, \nu[6] = 11$

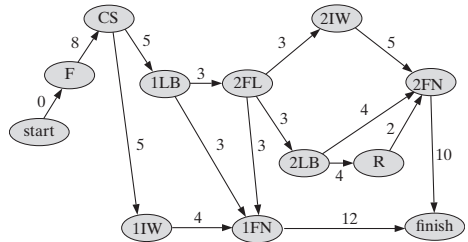
(c) 1 - 3, 1 - 3 - 4, 1 - 3 - 5, 1 - 3 - 5 - 6

9-23 (a)



(b) check arcs (i, j) have $i < j$ **(c)** $\nu[1] = 0, \nu[2] = 0, \nu[3] = 5, \nu[4] = 1, \nu[5] = 2, \nu[6] = 2, \nu[7] = 0, \nu[8] = 4, finish = 8$ **(d)** $start - 1 - 3 - finish$ **(e)** late starts: 1 = 2, 2 = 6, 3 = 7, 4 = 7, 5 = 8, 6 = 9, 7 = 3, 8 = 7; slacks: 1 = 2, 2 = 6, 3 = 2, 4 = 6, 5 = 6, 6 = 7, 7 = 3, 8 = 3

9-26 (a)



(b) one numbering: $F = 1, CS = 2, 1LB = 3, 1IW = 4, 2FL = 5, 1FN = 6, 2LB = 7, 2IW = 8, R = 9, 2FN = 10$ **(c)** $\nu[F] = 0, \nu[CS] = 8, \nu[1LB] = 13, \nu[1IW] = 13, \nu[2FL] = 16, \nu[1FN] = 19, \nu[2LB] = 19, \nu[2IW] = 19, \nu[R] = 23, \nu[2FN] = 25, finish = 35$ **(d)** $start - F - CS - 1LB - 2FL - 2LB - R - 2FN - finish$ **(e)** late starts: $F = 0, CS = 8, 1LB = 13, 1IW = 19, 2FL = 16, 1FN = 23, 2LB = 19, 2IW = 20, R = 23, 2FN = 25$; slacks: $F = 0, CS = 0, 1LB = 0, 1IW = 6, 2FL = 0, 1FN = 4, 2LB = 0, 2IW = 1, R = 0, 2FN = 0$

9-28 (c) for 8: 1-2-13-3-8, length 26; for 10: 1-2-13-3-8-10, length 41; for 11: 1-2-13-4-11, length 19; for 12: 1-2-13-3-8-6-12, length 35

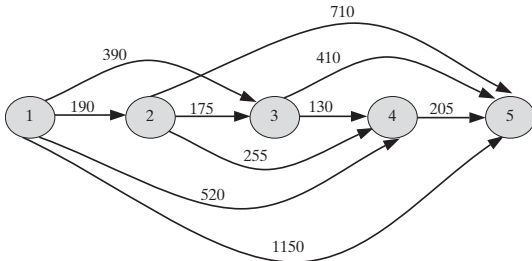
9-29 (c) 1-2, length 6; 1-2-7-3, length 42; 1-2-7-4, length 30; 1-2-7-5, length 20; 1-6, length 11; 1-2-7, length 12; 1-2-7-5-8, length 34; 1-2-7-5-8-9, 57; 1-2-7-5-8-10, length 59

9-30 (b) yes **(d)** 7-11-2 with one 4-hour and one 3-hour, cost \$90

9-31 (d) 1-3-6-7 (or 1-3-6-5-7), length 9 gridsize

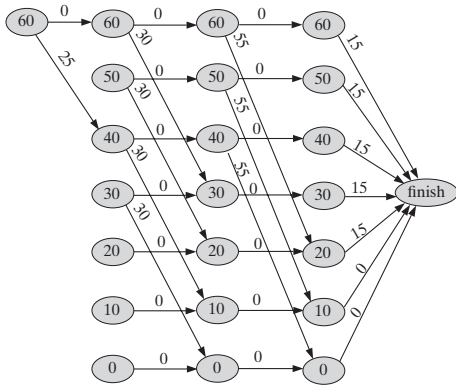
9-32 (d) Using S for shed and I for the unlabeled intersection: S-5-I-1, length 24; S-5-3-2, length 26; S-5-3, length 19; S-5-4, length 25; S-5, length 5

9-33 (b)



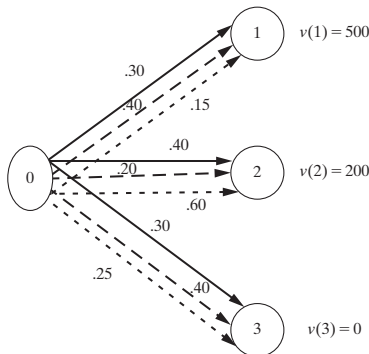
(d) produce 30, 35, 0, 35 **(e)** produce 30, 25

9-35 (b)



(d) take 1 and 3

9-39 (a) Actions $K = \{ \text{all Alpha, all Betas, Split} \}$, rewards all = 0. **(b)** Using solid lines for X14Alpha, dashed lines for X14Beta, and dotted lines for Split



(c) Boundary states $v(1) = 500, v(2) = 200, v(3) = 0. v(0) = \max\{.30 \cdot 500 + .40 \cdot 200,$

$$.40 \cdot 500 + .20 \cdot 200, .15 \cdot 500 + .60 \cdot 200\} = 240$$

(d) Action 2 = All Beta is optimal.

Chapter 10

10-1 (a) $V = \{1, 2, 3, 4, 5\}, A = \{(1, 2), (1, 4), (2, 5), (3, 1), (3, 4), (3, 5), (4, 2), (4, 5), (5, 2)\}$

(b) source: 1, 3; sink: 2, 5; transshipment: 4

(d) $\min 5x_{1,2} + 10x_{1,4} - 6x_{3,1} + 2x_{3,4} + 8x_{4,2} + 6x_{4,5} + 1x_{5,2} + 3x_{5,3}, \text{ s.t.}$

$$-x_{1,2} - x_{1,4} + x_{3,1} = -50,$$

$$x_{1,2} - x_{2,5} + x_{4,2} + x_{5,2} = 20,$$

$$-x_{3,1} - x_{3,4} + x_{5,3} = -70,$$

$$x_{1,4} + x_{3,4} - x_{4,2} - x_{4,5} = 0,$$

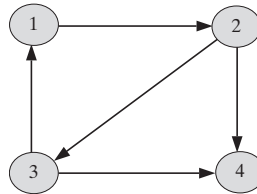
$$x_{2,5} + x_{4,5} - x_{5,2} - x_{5,3} = 100, x_{1,4} \leq 40,$$

$$x_{2,5} \leq 10, x_{3,1} \leq 4, x_{4,2} \leq 20, \text{ all } x_{i,j} \geq 0$$

(e)

$$A = \begin{pmatrix} -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

10-3 (a) check every column has at most a -1 and a +1 **(b)**



10-5 (a) $150 > 80$ **(b)** node 4, demand 70; zero cost arcs (1,4) and (2,4)

10-7 (a) $\min 25x_{P1,C1} + 30x_{P1,C2} + 15x_{P1,W} + 45x_{P2,C1} + 23x_{P2,C2} + 15x_{P2,W} + 11x_{W,C1} + 14x_{W,C2}, \text{ s.t. } x_{P1,C1} + x_{P1,C2} + x_{P1,W} \leq 400,$

$$x_{P2,C1} + x_{P2,C2} + x_{P2,W} \leq 600,$$

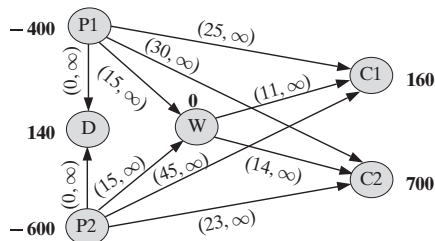
$$x_{P1,W} + x_{P2,W} - x_{W,C1} - x_{W,C2} = 0,$$

$$x_{P1,C1} + x_{P2,C1} + x_{W,C1} = 160,$$

$$x_{P1,C2} + x_{P2,C2} + x_{W,C2} = 700, \text{ all } x_{i,j} \geq 0$$

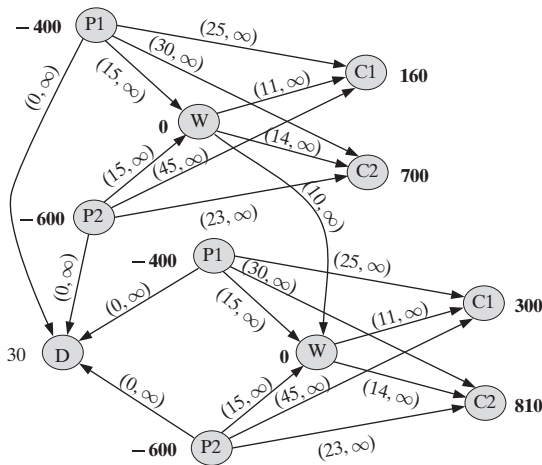
(b) $x_{P1,C1}^* = 160, x_{P1,W}^* = 100, x_{P2,C2}^* = 600,$

$x_{W,C2}^* = 100$ **(c)**



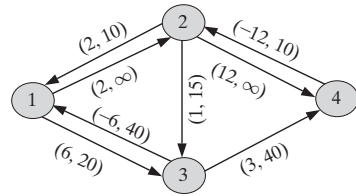
source: P1,P2; sink: C1,C2,D; transshipment: W

10-9 (a) $\min 25x_{P1,1,C1,1} + 30x_{P1,1,C2,1} + 15x_{P1,1,W,1} + 45x_{P2,1,C1,1} + 23x_{P2,1,C2,1} + 15x_{P2,1,W,1} + 11x_{W,1,C1,1} + 14x_{W,1,C2,1} + 25x_{P1,2,C1,2} + 30x_{P1,2,C2,2} + 15x_{P1,2,W,2} + 45x_{P2,2,C1,2} + 23x_{P2,2,C2,2} + 15x_{P2,2,W,2} + 11x_{W,2,C1,2} + 14x_{W,2,C2,2} + 10x_{W,1,W,2}$,
 s.t. $x_{P1,1,C1,1} + x_{P1,1,C2,1} + x_{P1,1,W,1} \leq 400$,
 $x_{P2,1,C1,1} + x_{P2,1,C2,1} + x_{P2,1,W,1} \leq 600$,
 $x_{P1,1,W,1} + x_{P2,1,W,1} - x_{W,1,C1,1} - x_{W,1,C2,1} - x_{W,1,W,2} = 0$, $x_{P1,1,C1,1} + x_{P2,1,C1,1} + x_{W,1,C1,1} = 150$,
 $x_{P1,1,C2,1} + x_{P2,1,C2,1} + x_{W,1,C2,1} = 700$,
 $x_{P1,2,C1,2} + x_{P1,2,C2,2} + x_{P1,2,W,2} \leq 400$,
 $x_{P2,2,C1,2} + x_{P2,2,C2,2} + x_{P2,2,W,2} \leq 600$,
 $x_{P1,2,W,2} + x_{P2,2,W,2} - x_{W,2,C1,2} - x_{W,2,C2,2} + x_{W,1,W,2} = 0$,
 $x_{P1,2,C1,2} + x_{P2,2,C1,2} + x_{W,2,C1,2} = 300$,
 $x_{P1,2,C2,2} + x_{P2,2,C2,2} + x_{W,2,C2,2} = 810$, all
 $x_{i,k,j,l} \geq 0$ **(b)** $x_{P1,1,C1,1}^* = 160$, $x_{P1,1,W,1}^* = 210$,
 $x_{P2,1,C2,1}^* = 600$, $x_{W,1,C2,1}^* = 100$, $x_{P1,2,C1,2}^* = 300$,
 $x_{P1,2,W,2}^* = 100$, $x_{P2,2,C2,2}^* = 600$, $x_{W,2,C2,2}^* = 210$,
 $x_{W,1,W,2}^* = 110$ **(c)**



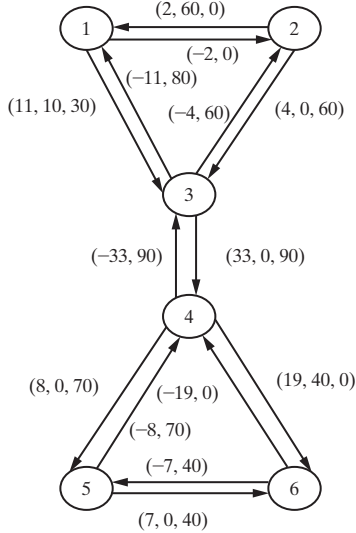
10-11 (a) cycle **(b)** chain, path **(c)** chain **(d)** cycle, dicycle
10-13 (a) check flow balance and capacities
(b) $\Delta x = \pm(1, 0, 1, -1, 0), \pm(0, 1, 0, 1, -1), \pm(1, 1, 1, 0, -1)$ **(d)** no, yes, yes, no, no, no
(e) no, yes, yes, no, no, yes **(f)** 20, 20, 10
10-15 (a) $x^* = (0, 35, 60, 40, 0)$
10-17 (a) add node 0; (1,0) flow 50, (3,0) flow 70, (0,2) flow 20, (0,5) flow 100

10-18 (a) check flow balance and capacities **(b)**



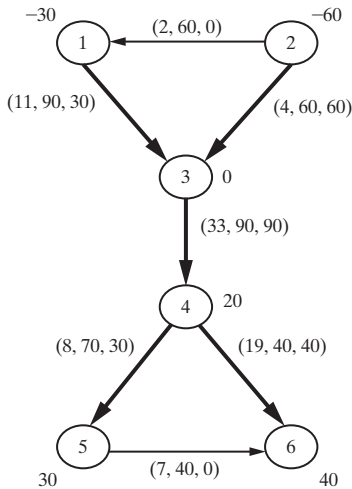
(c) direction of 1-2-3-1 or 2-3-4-2 **(d)** 15 or 10

10-20 (a) Check flows between 0 and capacity, and flow balance at all nodes. **(b)** Showing forward and reverse arcs of the residual digraph with costs and limits, plus current flow on forward arcs gives negative dicycle 1-2-3-1, cost -9, maximum step $\lambda = 50$ in first residual. Then negative dicycle 4-5-6-4, cost -4, maximum step $\lambda = 28$ in the second leaving

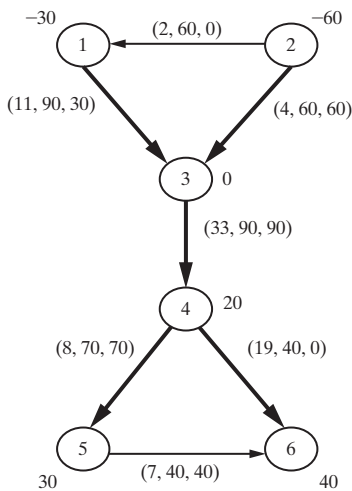


No negative dicycles. Last flow optimal.
10-22 (a) apply weights +1, -1 **(c)** apply weights +1, -1, -1
10-23 (a) $x_{1,3} = 50, x_{1,4} = 10, x_{2,3} = 25, x_{2,5} = 5, x_{3,5} = 65, x_{4,5} = 10$; feasible **(c)** $x_{1,2} = 25, x_{1,4} = 35, x_{2,3} = 55, x_{3,5} = 45, x_{4,5} = 35$; infeasible **(e)** not basis, cycle 1-3-2-1 **(g)** not basis, not connected

10-24 (a) Bold arcs below. Basis because spanning tree.



(b) Yields the basic flow shown above. Feasible because within bounds and balances. **(c)** Degenerate because $x_{2,3} = 60 = u_{2,3}$, and $x_{3,4} = 90 = u_{3,4}$ **(d)** Nonbasic $x_{5,6}$ produces cycle direction 5-6-4-5, net cost -4 , step $\lambda = 40$, updated basis and flows below



Now optimal because neither nonbasic produces an improving direction.

10-25 (a) forms spanning tree, all off-tree at bounds **(b)** increase (2, 3), $\Delta \mathbf{x} = (1, -1, 1, 0, 0)$; increase (3, 4), $\Delta \mathbf{x} = (-1, 1, 0, -1, 1)$ **(c)** yes, yes **(d)** 15, 10

10-27 (a) $x_{2,4}^* = 35, x_{3,1}^* = 60, x_{3,2}^* = 40$
(c) $x_{1,2}^* = 15, x_{1,3}^* = 35, x_{2,3}^* = 15, x_{3,4}^* = 10$

10-28 (a) feasible, spanning tree, all off-tree at bounds **(b)** 2-3-4-2 **(c)** 4-2-1-3-4

10-29 (a) yes **(c)** no

10-30 (a) For $V \triangleq \{1, 2, 3, 4\}$,
 $A \triangleq \{(1, 2), (1, 3), (2, 4), (3, 2), (3, 4)\}$

$$\mathbf{A} = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\ +1 & 0 & -1 & +1 & 0 \\ 0 & +1 & 0 & -1 & -1 \\ 0 & 0 & +1 & 0 & +1 \end{pmatrix}$$

(b) First 3 cols **(c)** Delete last row gives **B**

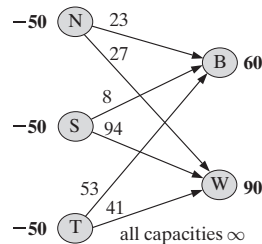
with $\det \mathbf{B} = -1$. **(d)** $\det \begin{pmatrix} -1 & -1 \\ 1 & 0 \end{pmatrix} = +1$,

$$\det \begin{pmatrix} 0 & +1 \\ 0 & 0 \end{pmatrix} = 0, \det \begin{pmatrix} -1 & -1 & 0 \\ +1 & 0 & -1 \\ 0 & +1 & 0 \end{pmatrix} = -1,$$

$$\det \begin{pmatrix} -1 & +1 & 0 \\ 0 & -1 & -1 \\ +1 & 0 & -1 \end{pmatrix} = 0$$

$\det(\text{first 4 columns}) = \det(\text{last 4}) = 0$

10-31 (a) $\min 23x_{N,B} + 77x_{N,W} + 8x_{S,B} + 94x_{S,W} + 53x_{T,B} + 41x_{T,W}$, s.t. $x_{N,B} + x_{N,W} = 50$,
 $x_{S,B} + x_{S,W} = 50, x_{T,B} + x_{T,W} = 50$,
 $x_{N,B} + x_{S,B} + x_{T,B} = 60$,
 $x_{N,W} + x_{S,W} + x_{T,W} = 90$, all $x_{i,j} \geq 0$
(b) $x_{N,B}^* = 10, x_{N,W}^* = 40, x_{S,B}^* = 50, x_{T,W}^* = 50$
(c)



10-34 (a) $\max \sum_{i=1}^4 \sum_{j=1}^4 r_{i,j} x_{i,j}$,
s.t. $\sum_{j=1}^4 x_{i,j} = 1, i = 1, \dots, 4, \sum_{i=1}^4 x_{i,j} = 1$,
 $j = 1, \dots, 4$, all $x_{i,j} = 0$ or 1, where $r_{i,j} \triangleq$ the rating of member i for task j

(b) $x_{1,1}^* = x_{2,4}^* = x_{3,3}^* = x_{4,2}^* = 1$ **(c)** 4 member nodes with supply 1, 4 task nodes with demand 1, arcs from every member to every task with cost the negative of the corresponding rating **(d)** network flow with integer supplies and demands

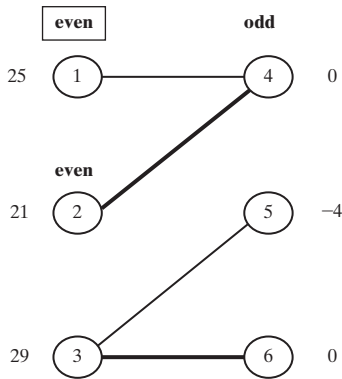
10-36 (a) Define values given in table as $c_{i,j}$.

Then $\max \sum_{i=1}^3 \sum_{j=1}^6 c_{i,j} x_{i,j}$ subject to

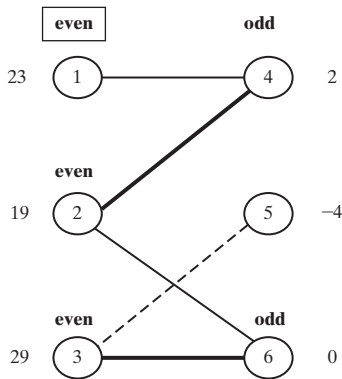
$$\sum_{j=4}^6 x_{i,j} = 1 \quad i = 1, \dots, 3,$$

$$\sum_{i=1}^3 x_{i,j} = 1 \quad j = 4, \dots, 6 \text{ all } x_{i,j} \text{ binary}$$

(b)

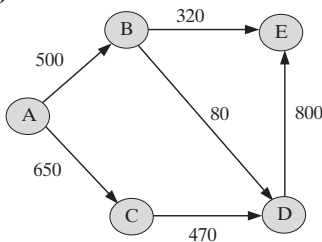


(duals on nodes) (c) Starting assignment shown above in bold; complementary because both edges have $\bar{c}_{i,j} = 0$. (d) Initial even-odd-even labeling also shown above. Now blocked. Using $D = \{(2, 5), (2, 6)\}$, update duals with $\lambda = 2$ to produce



Now further labeling of 6 odd and 3 even gives opportunity for solution growth. Adjusting along augmenting path $1-4-2-6-3-5$ yields optimal solution. Nonzeros $x_{1,4}^* = x_{2,6}^* = x_{2,5}^* = 1$, value 69. (e) Duals as shown above. Value also 69. Complementary because all chosen (i, j) have $\bar{c}_{i,j} = 0$. (f) With $|I| = 3$ bound is $(3)^3 = 27$ steps. Here less with only one dual change and one solution growth.

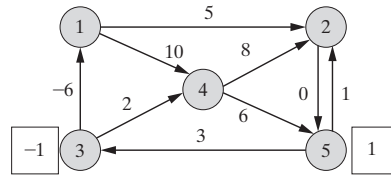
10-38 (a)



(b) $x_{A,B}^* = 400, x_{A,C}^* = 470, x_{B,D}^* = 80, x_{B,E}^* = 320, x_{C,D}^* = 470, x_{D,E}^* = 550$ (c) add infinite capacity, cost = -1 arc from E to A; all other cost = 0; all net demands = 0

10-40 (a) add infinite capacity, cost -1 return arc (2,3), all other costs and all net demands 0; $\mathbf{x}^* = (30, 0, 30, 100, 0)$ (c) add infinite capacity, cost -1 return arc (4,1), all other costs and all net demands 0; $\mathbf{x}^* = (50, 40, 0, 50, 40)$

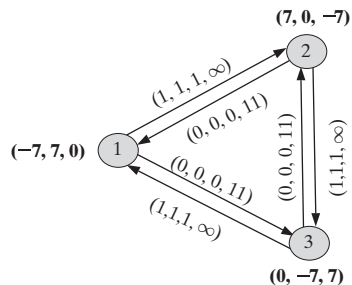
10-45 (a)



Solution must be a dipath from $s = 3$ to $t = 5$ of minimum total length.

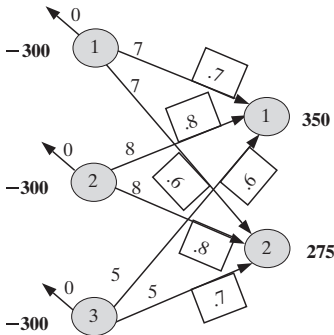
10-46 (a) $\min \sum_{k=1}^3 (x_{1,3,k} + x_{2,1,k} + x_{3,2,k})$,
 s.t. $x_{3,1,1} + x_{2,1,1} - x_{1,2,1} - x_{1,3,1} = -7$,
 $x_{1,2,1} + x_{3,2,1} - x_{2,3,1} - x_{2,1,1} = 0$,
 $x_{2,3,1} + x_{1,3,1} - x_{3,1,1} - x_{3,2,1} = 7$,
 $x_{3,1,2} + x_{2,1,2} - x_{1,2,2} - x_{1,3,2} = 7$,
 $x_{1,2,2} + x_{3,2,2} - x_{2,3,2} - x_{2,1,2} = -7$,
 $x_{2,3,2} + x_{1,3,2} - x_{3,1,2} - x_{3,2,2} = 0$,
 $x_{3,1,3} + x_{2,1,3} - x_{1,2,3} - x_{1,3,3} = 0$,
 $x_{1,2,3} + x_{3,2,3} - x_{2,3,3} - x_{2,1,3} = 7$,
 $x_{2,3,3} + x_{1,3,3} - x_{3,1,3} - x_{3,2,3} = -7$,
 $\sum_{k=1}^3 x_{1,2,k} \leq 11, \sum_{k=1}^3 x_{2,3,k} \leq 11$,
 $\sum_{k=1}^3 x_{3,1,k} \leq 11, \text{ all } x_{i,j,k} \geq 0$

(b) $x_{1,2,1}^* = x_{1,2,3}^* = x_{2,3,1}^* = x_{2,3,2}^* = x_{3,1,2}^* = x_{3,1,3}^* = 5.5, x_{1,3,1}^* = x_{2,1,2}^* = x_{3,2,3}^* = 1.5$
 (c)



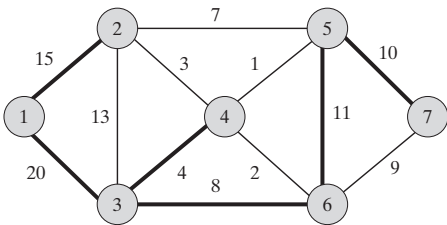
(d) an optimal solution would ship each 7 units to itself at 0 cost (e) integer flows not guaranteed in multicommodity flows

10-48 (a) $\min 7(x_{1,1} + x_{1,2}) + 8(x_{2,1} + x_{2,2}) + 5(x_{3,1} + x_{3,2}), \text{s.t. } \sum_{j=1}^2 x_{i,j} \leq 300, i = 1, 2, .7x_{1,1} + .8x_{2,1} + .6x_{3,1} = 350, .6x_{1,2} + .8x_{2,2} + .7x_{3,2} = 275, \text{all } x_{i,j} \geq 0$
(b) $x_{1,1}^* = 300, x_{2,1}^* = 175, x_{2,2}^* = 81.25, x_{3,2}^* = 300$ **(c)**

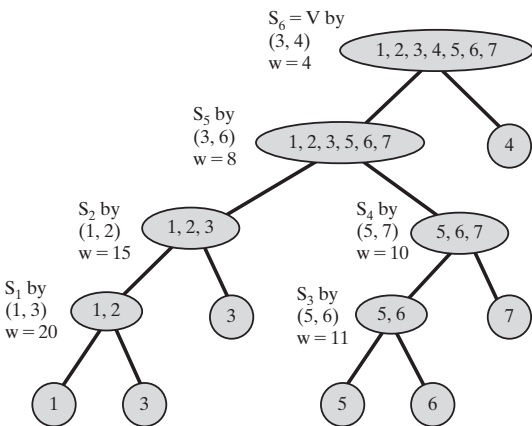


(d) integer flows not guaranteed in flow with losses

10-50 (a) Tree shown in bold with total weight 68



(b)



(c) $\sum_{i,j \in S_k} x_{i,j} \leq |S_k| - 1$ **(d)** Nonzeros = weight of corresponding edge minus that of parent: $u_{S_1} = 5, u_{S_2} = 7, u_{S_3} = 1, u_{S_4} = 2, u_{S_5} = 4, u_{S_6} = 4$. **(e)** Primal feasible with part (a) solution

of $|V| - 1$ edges; part (c) constraints active (satisfied and complementary), main constraints for S not mentioned in (c) satisfied but inactive with complementary $u_S = 0$ because they include multiple subtree components. Dual value = $\sum_k (|S_k| - 1)u_{S_k} = 68$. Dual constraints on $x_{i,j} = 1$ active (satisfied and complementary) because summing upwards in tree of (b) from that edge adds and subtracts to produce $c_{i,j}$. Dual constraints for other $x_{i,j} = 0$ satisfied and complementary because summing upwards from first S_k containing (i, j) will total edge weight associated with S_k which is $\geq c_{i,j}$ because (i, j) was not selected by greedy. **(f)** With $|E| = 12$ and $|V| = 7$ bound = $12 \log_2 12 + (7)^2 \approx 92$ steps.

Chapter 11

11-1 (a) proportional to variable magnitude

(b) $\mathbf{x}^* = (96.67, 5, 48.33)$ **(c)** $\min 2250y_1 + 1650y_2 + 2700y_3, \text{s.t. } 150y_1 + 150y_2 + 150y_3 = 150, 300y_1 + 600y_2 + 300y_3 \leq 310, 600y_1 + 450y_2 = 150y_3 \leq 450, y_j = 0 \text{ or } 1, j = 1, \dots, 3$ **(d)** $\mathbf{x}^* = (0, 0, 150), \mathbf{y}^* = (0, 0, 1)$
(e) $\min 15x_1 + 11x_2 + 18x_3 + 400(y_1 + y_2 + y_3), \text{s.t. } x_j \leq 150y_j, y_j = 0 \text{ or } 1, j = 1, \dots, 3, \text{and all original constraints}$ **(f)** $\mathbf{x}^* = (0, 0, 150), \mathbf{y}^* = (0, 0, 1)$ **(g)** add variables y_1, y_2, y_3 , and constraints $50y_j \leq x_j \leq 150y_j, y_j = 0 \text{ or } 1, j = 1, \dots, 3$ **(h)** $\mathbf{x}^* = (100, 0, 50), \mathbf{y}^* = (1, 0, 1)$

11-2 (h) $\mathbf{x}^* = (175, 0, 125), \mathbf{y}^* = (1, 0, 1)$

11-3 (a) $\max 4.5x_1 + 4.1x_2 + 8x_3 + 7x_4, \text{s.t. } 4x_1 + 3.8x_2 + 6x_3 + 7.2x_4 \leq 8, x_j = 0 \text{ or } 1, j = 1, \dots, 4$ **(b)** $\mathbf{x}^* = (1, 1, 0, 0)$

11-5 (a) budget: \$100 total; mutual exclusiveness: alternatives for NW and SE parcels; dependency: IE tunnel upon IE lab **(b)** $\max 9x_1 + 2x_2 + 10x_3 + 2x_4 + 5x_5 + 8x_6 + 10x_7 + 1x_8, \text{s.t. } 48x_1 + 20.8x_2 + 32x_3 + 28x_4 + 44x_5 + 17.2x_6 + 36.8x_7 + 1.2x_8 \leq 100, x_1 + x_4 + x_5 + x_6 \leq 1, x_2 + x_7 \leq 1, x_8 \leq x_7, x_1, \dots, x_8 = 0 \text{ or } 1$ **(c)** $\mathbf{x}^* = (0, 0, 1, 0, 0, 1, 1, 1)$

11-7 (a) $\max 45x_1 + 30x_2 + 84x_3 + 73x_4 + 80x_5 + 70x_6 + 61x_7 + 91x_8, \text{s.t. } x_1 + x_2 \leq 1, x_2 + x_5 \leq 1, x_4 + x_7 \leq 1, x_5 + x_6 + x_7 \leq 1, x_1, \dots, x_8 = 0 \text{ or } 1$ **(b)** $\mathbf{x}^* = (1, 0, 1, 1, 0, 0, 0, 1)$

11-9 (a) $\min 40x_1 + 65x_2 + 43x_3 + 48x_4 + 72x_5 + 36x_6, \text{s.t. } x_1 + x_2 \geq 1, x_1 + x_4 \geq 1,$

$x_2 + x_3 \geq 1, x_2 + x_4 \geq 1, x_2 + x_5 \geq 1,$
 $x_3 + x_5 \geq 1, x_3 + x_6 \geq 1, x_4 + x_5 \geq 1,$
 $x_5 + x_6 \geq 1, x_1, \dots, x_6 = 0$ or 1
(b) $\mathbf{x}^* = (0, 1, 1, 1, 0, 1)$ **(c)** $\min \sum_{i=1}^9 y_i,$
s.t. $x_1 + x_2 + y_1 \geq 1, x_1 + x_4 + y_2 \geq 1,$
 $x_2 + x_3 + y_3 \geq 1, x_2 + x_4 + y_4 \geq 1,$
 $x_2 + x_5 + y_5 \geq 1, x_3 + x_5 + y_6 \geq 1,$
 $x_3 + x_6 + y_7 \geq 1, x_4 + x_5 + y_8 \geq 1,$
 $x_5 + x_6 + y_9 \geq 1, \sum_{j=1}^6 x_j \leq 2, x_1, \dots, x_6 = 0$ or
1, $y_1, \dots, y_9 = 0$ or 1 **(d)** $\mathbf{x}^* = (0, 1, 0, 0, 1, 0),$
 $\mathbf{y}^* = (0, 1, 0, 0, 0, 1, 0, 0)$

11-11 (a) $\min 1.40x_1 + .96x_2 + 1.52x_3 +$
 $1.60x_4 + 1.32x_5 + 1.12x_6 + .84x_7 + 1.54x_8,$
s.t. $x_2 + x_5 + x_8 = 1, x_1 + x_3 + x_4 = 1,$
 $x_5 + x_6 + x_8 = 1, x_1 + x_3 + x_7 + x_8 = 1,$
 $x_3 + x_4 + x_6 = 1, x_2 + x_4 + x_5 + x_7 = 1,$
 $x_1, \dots, x_8 = 0$ or 1 **(b)** $\mathbf{x}^* = (0, 0, 1, 0, 1, 0, 0, 0)$

11-13 (a) $\min 18x_{1,1} + 26x_{1,2} + 31x_{1,4} +$
 $50x_{2,2} + 22x_{2,3} + 40x_{3,1} + 29x_{3,2} + 52x_{3,3} +$
 $39x_{3,4} + 43x_{4,3} + 46x_{4,4},$
s.t. $x_{1,1} + x_{1,2} + x_{1,4} = 1, x_{2,2} + x_{2,3} = 1,$
 $x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 1, x_{4,3} + x_{4,4} = 1,$
 $x_{1,1} + x_{3,1} = 1, x_{1,2} + x_{2,2} + x_{3,2} = 1,$
 $x_{2,3} + x_{3,3} + x_{4,3} = 1, x_{1,4} + x_{3,4} + x_{4,4} = 1,$
all $x_{i,j} \geq 0$ **(b)** can be viewed as a network
flow with unit supplies and demands **(c)**
 $x_{1,1}^* = x_{2,3}^* = x_{3,2}^* = x_{4,4}^* = 1$

11-19 (a) \min
 $60x_{H,1}x_{E,2} + 120x_{H,1}x_{E,3} + 36x_{H,1}x_{M,2} +$
 $72x_{H,1}x_{M,3} + 60x_{H,2}x_{E,1} + 20x_{H,2}x_{E,3} +$
 $36x_{H,2}x_{M,1} + 12x_{H,2}x_{M,3} + 120x_{H,3}x_{E,1} +$
 $20x_{H,3}x_{E,2} + 72x_{H,3}x_{M,1} + 12x_{H,3}x_{M,2} +$
 $42x_{E,1}x_{M,2} + 84x_{E,1}x_{M,3} + 42x_{E,2}x_{M,1} +$
 $14x_{E,2}x_{M,3} + 84x_{E,3}x_{M,1} + 14x_{E,3}x_{M,2},$ s.t.
 $x_{H,1} + x_{H,2} + x_{H,3} = 1, x_{E,1} + x_{E,2} + x_{E,3} = 1,$
 $x_{M,1} + x_{M,2} + x_{M,3} = 1, x_{H,1} + x_{E,1} + x_{M,1} = 1,$
 $x_{H,2} + x_{E,2} + x_{M,2} = 1, x_{H,3} + x_{E,3} + x_{M,3} = 1,$
all $x_{i,j} = 0$ or 1 **(b)** can assess cost only after
pairs of assignments **(c)** $x_{H,3}^* = x_{E,2}^* = x_{M,1}^* = 1$

11-21 (a) $\min \sum_{i=1}^6 (c_{i,F}x_{i,F} + c_{i,B}x_{i,B}),$ s.t.
 $x_{i,F} + x_{i,B} = 1, i = 1, \dots, 6, \sum_{i=1}^6 t_i x_{i,F} \leq 200,$
 $\sum_{i=1}^6 t_i x_{i,B} \leq 190,$ all $x_{i,j} = 0$ or 1, where
 $c_{i,j} \triangleq$ the materials handling to move i to
 j and $t_i \triangleq$ the time for i **(b)** more than one
unit of time allocated by each decision **(c)**
 $x_{1,B}^* = x_{2,B}^* = x_{3,F}^* = x_{4,F}^* = x_{5,B}^* = x_{6,F}^* = 1$

11-24 (a) $\min \sum_{i=1}^5 \sum_{j=i+1}^6 c_{i,j} x_{i,j},$
s.t. $x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 1,$
 $x_{1,2} + x_{2,3} + x_{2,4} + x_{2,5} + x_{2,6} = 1,$
 $x_{1,3} + x_{2,3} + x_{3,4} + x_{3,5} + x_{3,6} = 1,$

$x_{1,4} + x_{2,4} + x_{3,4} + x_{4,5} + x_{4,6} = 1,$
 $x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{5,6} = 1,$
 $x_{1,6} + x_{2,6} + x_{3,6} + x_{4,6} + x_{5,6} = 1,$ all $x_{i,j} = 0$
or 1, where $c_{i,j} \triangleq$ the cost of pairing i with j
(b) objects to be paired do not come from dis-
tinct sets **(c)** $x_{1,5}^* = x_{2,3}^* = x_{4,6}^* = 1$

11-26 (a) seeks a minimum total length closed
route visiting every point **(b)**

	1	2	3	4	5	6
1	-	40	230	160	220	40
2	40	-	190	120	180	20
3	230	190	-	70	30	190
4	160	120	70	-	60	120
5	220	180	30	60	-	180
6	40	20	190	120	180	-

(c) $\min \sum_{i=1}^5 \sum_{j=i+1}^6 d_{i,j} x_{i,j},$
s.t. $x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 2,$
 $x_{1,2} + x_{2,3} + x_{2,4} + x_{2,5} + x_{2,6} = 2,$
 $x_{1,3} + x_{2,3} + x_{3,4} + x_{3,5} + x_{3,6} = 2,$
 $x_{1,4} + x_{2,4} + x_{3,4} + x_{4,5} + x_{4,6} = 2,$
 $x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{5,6} = 2,$
 $x_{1,6} + x_{2,6} + x_{3,6} + x_{4,6} + x_{5,6} = 2,$ all $x_{i,j} = 0$ or
1, where $d_{i,j} \triangleq$ the i to j distance of **(b)**
(d) $x_{1,2}^* = x_{1,6}^* = x_{2,6}^* = x_{3,4}^* = x_{3,5}^* = x_{4,5}^* = 1$
(e) $x_{1,3} + x_{1,4} + x_{1,5} + x_{2,3} + x_{2,4} + x_{2,5} + x_{3,6} +$
 $x_{4,6} + x_{5,6} \geq 2$ **(f)** $x_{1,2}^* = x_{1,6}^* = x_{2,3}^* = x_{3,5}^* =$
 $x_{4,5}^* = x_{4,6}^* = 1$ **(g)** model [\[11.27\]](#) with $d_{i,j}$ as in **(b)**

11-28 (a) seeks a minimum total setup time
closed tour of the 4 products, and times are not
symmetric **(b)** $\min \sum_{i=1}^4 \sum_{j \neq i} c_{i,j} x_{i,j},$
s.t. $\sum_{j \neq i} x_{i,j} = 1, i = 1, \dots, 4,$
 $\sum_{i \neq j} x_{i,j} = 1, j = 1, \dots, 4,$ all $x_{i,j} \geq 0,$
where $c_{i,j} \triangleq$ the given i to j setup time.

(c) $x_{1,3}^* = x_{3,1}^* = x_{2,4}^* = x_{4,2}^* = 1$

(d) $x_{1,2}^* + x_{1,4}^* + x_{3,2}^* + x_{3,4}^* \geq 1$

(e) $x_{1,2}^* = x_{2,4}^* = x_{3,1}^* = x_{4,3}^* = 1$

(f) model [\[11.27\]](#) with $d_{i,j}$ the given setup times

11-30 (a) $\min \sum_{i=1}^3 (f_i y_i + \sum_{j=1}^4 d_j c_{i,j} x_{i,j}),$ s.t.
 $\sum_{j=1}^4 x_{i,j} \leq 4y_i, i = 1, \dots, 3, \sum_{i=1}^3 y_i = 1,$
 $j = 1, \dots, 4, y_1, \dots, y_3 = 0$ or 1, all $x_{i,j} \geq 0,$
where $f_i \triangleq$ the given fixed cost for $i,$
 $c_{i,j} \triangleq$ the given transportation cost from
 i to $j,$ and $d_j \triangleq$ the given demand at $j.$

(b) $y_3^* = x_{3,1}^* = x_{3,2}^* = x_{3,3}^* = 1$

11-32 (a) \min

$2x_{F1,S1} + 8000y_{F1,S1} + 2x_{F2,F1} + 6000y_{F2,F1} +$
 $2x_{F2,S1} + 10000y_{F2,S1} + 2x_{F2,S2} + 14000y_{F2,S2} +$
 $2x_{S1,S2} + 2000y_{S1,S2} + 2x_{S1,T} + 2x_{S2,T},$

s.t. $x_{F2,F1} - x_{F1,S1} = -800$,
 $-x_{F2,F1} - x_{F2,S1} - x_{F2,S2} = -600$,
 $x_{F1,S1} + x_{F2,S1} - x_{S1,S2} - x_{S1,T} = 0$,
 $x_{F2,S2} + x_{S1,S2} - x_{S2,T} = 0$, $x_{S1,T} + x_{S2,T} = 1400$,
 $x_{F1,S1} \leq 1000y_{F1,S1}$, $x_{F2,F1} \leq 1000y_{F2,F1}$,
 $x_{F2,S1} \leq 1000y_{F2,S1}$, $x_{F2,S2} \leq 1000y_{F2,S2}$,
 $x_{S1,S2} \leq 1000y_{S1,S2}$, all $x_{i,j} \geq 0$, all $y_{i,j} = 0$ or 1
(b) $y_{F1,S1} = y_{F2,S2} = 1$, $x_{F1,S1} = x_{S1,T} = 800$,
 $x_{F2,S2} = x_{S2,T} = 600$

11-34 (a) $x_1 + 10 \leq x_2 + M(1 - y_{1,2})$,
 $x_2 + 2 \leq x_1 + My_{1,2}$,
 $x_1 + 10 \leq x_3 + M(1 - y_{1,3})$,
 $x_3 + 16 \leq x_1 + My_{1,3}$,
 $x_1 + 10 \leq x_4 + M(1 - y_{1,4})$,
 $x_4 + 8 \leq x_1 + My_{1,4}$,
 $x_2 + 3 \leq x_3 + M(1 - y_{2,3})$,
 $x_3 + 16 \leq x_2 + My_{2,3}$,
 $x_2 + 3 \leq x_4 + M(1 - y_{2,4})$,
 $x_4 + 8 \leq x_2 + My_{2,4}$,
 $x_3 + 16 \leq x_4 + M(1 - y_{3,4})$,
 $x_4 + 8 \leq x_3 + My_{3,4}$, $x_1 \geq 0$, $x_2 \geq 20$, $x_3 \geq 1$,
 $x_4 \geq 12$, all $y_{i,j} = 0$ or 1 **(b)** 39, 23.5, 38,
15.25, 19, 2.75, 19, 4.75 **(c)**
 $\min 1/4(x_1 + 10 + x_2 + 3 + x_3 + 16 + x_4 + 8)$,
s.t. constraints of **(a)** **(d)** $x_1^* = 0$, $x_2^* = 20$,
 $x_3^* = 23$, $x_4^* = 2$ **(e)** mean flow time and mean
lateness **(f)** $\min z$, s.t. $z \geq x_1 - 2$, $z \geq x_2 - 27$,
 $z \geq x_3 - 4$, $z \geq x_4 - 13$, plus all constraints of
(a) **(g)** $x_1^* = 0$, $x_2^* = 34$, $x_3^* = 10$, $x_4^* = 26$ **(h)**
maximum tardiness

11-36 (a) $x_{1,1} + 10 \leq x_{1,2}$, $x_{1,2} + 3 \leq x_{1,3}$,
 $x_{2,1} + 2 \leq x_{2,3}$, $x_{2,3} + 1 \leq x_{2,2}$,
 $x_{3,2} + 6 \leq x_{3,1}$, $x_{3,1} + 12 \leq x_{3,3}$,
 $x_{1,1} + 10 \leq x_{2,1} + M(1 - y_{1,2,1})$,
 $x_{2,1} + 2 \leq x_{1,1} + My_{1,2,1}$,
 $x_{1,1} + 10 \leq x_{3,1} + M(1 - y_{1,3,1})$,
 $x_{3,1} + 12 \leq x_{1,1} + My_{1,3,1}$,
 $x_{2,1} + 2 \leq x_{3,1} + M(1 - y_{2,3,1})$,
 $x_{3,1} + 12 \leq x_{2,1} + My_{2,3,1}$,
 $x_{1,2} + 10 \leq x_{2,2} + M(1 - y_{1,2,2})$,
 $x_{2,2} + 2 \leq x_{1,2} + My_{1,2,2}$,
 $v x_{1,2} + 10 \leq x_{3,2} + M(1 - y_{1,3,2})$,
 $x_{3,2} + 12 \leq x_{1,2} + My_{1,3,2}$,
 $x_{2,2} + 2 \leq x_{3,2} + M(1 - y_{2,3,2})$,
 $x_{3,2} + 12 \leq x_{2,2} + My_{2,3,2}$,
 $x_{1,3} + 10 \leq x_{2,3} + M(1 - y_{1,2,3})$,
 $x_{2,3} + 2 \leq x_{1,3} + My_{1,2,3}$,
 $x_{1,3} + 10 \leq x_{3,3} + M(1 - y_{1,3,3})$,
 $x_{3,3} + 12 \leq x_{1,3} + My_{1,3,3}$,
 $x_{2,3} + 2 \leq x_{3,3} + M(1 - y_{2,3,3})$,

$x_{3,3} + 12 \leq x_{2,3} + My_{2,3,3}$, all $x_{j,k} \geq 0$,
all $y_{i,j,k} = 0$ or 1 **(b)** mean flow time **(c)**
 $\min 1/3(x_{1,3} + 14 + x_{2,2} + 4 + x_{3,3} + 8)$, s.t.
all constraints of **(a)** **(d)** $x_{1,1}^* = 2$, $x_{1,2}^* = 12$,
 $x_{1,3}^* = 15$, $x_{2,1}^* = 0$, $x_{2,2}^* = 6$, $x_{2,3}^* = 5$, $x_{3,1}^* = 12$,
 $x_{3,2}^* = 0$, $x_{3,3}^* = 29$

Chapter 12

12-1 (a) $x^* = (0, 1, 1, 0)$

12-2 (a) 16, 21

12-3 (a) yes **(c)** no

12-4 (a) same except last two constraints
replaced by $0 \leq x_1 \leq 1$

12-5 (a) yes **(c)** no

12-6 (a) optimal ILP value ≤ 54.5 **(c)** optimal
ILP value ≥ 19

12-7 (a) no **(c)** yes

12-8 (a) ILP optimal value ≤ 18.5 **(b)** not
optimal; $\hat{x} = (0, 0, 1, 1, 0)$ **(c)** $14 \leq$ optimal
value ≤ 18.5 **(d)** $x^* = (0, 0, 1, 1, 0)$, value 14

12-12 (a) $x^* = (0, 0, 0, 0)$, $y^* = 0$ **(c)** same inte-
ger-feasible solutions of all = 0 or all = 1 **(d)**
solution value 0 vs. solution value -42

12-15 (a) 4, 6 **(b)** original LP optimum violates
 $x_2 \leq 6y_2$ **(c)** new $\tilde{x} = (0, 3.2)$, $\tilde{y} = (0, .533)$

12-17 (a) (0, 0, 0), (0, 0, 1), (1, 0, 0), (1, 0, 1)

12-18 (a) (#, #, #, #), (#, #, 1, #), (#, 1, 1, #), (#, 0, 1, #),
(#, #, 0, #) **(b)** branched: 0, 1; terminated: 2, 3, 4
(c) 0, 3

12-20 (a) given ILP plus constraints $x_2 = 1$, $x_5 = 0$

12-21 (a) branch on $x_1 = 1$ vs. $x_1 = 0$ **(c)** termi-
nate by solving after saving \tilde{x} as a new incumbent
solution **(e)** terminate by bound

12-22 partial solutions: (#, #, #), (#, #, 0), (0, #, 0),
(0, 1, 0), (0, 0, 0), (1, #, 0), (1, 0, 0), (1, 1, 0), (#, #, 1),
(1, #, 1), (0, #, 1); $x^* = (1, 0, 1, 0)$

12-29 0: branch on fractional x_3 ; 1: branch on
fractional x_2 ; 2: branch on fractional x_1 ; 3: termi-
nate by solving after saving incumbent solution
 $\hat{x} = (1, 0, 0)$, $\hat{v} = 90$; 4: terminate infeasible; 5:
terminate by solving after saving incumbent solu-
tion $\hat{x} = (0, 1, 0)$, $\hat{v} = 50$; 6: terminate by bound
 $54 \geq 50$; $x^* = (0, 1, 0)$

12-31 (b) 0: round for incumbent solution
 $\hat{x} = (0, 0, 1)$, $\hat{v} = 54$, branch on fractional
 x_3 ; 1: round for new incumbent solution

$\hat{x} = (0, 1, 0)$, $\hat{v} = 50$, branch on fractional x_2 ; 2: terminate by bound $75 \geq 50$; 3: terminate by bound $50 \geq 50$; 4: terminate by bound $54 \geq 50$; $x^* = (0, 1, 0)$

12-33 (a) bounds 18 to ∞ , 18 to ∞ , 18 to ∞ , 18 to 90, 18 to 90, 18 to 50, 50 to 50. Errors from node 3, 400%, 400%, 178%, 0%.

12-34 (a) a, b, a, b, c **(b)** 212 **(c)** 17; 8.0 percent

12-36 (a) partial solutions: $(\#, \#, \#)$, $(\#, \#, 0)$, $(\#, \#, 1)$, $(0, \#, 0)$, $(1, \#, 0)$, $(1, 0, 0)$, $(1, 1, 0)$, $(1, \#, 1)$, $(0, \#, 1)$; $(0, 1, 0)$, $(0, 0, 0)$; $x^* = (1, 0, 1, 0)$ **(b)** partial solutions: $(\#, \#, \#)$, $(\#, \#, 0)$, $(0, \#, 0)$, $(0, 1, 0)$, $(\#, \#, 1)$, $(1, \#, 1)$, $(1, \#, 0)$, $(1, 0, 0)$, $(1, 1, 0)$, $(0, \#, 1)$; $(0, 0, 0)$; $x^* = (1, 0, 1, 0)$

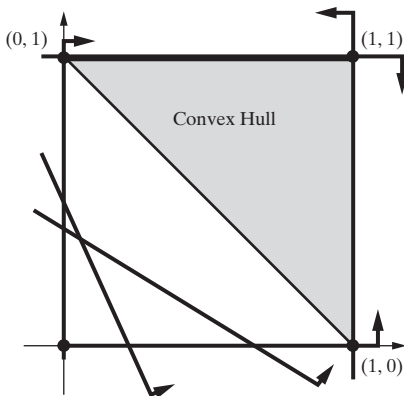
12-39 (a) valid; yes **(b)** valid; no **(c)** not valid **(d)** valid, yes

12-41 (a) valid; yes **(c)** not valid

12-42 0: fractional so introduce valid inequality $4x_1 + 3x_2 \geq 3$ which cuts off the current relaxation; 1: lacking further valid inequalities, branch on fractional x_1 ; 2: fractional so introduce valid inequality $x_2 + x_3 \leq 1$ which cuts off the current relaxation; 3: terminate infeasible; 4: lacking further valid inequalities, branch on fractional x_3 ; 5: terminate by solving after saving incumbent solution $\hat{x} = (0, 1, 1)$, $\hat{v} = 75$; 6: terminate infeasible; $x^* = (0, 1, 1)$

12-45 (a) $0.3x_3 + 0.1x_4 + (0.6/0.4)0.3x_6 + 0.4x_7 \geq 0.6$

12-49 (a)



(b) dimension = 2 justified by $(1,0)$, $(0,1)$, $(1,1)$. **(c)** original constraint, valid, no intersection. intersects at $(0,1)$, valid, lower-dimensional face. facet, valid, justified by $(1,0)$ and $(0,1)$.

Chapter 13

13-1 (a) for $j = 1, \dots, 5$, $a_{i,j} \leftarrow b_i$ if $i = j$ and $\leftarrow 0$ otherwise. $c_j \leftarrow \log_{10}(a_{j,j} + 1)$.

(b) using LP allows quick solutions of interim partial problems, and easy computation of duals \bar{v}_i needed in column generation. **(c)** $\min \bar{c}_g \triangleq \sum_{i=1}^4 [\log_{10}(a_{i,g} + 1) - a_{i,g}\bar{v}_i]$, subject to, $\sum_{i=1}^4 a_{i,g} \leq 5$, $a_{i,g} \geq 0$ and integer, $i = 1, \dots, 5$. Use optimal $a_{i,g}$ and $c_g \leftarrow \sum_{i=1}^4 \log_{10}(a_{i,g} + 1)$. **(d)** the result of (c) will be a column qualified to enter if its $\bar{c}_g < 0$. The previous partial problem LP could not have terminated optimal if one of its columns have negative reduced cost. **(e)** if the optimal \bar{c}_g in (c) is ≥ 0 , there can be no new column that can enter the partial problem to improve the previous solution.

13-4 (a) minimizing. **(b)** 0: fractional, generate new x_4 . 1: integer, incumbent $\hat{x} = (1, 1, 0, 1)$, $\hat{v} = 43$, generate new x_5 . 2: fractional, no new columns, branch on $\tilde{x}_2 = 2/5$. 3: integer, not incumbent, no new columns, terminate. 4: fractional, cannot terminate, generate new x_6 . 5: fractional, no new columns, terminate $46 > \tilde{v} = 43$. $x^* = (1, 1, 0, 1)$

13-6 (a) $\max 30x_1 + 55x_2 + 20x_3 + v_1(55 - 40x_1 + 12x_2 - 11x_3) + v_2(20 - 19x_1 - 60x_2 - 3x_3)$, s.t. all undualized constraints; $v_1 \geq 0, v_2 \leq 0$

13-7 (a) $x_{1,1}^* = x_{1,2}^* = y_1^* = 1$ with objective value 259. **(b)** $\min 3x_{1,1} + 6x_{1,2} + 5x_{2,1} + 2x_{2,2} + 250y_1 + 300y_2 + v_1(1 - x_{1,1} - x_{2,1}) + v_2(1 - x_{1,2} + x_{2,2})$, s.t. all undualized constraints. both v_1 and v_2 are URS because they relate to equality constraints. **(c)** in the relaxation, optimizations are now independent for facilities 1 and 2. we may solve them separately and sum the results along with the constant $(1v_1 + 1v_2)$ in the objective function. **(d)** all $\tilde{x}_{i,j}$ and $\tilde{y}_i = 0$, with relaxation value $0 \leq 259$. **(e)** all $\tilde{x}_{i,j}$ and $\tilde{y}_i = 0$, with relaxation value $-200 \leq 259$. **(f)** $\tilde{x}_{1,1} = 1, \tilde{x}_{1,2} = 0, \tilde{x}_{2,1} = 1, \tilde{x}_{2,2} = 1, \tilde{y}_1 = 1, \tilde{y}_2 = 1$, with relaxation value $-440 \leq 259$.

13-8 (g) subgradient $\Delta v = (0, 1)$. new duals $v_1 \leftarrow 1000, v_2 \leftarrow 1000$. same solution as (f).

(h) $v_1 \leftarrow 1000, v_2 \leftarrow 750$. same solution as (f).

(i) yes.

13-12 (b) polyhedral sets are convex

(c) $w^{(1)} = (1, 0), w^{(2)} = (0, 1/2)$,

$\mathbf{w}^{(3)} = (0, 1)$ **(d)** $\Delta \mathbf{w}^{(1)} = (1, 1)$,
 $\Delta \mathbf{w}^{(2)} = (1, 0)$ **(e)** $\bar{\mathbf{w}}^{(1)} = (0, 1) = 1\mathbf{w}^{(3)}$,
 $\bar{\mathbf{w}}^{(2)} = (1, 1) = 1\mathbf{w}^{(3)} + 1\Delta \mathbf{w}^{(2)}$,
 $\bar{\mathbf{w}}^{(3)} = (3/4, 3/2) = 1/2\mathbf{w}^{(2)} + 1/2\mathbf{w}^{(3)} + 3/2\Delta \mathbf{w}^{(2)}$
(f) no.

13-13 (a) set is polyhedral so convex. **(b)** extreme points $\mathbf{x}^{(1)} = (7, 0)$, $\mathbf{x}^{(2)} = (3, 0)$, $\mathbf{x}^{(3)} = (7, 4)$, no extreme-directions. **(c)** set is polyhedral so convex. **(d)** extreme points $\mathbf{x}^{(2,1)} = (0, 10)$, $\mathbf{x}^{(2,2)} = (8, 2)$. extreme direction $\Delta \mathbf{x}^{(2,1)} = (0, 1)$. **(e)** substitute in linking constraints. partial master $\ell = 1$ is

$$\begin{aligned} \max \quad & 52(x_1^{(1,1)}\lambda^{(1,1)} + 19(x_2^{(1,1)}\lambda^{(1,1)} \\ & + 41(x_3^{(2,1)}\lambda^{(2,1)} + 9(x_4^{(2,1)}\lambda^{(2,1)} \\ \text{s.t.} \quad & 12(x_1^{(1,1)} * \lambda^{(1,1)} + 20(x_2^{(1,1)} * \lambda^{(1,1)} \\ & + 15(x_3^{(2,1)}\lambda^{(2,1)} + 8(x_4^{(2,1)}\lambda^{(2,1)} \leq 90 \\ & 9(x_1^{(1,1)}\lambda^{(1,1)} + 10(x_2^{(1,1)}\lambda^{(1,1)} \\ & + 13(x_3^{(2,1)}\lambda^{(2,1)} + 18(x_4^{(2,1)}\lambda^{(2,1)} \leq 180 \\ & \lambda^{(1,1)} = 1 \\ & \lambda^{(2,1)} = 1 \\ & \lambda^{(1,1)} \geq 0 \\ & \lambda^{(2,1)} \geq 0 \end{aligned}$$

(f) an optimal primal solution is $\bar{\lambda}^{(1,1)} = \bar{\lambda}^{(2,1)} = 1$, objective value 90. optimal main duals $\bar{v}_1 = \bar{v}_2 = 0$, and the optimal duals on $= 1$ constraints are $\bar{q}_1^{(1)} = 0, \bar{q}_2^{(1)} = 90$. **(g)** $\bar{c}_1^{(1)} = 52, \bar{c}_2^{(1)} = 19, \bar{c}_3^{(2)} = 41, \bar{c}_4^{(2)} = 9$. subproblem 1 max $52x_1 + 19x_2 - 0$, subproblem 2 max $41x_3 + 9x_4 - 90$. first solves at $\bar{x}_1 = 7, \bar{x}_2 = 4$. Second is unbounded and yielding extreme direction $\Delta \mathbf{x} = (0, 1)$. **(h)** add weighting of $\lambda^{(1,2)}$ in terms for x_1 and x_2 , and weighting of $\mu^{(2,2)}$ in terms for x_3 and x_4 .

13-15 (a) primal max $60x_1 + 50x_2 - 25y_1^{(\ell)} - 100y_2^{(\ell)}$, subject to $20x_1 + 17x_2 \leq 10 + 60y_1^{(\ell)} + 30y_2^{(\ell)}$, $11x_1 + 13x_2 \leq 10 + 30y_1^{(\ell)} + 60y_2^{(\ell)}$, $x_1, x_2 \geq 0$. dual min $(10 + 60y_1^{(\ell)} + 30y_2^{(\ell)})v_1 + (10 + 30y_1^{(\ell)} + 60y_2^{(\ell)})v_2 - 25y_1^{(\ell)} - 100y_2^{(\ell)}$ subject to $20v_1 + 11v_2 \geq 60, 17v_1 + 13v_2 \geq 50, v_1, v_2 \geq 0$. **(b)** dual 1: $v_1 = 3, v_2 = 0$, value = 30. mstr 1: $z = 1580, y_1 = 10, y_2 = 0$. dual 2: $v_1 = 0, v_2 = 5.45455$, value 1440.91. mstr 2: $z = 1570, y_1 = 10, y_2 = 1$. Dual 3: $v_1 = 3, v_2 = 0$, value = 1570. stop on repeat and recover $x_1^* = 32, x_2^* = 0$.

Chapter 14

14-2 (a) $\{0, 1, \dots, 9, \#, \&\}$,
 $4\#7\&8\#9\&4\#21\&12\#15\&7\#19$. **(b)** $O(n)$ each with length number of parameter digits. **(c)** $n = 4, c_1 = 7, c_2 = 9, c_3 = 21, c_4 = 15, a_1 = 8, a_2 = 4, a_3 = 12, a_4 = 7, b = 19$. **(d)** polynomial $O(nb)$; exponential $O(n2^k)$ in n and digits $k \triangleq \lceil \log b + 1 \rceil$ of b . **(e)** pseudo-polynomial means polynomial in magnitudes. **(f)** parts (d) and (e) say instances solvable for modest coefficients; instances with large coefficients remain hard. **(g)** yes; reduction allows instances of very hard problems to be reduced to instances of **(BKP)**; unlikely to be easy after reduction.

14-3 (a) $\{0, 1, \dots, 9, \#, \&\}$. $3\&4\#1880\&350\#1633\&250\#1455\&200\#110\#213\#75\#96\#12\&4\&7\&9\#5\&8\&12\&17\#22\&3\&6\&16$. **(b)** parameter clusters grow with m, n ; cluster size is number parameter value digits; logarithmic in magnitudes. **(c)** given indices $i = 1, \dots, m$ and $j = 1, \dots, n$, corresponding costs $c_{i,j}$; fixed costs f_i , capacities u_i , demands d_j , and threshold v , do there exist $x_{i,j} \geq 0$ and y_i binary, such that $\sum_i \sum_j c_{i,j}d_jx_{i,j} + \sum_i f_iy_i \leq v, \sum_i x_{i,j} = 1$ for all j , and $\sum_i d_ix_{i,j} \leq u_iy_i$ for all i . **(d)** question for **(FLP[≤])** is yes/no with yes solution checkable in polynomial time; thus **NP**; full model **(FLP)** requires an optimal solution. **(e)** **(FLP[≤])** \propto **(FLP)** because solving any instance of **(FLP)** would answer **(FLP[≤])** question for any v ; **(FLP[≤])** in **NP-Complete** means **(FLP)** in **NP-Hard**. **(f)** polynomial solution of any **NP-Hard** implies one for all **NP**; highly unlikely.

14-4 (d) add a super-source node 0 and create arcs from 0 to all sources i with $u_{0,i} =$ capacity at $i, f_{0,i} = f_i$ and $c_{0,i} = 0$. then make supply at node $\sum_i u_{0,i}$, and all supply points transshipment. then with same threshold, a qualifying solution proving 'yes' to this instance of **(FCNP[≤])** will serve for **(FLP[≤])**. **(e)** **(FLP[≤])** \in **NP - Complete**, **(FLP[≤])** \propto **(FCNP[≤])** and **(FCNP[≤])** \in **NP** proves **(FCNP[≤])** \in **NP - Complete**. **(f)** **(FCNP[≤])** \propto **(FCNP)** **(g)** polynomial algorithm for either would provide one for all members of **NP**, which is highly unlikely.

14-8 (a) $\hat{V} = \{1, 2\}, (3, 5)$ gives $\hat{V} = \{1, 2, 3, 5\}, (6, 7)$ gives final $\hat{V} = \{1, 2, 3, 5, 6, 7\}$. **(b)** Proceeds until there is no uncovered edge. **(c)** $O(|E|)$ steps to perform $O(|V|)$ checks/updates.

14-10 (a) (MST) is the special case of **(Stein)** with all nodes Steiner. Easier because no combinatorial decisions about non-Steiner nodes to span.

Chapter 15

15-1 (a) $\mathbf{x}^* = (1, 0, 1, 0)$ **(b)** most payoff per unit constraint usage **(c)** $\hat{\mathbf{x}} = (0, 1, 0, 1)$

15-8 (a) taking items in sequence, bins are, 1,1,2,2,1,3,3,3,4,4,5,4.

15-9 (a) $\mathbf{x}^* = (1, 1, 0)$ **(b)** (0,0,1), (1,0,1), (0,1,1), (0,0,2) **(c)** $\hat{\mathbf{x}} = (1, 0, 1)$ **(d)** $\hat{\mathbf{x}} = (1, 1, 0)$ **(f)** $\hat{\mathbf{x}} = (1, 1, 0)$

15-11 (a) $\mathbf{x}^* = (0, 1, 1, 0)$ **(b)** $\hat{\mathbf{x}} = (1, 0, 0, 0)$ **(c)** $\hat{\mathbf{x}} = (0, 1, 1, 0)$

15-15 $\hat{\mathbf{x}} = (0, 1, 1, 0)$

15-17 $\hat{\mathbf{x}} = (0, 1, 0, 1)$

15-23 (a) check both feasible **(b)** after 1 or 2: $\mathbf{x}^{(3)} = \mathbf{x}^{(2)}$, $\mathbf{x}^{(4)} = \mathbf{x}^{(1)}$; after 3: $\mathbf{x}^{(3)} = (0, 0, 1, 1)$, $\mathbf{x}^{(4)} = (0, 0, 0, 0)$ **(c)** all feasible except $\mathbf{x}^{(3)}$ cutting after 3; infeasibles must either be excluded from the population or included with a large negative objective value

15-25 (0, 1, 1, 0), value 16; (0, 0, 1, 0), value 9; (0, 1, 0, 1), value 15; and any feasible immigrant such as (0, 1, 0, 0), value 7

Chapter 16

16-1 (a) $\min 40(x/2) + 2000/(x/5)$ **(b)** $x^* = 22.4$

16-2 (b) $\alpha^* = .031$

16-3 (a) $\min \sqrt{(x_1)^2 + (x_2 + 30)^2} + \sqrt{(x_1 - 50)^2 + (x_2 + 10)^2} + \sqrt{(x_1 - 70)^2 + (x_2 + 20)^2} + \sqrt{(x_1 - 30)^2 + (x_2 + 50)^2}$
(b) $\mathbf{x}^* = (45.8, 2.7)$

16-5 (a) $\min \sum_{i=1}^4 (t_i - a(u_i)^b)^2$ where u_i and t_i are the given units and average time values

(b) $a^* = 10.36$, $b^* = -.322$

16-6 (b) $k^* = 62.79$, $a^* = 2.011$, $b^* = -.363$

16-11 (b) defining $\alpha = am$, $\beta = (bm - a)$, $\gamma = -b$, the fitted equation has the linear form $\alpha + \beta n_{t-1} + \gamma(n_t)^2$

16-13 (a) yes **(c)** no **(e)** yes

16-14 (a) local maximum; local and global minimum; nothing; local and global maximum; nothing; local minimum

16-15 (a) local maximum; local and global maximum; nothing; local and global minimum; local minimum

16-16 at $t = 4$, $x^* \approx 2.47$

16-18 (a) $x^{(hi)} = 4.5$ **(b)** $x^{(hi)} = 9$

16-20 at $t = 3$, $x^* \approx 2.79$

16-22 (a) $f_1(3 + \lambda) = 33 + 20\lambda$

(b) $f_2(3 + \lambda) = 33 + 20\lambda + 6\lambda^2$

16-24 (a) $f_1((0, 2) + \lambda(1, -1)) = 24 - 34\lambda$

(b) $f_2((0, 2) + \lambda(1, -1)) = 24 - 34\lambda + 11\lambda^2$

16-26 (a) stationary **(c)** $\Delta \mathbf{x} = (-16, 3)$

16-27 (a) definitely local minimum **(c)** definitely neither **(e)** possibly local maximum and possibly local minimum **(g)** possibly local minimum

16-28 (a) concave **(c)** neither **(e)** both **(g)** convex **(i)** convex

16-29 (a) global minimum

16-30 (b) $\Delta \mathbf{x} = (23, 97)$ **(c)** max

$100 - 5((1 + 23\lambda) - 2)^4 - 3((3 + 97\lambda) - 5)^4 + (1 + 23\lambda)(3 + 97\lambda)$, s.t. $\lambda > 0$ **(d)** $\lambda = .027$, $\mathbf{x}^{(1)} = (1.621, 5.619)$ **(e)** $\mathbf{x}^{(2)} = (2.627, 5.435)$, $\mathbf{x}^{(3)} = (2.675, 5.592)$

16-32 (a) $f_2((3, 7) + \Delta \mathbf{x}) \triangleq 68 +$

$(-13, -93) \Delta \mathbf{x} + \frac{1}{2} \Delta \mathbf{x} \begin{pmatrix} -60 & 1 \\ 1 & -144 \end{pmatrix} \Delta \mathbf{x}$

(b) $\Delta \mathbf{x} = (-.2275, -.6474)$

(c) $\mathbf{x}^{(1)} = (2.773, 6.353)$, $\mathbf{x}^{(2)} = (2.681, 5.942)$

16-34 (a) $\Delta \mathbf{x} = (3, 98)$ **(b)** $\mathbf{x}^{(1)} = (2.078, 5.548)$,

$\mathbf{D} = \begin{pmatrix} -1.003 & -.0268 \\ -.0268 & -.0267 \end{pmatrix}$, $\Delta \mathbf{x} = (5.555, 0.151)$

16-36 (a) the function is not differentiable everywhere **(b)** $\mathbf{x}^{(0)} = (5, 2.5)$,

$\mathbf{x}^{(1)} = (0, 1.25)$, $\mathbf{x}^{(2)} = (-1.25, -.9375)$,

$\mathbf{x}^{(3)} = (-1.5625, -.2344)$

16-38 (a) $\mathbf{y}^{(1)} = (1, 2, 1)$, $\mathbf{y}^{(2)} = (3, 3, 3)$,

$\mathbf{y}^{(3)} = (2, 2, 4)$, $\mathbf{y}^{(4)} = (4, 2, 4)$

Chapter 17

17-1 (a) max lwh , s.t. $w + 2h \leq 30$, $l + 2h \leq 40$, $l, w, h \geq 0$ **(b)** $l^* = 28.685$, $w^* = 18.685$, $h^* = 5.657$

17-2 (b) $l^* = 27.386$, $w^* = 27.386$, $h^* = 66.667$, $d^* = 52.974$

17-3 (a) $\min \sum_{j=1}^4 (h_j x_j / 2 + s_j d_j / x_j)$, s.t. $\sum_{j=1}^4 v_j x_j \leq 4000$, $x_1, \dots, x_5 \geq 0$, where s_j ,

v_j, h_j and d_j are the values in the table. **(b)**
 $x^* = (41.423, 55.937, 47.602, 52.600, 27.441)$

17-4 (b) $x^* = (10.338, 9.456, 6.304, 8.189, 3.740)$

17-5 (a) $\min 52 (42/(60Nf)) + 0.1$
 $(42/(5^{6.667} N^{-5.667} f^{-3})) +$
 $87 (42/(5^{6.667} N^{-5.667} f^{-3})),$
 s.t. $200 \leq N \leq 600, .001 \leq f \leq .005$

(b) $N^* = 200, f^* = .001$

17-6 (b) $x^* = (83.052, 26.263, 37.142, 31.391, 74.284)$

17-7 (a) $\min \sum_{i=1}^4 \sum_{j=1}^2 w_{ij}$
 $\sqrt{(e_i - x_j)^2 + (n_i - y_j)^2},$

s.t. $\sum_{j=1}^2 w_{ij} = d_i, i = 1, \dots, 5, \sum_{i=1}^4 w_{ij} \leq 200,$
 $j = 1, 2, w_{ij} \geq 0, i = 1, \dots, 5, j = 1, 2,$ where $d_i,$
 e_i and n_i are the values in the table. **(b)** $x_1^* = 20,$
 $y_1^* = 17, x_2^* = 4, y_2^* = 30, w_{1,1}^* = 10, w_{1,2}^* = 50,$
 $w_{2,2}^* = 90, w_{3,1}^* = 35, w_{4,1}^* = 85, w_{5,1}^* = 80$

17-8 (b) $x^* = 1.897, y^* = 7.186,$
 $w^* = (10, 0, 20, 40, 30)$

17-9 (a) $\min \sum_{i=1}^3 \sum_{j=1}^3 v_{ij} x_i x_j,$ s.t. $\sum_{i=1}^3 x_i = 1.5,$
 $\sum_{i=1}^3 m_i x_i \geq (10)(1.5), x_1, x_2, x_3 \geq 0,$ where
 the m_i are the mean return rates and v_{ij} the
 covariances in the given table. **(b)** $x_1^* = 0.282,$
 $x_2^* = 1.218, x_3^* = 0.0$

17-10 (b) $x^* = (0.518, 0.100, 0.382)$

17-11 (a) $\sum_{j=1}^5 c_j x_j,$ s.t. $\sum_{j=1}^4 x_j = 1,$

$12 \leq \sum_{j=1}^4 a_{1j} x_j \leq 16, 31 \leq \sum_{j=1}^4 a_{2j} x_j \leq 36,$
 $\ln(121) \leq \ln(\sum_{j=1}^4 a_{3j} x_j) \leq \ln(164),$
 $x_1, \dots, x_5 \geq 0,$ where the c_j are the costs, and
 a_{ij} the ingredient index values in the table.

(b) $x^* = (.778, 0, .222, 0, 0)$

17-12 (b) $x^* = (0, 0, 1217.742, 0, 32.288)$

17-15 $\max \sum_{j=1}^7 p_j s_j r_j(x_j),$ s.t.

$\sum_{j=1}^7 e_j x_j = \sum_{j=1}^7 e_j; .5 \leq x_j \leq 1.5, j = 1, \dots, 7$

17-21 $\min f_2(T_{1,1}, T_{1,2}, T_{2,1}, T_{2,2}, F_1, F_2) + f_3(S),$
 s.t. $T_{2,2} = f_1(T_{2,1}, F_2, S, T_0); (T_{1,1} - T_{1,2})F_1 = H;$
 $(T_{1,1} - T_{1,2})F_1 + kf_2(T_{1,1}, T_{1,2}, T_{2,1}, T_{2,2}, F_1, F_2)$
 $= (T_{2,1} - T_{2,2})F_2; \underline{T}_{ij} \leq T_{ij} \leq \bar{T}_{ij}, i, j = 1, 2;$
 $F_i \leq F_i \leq \bar{F}_i, i = 1, 2; \underline{S} \leq S \leq \bar{S}$

17-23 (a) yes **(c)** no **(e)** yes

17-24 (a) yes **(c)** yes **(e)** no

17-25 (a) not QP **(c)** QP: $c = (0, 0, 9),$

$$Q = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

17-26 (a) no **(c)** yes

17-27 (a) $\min 13x_1x_2(x_3)^{-1} + 9(x_1)^{\frac{1}{2}}(x_3)^{\frac{1}{2}},$
 s.t. $3x_1(x_3)^{-1} + 8x_2(x_3)^{-1} \leq 1.5(x_3)^{-4} \leq 1,$
 $x_1, x_2, x_3 > 0; K_0 = \{1, 2\}, K_1 = \{3, 4\},$
 $K_2 = \{5\}, d_1 = 13, d_2 = 9, d_3 = 3, d_4 = 8,$
 $d_5 = 5, a_{1,1} = 1, a_{1,2} = 1, a_{1,3} = -1, a_{2,1} = \frac{1}{2},$
 $a_{2,2} = 0, a_{2,3} = \frac{1}{2}, a_{3,1} = 1, a_{3,2} = 0, a_{3,3} = -1,$
 $a_{4,1} = 0, a_{4,2} = 1, a_{4,3} = -1, a_{5,1} = 0, a_{5,2} = 0,$
 $a_{5,3} = -4$

17-28 (a) $8(x_1 - 2)^2 + 2(x_2 - 1)^2 +$
 $v(126 - 32x_1 - 12x_2)$ **(b)** $16(x_1 - 2) - 32v = 0,$
 $4(x_2 - 1) - 12v = 0, 32x_1 + 12x_2 = 126$ **(c)**
 $x_1^* = 3, x_2^* = 5/2$ **(e)** $x_1^* = 2, x_2^* = 5.167$

17-30 (a) all given primal constraints,
 plus $2v_1 + 6v_2 + v_3 = 28(x_1 - 9),$
 $18v_1 + 8v_2 + v_4 = 6(x_2 - 5),$
 $-v_1 + 3v_2 = 2(x_3 - 11), v_2 \leq 0, v_3, v_4 \geq 0,$
 $v_2(20 - 6x_1 - 8x_2 - 3x_3), v_3(-x_1) = 0,$
 $v_4(-x_2) = 0$ **(c)** all given primal constraints,
 plus $(10(x_1 - 1))v_1 + v_3 = -2(x_1 - 3),$
 $(60(x_2 - 2))v_1 + 60v_2 + v_4 = -4(x_2)^3,$
 $39v_2 + v_5 = 19, v_1, v_3, v_4, v_5 = 0,$
 $v_1(35 - 5(x_1 - 1)^2 - 30(x_2 - 2)^2) = 0,$
 $v_3(-x_1) = 0, v_4(-x_2) = 0, v_5(-x_3) = 0$

17-31 (a) yes **(c)** no

17-32 (a) all given primal constraints, plus
 $3v_1 + v_2 = 30x_1, 2v_1 + v_3 = 8x_2, v_2, v_3 \geq 0,$
 $v_2(-x_1) = 0, v_3(-x_2) = 0$ **(e)** $v_1^* = 10,$
 $v_2^* = v_3^* = 0$

17-34 (a) $\min 14(x_1 - 9)^2 + 3(x_2 - 5)^2 +$
 $(x_3 - 11)^2 + \mu(|2x_1 + 18x_2 - x_3 - 19| +$
 $\max\{0, 6x_1 + 8x_2 + 3x_3 - 20\} +$
 $\max\{0, -x_1\} + \max\{0, -x_2\})$
(c) $\min 100 - (x_1 - 3)^2 - (x_2)^4 + 19x_3 +$
 $\mu(\max\{0, 35 - 5(x_1 - 1)^2 - 30(x_2 - 2)^2\} +$
 $|60x_2 + 39x_3 - 159| + \max\{0, -x_1\} +$
 $\max\{0, -x_2\} + \max\{0, -x_3\})$

17-35 (a) $\min 14(x_1 - 9)^2 + 3(x_2 - 5)^2 +$
 $(x_3 - 11)^2 + \mu(|2x_1 + 18x_2 - x_3 - 19|^2 +$
 $\max^2\{0, 6x_1 + 8x_2 + 3x_3 - 20\} +$
 $\max^2\{0, -x_1\} + \max^2\{0, -x_2\})$
(c) $\min 100 - (x_1 - 3)^2 - (x_2)^4 + 19x_3 +$
 $\mu(\max^2\{0, 35 - 5(x_1 - 1)^2 - 30(x_2 - 2)^2\} +$
 $|60x_2 + 39x_3 - 159|^2 + \max^2\{0, -x_1\} +$
 $\max^2\{0, -x_2\} + \max^2\{0, -x_3\})$

17-36 (a) $\min 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2 +$
 $\mu(\max\{0, (x_1)^2 + (x_2)^2 - 4\} + \max\{0, x_1 - 2\} +$
 $\max\{0, -x_1\} + \max\{0, -x_2\})$ **(b)** original
 model was convex program and convexity is
 preserved by the chosen penalty functions

(c) no (d) yes (e) penalty multipliers should start relatively low and increase slowly (f) $\mathbf{x}^{(0)} = (3, 5)$; with $\mu = .5$, $\mathbf{x}^{(1)} = (3.229, 4.646)$; with $\mu = 1$, $\mathbf{x}^{(2)} = (2.457, 3.743)$; with $\mu = 2$, $\mathbf{x}^{(3)} = (1.841, 2.730)$; with $\mu = 4$, $\mathbf{x}^{(4)} = (1.147, 1.762)$; with $\mu = 8$, $\mathbf{x}^{(5)} = (1.088, 1.678)$

17-37 (a) $\min 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2 + \mu(\max^2\{0, (x_1)^2 + (x_2)^2 - 4\} + \max^2\{0, x_1 - 2\} + \max^2\{0, -x_1\} + \max^2\{0, -x_2\})$ (b) original model was convex program and convexity preserved by the chosen penalty functions (c) yes (d) no (e) penalty multipliers should start relatively low and increase slowly (f) $\mathbf{x}^{(0)} = (3, 5)$; with $\mu = .5$, $\mathbf{x}^{(1)} = (1.449, 2.190)$; with $\mu = 1$, $\mathbf{x}^{(2)} = (1.308, 1.991)$; with $\mu = 2$, $\mathbf{x}^{(3)} = (1.214, 1.858)$; with $\mu = 4$, $\mathbf{x}^{(4)} = (1.157, 1.777)$; with $\mu = 8$, $\mathbf{x}^{(5)} = (1.124, 1.730)$

17-40 (a) not applicable due to equality constraint (c) $\max x_1 + 6/x_1 + 5(x_2)^2 + \mu(\ln(35 - 4x_1 - 6x_2) + \ln(x_1 - 5) + \ln(x_2))$ (e) not applicable due to equality constraint (f) not applicable due to equality constraint

17-41 (a) not applicable due to equality constraint (c) $\max x_1 + 6/x_1 + 5(x_2)^2 - \mu(1/(35 - 4x_1 - 6x_2) + 1/(x_1 - 5) + 1/(x_2))$ (e) not applicable due to equality constraint

17-42 (a) $\min 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2 - \mu(\ln(4 - (x_1)^2 - (x_2)^2) + \ln(2 - x_1) + \ln(x_1) + \ln(x_2))$ (b) original model was convex program and convexity preserved by the chosen barrier functions for $\mathbf{x} > \mathbf{0}$ (c) yes (d) no (e) barrier multipliers should start relatively high and decrease to 0 (f) $\mathbf{x}^{(0)} = (3, 5)$; with $\mu = 2$, $\mathbf{x}^{(1)} = (0.991, 1.615)$; with $\mu = 1/2$, $\mathbf{x}^{(2)} = (1.059, 1.663)$; with $\mu = 1/8$, $\mathbf{x}^{(3)} = (1.081, 1.674)$; with $\mu = 1/32$, $\mathbf{x}^{(4)} = (1.086, 1.677)$

17-43 (a) $\min 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2 + \mu(1/(4 - (x_1)^2 - (x_2)^2) + 1/(2 - x_1) + 1/(x_1) + 1/(x_2))$ (b) original model was convex program and convexity preserved by the chosen barrier functions for $\mathbf{x} > \mathbf{0}$ (c) yes (d) no (e) barrier multipliers should start relatively high and decrease to 0 (f) $\mathbf{x}^{(0)} = (3, 5)$; with $\mu = 2$, $\mathbf{x}^{(1)} = (0.978, 1.553)$; with $\mu = 1/2$, $\mathbf{x}^{(2)} = (1.031, 1.614)$; with $\mu = 1/8$, $\mathbf{x}^{(3)} = (1.061, 1.645)$; with $\mu = 1/32$, $\mathbf{x}^{(4)} = (1.075, 1.661)$

17-46 (a) $\min (x_1 - 8)^2 + 2(x_2 - 4)^2$, s.t. $2x_1 + 8x_2 + x_3 = 16$, $x_1 + x_4 = 7$, $x_1, x_2, x_3, x_4 \geq 0$ (b) columns linearly independent (c) N,B,S,B (d) $\mathbf{r} = (-13, 0, 1.5, 0)$ (e) $\Delta \mathbf{x} = (13, -3.0625, -1.5, -13)$ (f) $\lambda_{\max} = .32653$, $\mathbf{x}^{(1)} = (4.2449, 0, 7.5102, 2.7551)$ (g) basic x_2 dropped to 0; new basis $\{x_3, x_4\}$ or $\{x_1, x_4\}$

17-47 (a) $\mathbf{x}^{(0)} = (0, 1, 8, 7)$, $\mathbf{x}^{(1)} = (4.2449, 0, 7.5102, 2.7551)$; then either $\mathbf{x}^{(2)} = (4.6393, .8402, 0, 2.3607)$, $\mathbf{x}^* = \mathbf{x}^{(3)} = (6.2222, .4444, 0, .7778)$, or $\mathbf{x}^{(2)} = (7, 0, 2, 0)$, $\mathbf{x}^{(3)} = (7, .25, 0, 0)$, $\mathbf{x}^* = \mathbf{x}^{(4)} = (6.2222, .4444, 0, .7778)$

$$\mathbf{17-50 (a)} \mathbf{Q} = \begin{pmatrix} 6 & -3 & 0 \\ -3 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix},$$

$$\mathbf{c} = \begin{pmatrix} 5 \\ 15 \\ -16 \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 1 & 3 & -2 \\ 3 & -1 & 1 \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad \mathbf{(b)} -12x_1 + 6x_2 + 1v_1 + 3v_2 = 5,$$

$$6x_1 - 4x_2 + 3v_1 - 1v_2 = 15, \\ -8x_3 - 2v_1 + 1v_2 = -16, 1x_1 + 3x_2 - 2x_3 = 2, \\ 3x_1 - 1x_2 + 1x_3 = 3 \quad \mathbf{(c)} x_1^* = x_2^* = x_3^* = 1, \\ v_1^* = 5, v_2^* = 2$$

$$\mathbf{17-52 (a)} c_0 = 96, \mathbf{c} = (-16, -16),$$

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \mathbf{a}^{(1)} = (2, 8), \mathbf{a}^{(2)} = (1, 0),$$

$$\mathbf{a}^{(3)} = (1, 0), \mathbf{a}^{(4)} = (0, 1), b_1 = 16, b_2 = 7, \\ b_3 = 0, b_4 = 0, G = \{3, 4\}, L = \{1, 2\}, E = \emptyset \\ \mathbf{(b)} -2\Delta x_1 + 1v_3 = -16, -4\Delta x_2 = -12, \\ 1\Delta x_1 = 0; \text{solution } \Delta x_1 = 0, \Delta x_2 = 3, v_3 = -16 \\ \mathbf{(c)} \lambda = 1/3, \mathbf{x}^{(1)} = (0, 2)$$

$$\mathbf{(d)} -2\Delta x_1 + 2v_1 + 1v_3 = -16, \\ -4\Delta x_2 + 8v_1 = -8, 2\Delta x_1 + 8\Delta x_2 = 0, \Delta x_1 = 0; \\ \text{solution } \Delta x_1 = 0, \Delta x_2 = 0, v_1 = -1, v_3 = -14; \\ \Delta \mathbf{x} = \mathbf{0} \text{ implies no further progress (e) drop } \\ i = 3, x_1 \geq 0 \quad \mathbf{(f)} \mathbf{x}^{(2)} = (0, 2), \\ \mathbf{x}^{(3)} = \mathbf{x}^* = (6.222, 0.444)$$

$$\mathbf{17-54 (a)} \min 1 + x_{1,1}/3 + 4x_{1,2}/3 + 4x_{2,2}, \\ \text{s.t. } 2x_{1,1} + 2x_{1,2} + 1x_{2,1} + 1x_{2,2} \geq 2, \\ 4 + 28x_{1,1} + 112x_{1,2} - 18x_{2,1} - 54x_{2,2} \leq 25, \\ 0 \leq x_{1,1} \leq 1, 0 \leq x_{1,2} \leq 2, 0 \leq x_{2,1} \leq 2, \\ 0 \leq x_{2,2} \leq 2$$

$$\mathbf{17-55 (b)} \text{unconstrained minimum is feasible (c) } \min 18 - 8x_1 + 4x_2, \text{s.t.} \\ 0 \leq x_1 + x_2 \leq 6, 0 \leq x_1 \leq 2, 0 \leq x_2 \leq 4$$

(d) $x_1^* = 2, x_2^* = 0$; sequence correct **(g)** $x_1^* = 0, x_2^* = 4$; sequence incorrect

17-56 (a) $\min 3e^{-.5z_1} + e^{z_1+z_2} +$

$10e^{-3z_3}$, s.t. $.5e^{z_1+z_2} - 2^{z_3} \leq 1,$

$.167e^{z_1} + .25e^{.4z_1+z_2} + .0833e^{z_3} \leq 1,$

z URS **(b)** $\mathbf{z}^* = (1.360, -31.1, 1.433),$

$\mathbf{x}^* = (3.898, 3 \times 10^{-14}, 4.191)$ **(c)** $\max \delta_1$

$\ln(3/\delta_1) + \delta_2 \ln(1/\delta_2) + \delta_3 \ln(10/\delta_3) +$

$\delta_4 \ln(.5/\delta_4) + \delta_5 \ln(.167/\delta_5) + \delta_6 \ln(.25/\delta_6) +$

$\delta_7 \ln(.0833/\delta_7) - v_1 \ln(-v_1) - v_2 \ln(-v_2),$

s.t. $-.5\delta_1 + \delta_2 + \delta_4 + \delta_5 + .4\delta_6 = 0,$

$\delta_2 + \delta_4 + \delta_6 = 0, -3\delta_3 - 2\delta_4 + \delta_7 = 0,$

$\delta_1 + \delta_2 + \delta_3 = 1, \delta_4 = -v_1, \delta_5 + \delta_6 + \delta_7 = -v_2,$

$\delta_1, \dots, \delta_7 \geq 0, v_1, v_2 \leq 0$ **(d)** 4

(e) $\delta^* = (0, 0, 1, 0, 0, 0, 3), \mathbf{v}^* = (0, -3),$

\mathbf{x}^* as in (b)

17-59 (a) Using $\mathbf{x} \triangleq (x_1, x_2), \mathbf{v} \triangleq (v_1, v_2, v_3, v_4),$

$f(\mathbf{x}) \triangleq 2(x_1 - 3)^2 - x_1x_2 + (x_2 - 5)^2,$

$g_1(\mathbf{x}) \triangleq (x_1)^2 + (x_2)^2 - 4, g_2(\mathbf{x}) \triangleq x_1 - 2,$

$g_3(\mathbf{x}) \triangleq -x_1, g_4(\mathbf{x}) \triangleq -x_2,$

$L(\mathbf{x}, \mathbf{v}) \triangleq f(\mathbf{x}) + \sum_{i=1}^4 v_i g_i(\mathbf{x})$ **(b)** $L(\mathbf{x}, \mathbf{v}) +$

$\nabla L(\mathbf{x}, \mathbf{v}) \Delta \mathbf{x} + 1/2 \Delta \mathbf{x} \nabla^2 L(\mathbf{x}, \mathbf{v}) \Delta \mathbf{x}$

(c) $\min \nabla f(\mathbf{x}) \Delta \mathbf{x} + 1/2 \Delta \mathbf{x} [\nabla^2 f(\mathbf{x}) +$

$\sum_{i=1}^4 v_i \nabla^2 g_i(\mathbf{x})] \Delta \mathbf{x}$ subject to

$g_i(\mathbf{x}) + \nabla g_i(\mathbf{x}) \Delta \mathbf{x} \leq 0, i = 1, \dots, 4$ **(d)**

$\min (-9, -9) \Delta \mathbf{x} + 1/2 \Delta \mathbf{x} \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix} \Delta \mathbf{x},$ sub-

ject to $-2 + (2, 2) \Delta \mathbf{x} \leq 0, -1 + (1, 0) \Delta \mathbf{x} \leq 0,$

$-1 + (-1, 0) \Delta \mathbf{x}, -1 + (0, -1) \Delta \mathbf{x} \leq 0.$

Index

- 0-1 variables (*see* discrete variables)
- 1-dimensional search 924–935
 - bracketing 929–932
 - golden section search 925–929
 - quadratic fit 932–935
- 3-point patterns 929–932
 - Algorithm 16B 931
- AA Crew Scheduling Application 673–674
- absolute value objectives (*see* minimum deviation objectives)
- actions of Markov Decision Processes 542
- active constraints 116, 216–217
 - conditions for feasible directions 118–119
- active partial solutions 754
- active set methods 1055–1061
 - Algorithm 17D 1059–1060
 - KKT conditions 1055
 - step sizes 1056–1057
 - updating the active set 1058–1059
- activities implicit pricing 309–310
- activities in project management 520
- activities interpretation of decision variables 292
- activity duration 520
- acyclic digraph shortest paths 515–519
 - Algorithm 9D 518
 - computational order 518
- acyclic digraphs project network case 528
- adjacent extreme points 216–217
- affine scaling 396–401
 - diagonal matrix formalization 396–398
 - inverse 398
 - standard form for LPs 399
- affine scaling search 402–409
 - Algorithm 7A 407
 - directions 403–404
 - step sizes 403–406
 - termination optimal 407
- affinely independent sets of points 794
 - versus linearly independent 795
- Agrico Chemical Application 566
- all or nothing constraints 656–658
- allocation decision variables 145
- allocation models 144–147
 - allocation decision variables 145
- alternative optimal solutions 34–36
- AMPL modeling language 65–72, 320–322
 - data section 67
 - dual and sensitivity 320–322
 - indexing and summation 67
 - model section 67
 - model vs. data sections 67
 - nonlinear and integer programs 70
 - syntax 66–67
 - variable types 70
- anti-cycling rules 260
- approximate optimization (*see* heuristic optimization)
- arcs 479, 557
 - capacities 559
 - costs 559
 - flows 559
- artificial network flow model 563–564
- artificial variables 129–130
- assignment constraints 608
- assignment problems 609 (*see also* each model class)
 - generalized 680–683
 - linear (*see* Hungarian algorithm)
 - linear 607–610
 - matching 683–684
 - quadratic 677–680
- asymmetric traveling salesman problems (*see* traveling salesman problems)

- backwards dynamic programming 534–535
- balance constraints 158–169, 559
 - in operations planning 158–159
 - in time-phased models 168–169
 - network flow 559
- balance of flow 559
- Bank Three Application 437–438
- barrier functions 1035
- barrier methods
- barrier methods (*see also* Newton step barrier search; primal-dual barrier search)
 - LPs 408–420
 - NLPs 1034–1038
 - sequential unconstrained barrier technique 1037–1038
- barrier multipliers 409, 1033
 - management of 417–419
- barrier objective function 409, 1031
- bases finite number 258
- basic feasible solutions 225 (*see also* basic solutions)
 - equivalence with extreme points 225–226
 - minimum cost flows 596–597
 - via two-phase simplex 250
- basic solutions 219–227 (*see also* basic feasible solutions)
 - existence 221–223
 - in lower- and upper-bounded form 274–275
 - minimum cost flows 596–597
- basic variables 219, 836
- basis inverse 261
 - product representation 264–266
- basis matrix 260
- basis 221
- basis connection to linear independence 222
- basis in reduced gradient 1040–1041
- basis minimum cost flows 594–595
- basis update in simplex 234–235
- Bay Ferry Multicommodity Flow Application 626
- Beer Belge Location Allocation Application 987–988
- Bellman, R.E. 495
- Bellman-Ford shortest paths one to all 494–501
 - Algorithm 9A 496
 - computational order 499
 - negative dicycles 500–501
- Benders decomposition 842–848
 - Algorithm 13E 846
 - attractive models 843
 - partial master problem 844–845
 - primal and dual subproblems 844
 - strategy 844–845
- Benders, J.F. 842
- benefits as LP objectives 288
- best first search 772
 - versus depth first 772–777
 - versus depth forward best back 772–777
- beta distribution 920
- Bethlehem Ingot Mold Application 53
- BFGS formula 966
 - approximation to Hessian inverse 972
- BFGS search 966–972
 - Algorithm 16F 968
- big-M constants 135, 749–751
- big-M method 135–138
 - artificial model 135
 - objective function 135
 - outcomes 136
- bill of materials 156, 159, 187
- binary variables (*see* discrete variables)
- binding constraints (*see* active constraints)
- Bison Boosters Application 734–735
- blending models 147–151, 989–992
 - composition constraints 148–149
 - ingredient decision variables 148
 - linear program case 147–151
 - nonlinear refining 990–993
- boundary points 203–205, 252
 - optimal in Llinear programs 207
- bounds on dual values from primal 345
- bounds on primal values from dual 345
- bounds from relaxations 739–742
- bracketing (*see* 3-point patterns)
- branch and bound 751–764
 - candidate problems 757–778
 - enumeration sequences 772–777
 - error bounds 770–772
 - global optimality 756
 - incumbent solutions 756–757
 - LP-based 760–764
 - partial solutions 752–753
 - stopping early 770–772
 - stopping 755
 - terminating by bound 759
 - terminating by parent bound 769–770

- terminating by solving 759
- terminating infeasible 759
- trees 753
- tree search 753–756
- branch and cut 779–784, 874 (*see also*
 - branch and bound)
 - Algorithm 12B 779
 - valid inequalities 777–782
- branch and price 819–822
 - Algorithm 13B 820
- branch and price strategy 819–820
- Broyden, C. 966
- budget constraints 663–664
- Building Evacuation Maximum Flow
 - Application 619–620
- CAM Application linear assignment
 - problems 676
- Canadian Department of Transportation
 - (*see* CDOT application)
- Canadian Forest Products Limited
 - Application (*see* CFLP application)
- candidate problems 757–758
- capital budgeting problems 662–666
 - budget constraints 663–664
 - dependent projects 665
 - mutually exclusive choices 664–665
- cash flow models 166–170, 630–631
- CDOT generalized assignment
 - Application 681, 823
- ceiling least integer $\geq q$ 745
 - denoted $\lceil q \rceil$ 745
- CFPL Application generalized
 - assignment 154–155
- chains 570
 - versus paths 570
- child of a tree node 768
- classes of optimization models 65
- Clever Clyde Application 244
- closed form solutions 11
- Cofferdam Design Application 1006–1007
- column generation methods 674–677,
 - 814–821 (*see also* delayed column
 - generation)
- combinatorial optimization (*see* discrete
 - optimization)
- complementary dual basic solution 355–356
- complementary slackness
 - dual LP 312–313, 349–351
 - equivalence to primal vs. dual value
 - difference 349–350
 - primal LP 311–312
 - primal NLP 1019–1020
 - role in KKT conditions 349–351
- complete enumeration (*see* total enumeration)
- completion time 706
- completions of partial solutions 752–753
- complexity theory
 - class Undecidable 865
 - classes PTime and NP-Hard 869
 - computational orders 857–858
 - decision problems 862–863
 - guaranteed performance heuristics 872–874
 - improved enumerative algorithms
 - (*see* branch and bound)
 - improved enumerative algorithms
 - (*see* branch and cut)
 - length of instance encoding 859
 - linear programming 430
 - linear programming length of input 428
 - linear programming polynomial order 430
 - linear programming simplex worst
 - cases 429
 - minimum cost network flows 589–591
 - nondeterministic polynomial solution and
 - class NP 864
 - optimization vs. threshold vs. feasibility
 - problems 863
 - $P \neq NP$ conjecture and its
 - implications 869–871
 - polynomial reduction among
 - problems 866–867
 - polynomially solvability and class P versus
 - PTime 864–866
 - polynomially solvability and class P 864
 - polynomial-time standard for
 - solvability 861–862
 - problems vs. instances 855
 - pseudo-polynomial algorithms 871–872
 - recognizing tractability 856
 - tractable special cases 871
 - worst-case standard 857
- composition constraints 148
- compositions of convex/concave
 - functions 952
- concave functions 948–950
 - sufficient for global maxima 950–951
 - tests 951–954

- conflict constraints 704–705
- conservation of flow (*see* balance of flow)
- constrained nonlinear programming 987–995
 - active set methods 1055–1061
 - barrier methods 1034–1038
 - blending models 990–993
 - convex programs 998–1001
 - differentiable 1019
 - engineering design models 992–996
 - geometric programs 1006–1008, 1073–1082
 - Karush-Kuhn-Tucker conditions 1020–1021
 - Lagrange multiplier methods 1011–1019
 - linearly constrained 989–992
 - location allocation problems 988–990
 - penalty methods 1028–1034
 - quadratic programs 1003–1006, 847–856
 - reduced gradient methods 1038–1051
 - separable programs 1001–1003, 1065–1072
 - sequential quadratic programming 1061–1065
- constraint coefficients 184
 - interpretation 290–291
- constraint matrix 184
- constraint qualifications 1027
- constraint relaxations 737–738 (*see also* relaxations)
- constraints 4, 24–25
 - adding and dropping 308–309, 350–352
 - interpretation 288–290
 - relaxing versus tightening constraints 293
 - sensitivity analysis 293–303, 324–326
 - sensitivity computer outputs 320–322
- construction costs (*see* fixed charges)
- constructive search 879–886
 - Algorithm 15A 880
 - greedy rules 885–889
- continuous improving search (*see also* improving search)
 - Algorithm 3A 106
- continuous optimization model, 56
- continuous relaxations 737–738
- continuous variables 54
 - versus discrete 54, 144
- contours of objective functions 32–33
- contraction step in Nelder-Mead 977
- convergence of algorithms 386
- convex feasible sets 121–122
 - linearly constrained 126–127
 - nonlinearly constrained 998
- convex functions 948–950
 - sufficient for global minima 950–951
 - tests 951–954
- convex hull of feasible solutions 789–791
 - affine independence 794
 - dimension 793–797
 - dimension affine independence
 - characterization 794–795
 - faces 793
 - facets 793
 - finite optimum at extreme point of 792
 - polyhedral form 791
 - supporting inequalities 793
 - valid equalities 796
- convex programming 998–1001
 - global optima 1001
 - separable 1071–1072
 - sufficiency of KKT conditions 1027
- Cook, W. 861
- corner points 203–205
- costs as LP objectives 288
- covariances of returns 1005
- covering constraints 667
 - in shift scheduling 164–165
- CPLEX solver 67
- CPM (*see* critical path methods)
- crew pairing 673
- criterion functions (*see* objective functions)
- critical path methods 520–528
 - Algorithm 9E 525
 - computational order 528
 - versus longest paths 523
- critical paths 523
- crossover genetic algorithms 902–903
- curve fitting 916–919
 - linear versus nonlinear 917–918
- Custom Computer Curve Fitting
 - Application 916
- Custom Metalworking Job Shop
 - Application 711–714
- cut sets 619
- cutting plane theory 788–797
- cutting planes 777
- cycle canceling search 582–591
 - Algorithm 10B 587
 - computational order 589–591
 - feasible directions 584–585
 - improving directions 583–584
 - minimum mean length 589–591

- residual digraphs 583–584
 - with Floyd-Warshall algorithm 586–587
- cycle direction search 580–582
 - Algorithm 10A 582
 - network simplex 591–601
 - starting feasible solutions 563–564
 - step sizes 577–578
- cycle directions 571–572
 - feasible 574–575
 - from Floyd-Warshall algorithm 586–587
 - from residual digraphs 584–585
 - improving 576–577
 - preserving flow balance 573
 - step sizes 577–578
 - sufficiency for optimality 578–580
- cycles 570
- cycles versus dicycles 571
- cycling with degeneracy 259

- Dantzig, G. 836
- Dantzig-Wolfe decomposition 836–842
 - Algorithm 13D 841
 - attractive applications 836–837
 - partial master problem 836–838
 - reformulation with extreme-points and directions 838–839
 - strategy 836–840
- DClub Location Application 91
- Decision Problem** complexity
 - class 862–863
- decision problems in OR models 3
- decision variables 5, 24
 - activity interpretation 288
 - adding and dropping 303
 - implicit pricing 309–310
- decisions in OR models 4
- decisions in dynamic programs 530
- decomposition methods 811–848
 - Benders decomposition 842–848
 - Dantzig-Wolfe decomposition 836–842
 - delayed column generation 812–819
 - Lagrangian relaxation 822–836
- decrease arc 583
- decreasing returns to scale 52
- deficiency variables 455–456
- defined to be equals by definition 5
 - denoted \triangleq 5
- deflection matrices 964–965
- degeneracy 253
 - ambiguity of rates of change in optimal values 329, 335–338
 - cycling 259
 - difficulties for simplex 255–257
- delayed column generation 812–819
 - Algorithm 13A 816
 - attractive model targets 813
 - generating eligible columns to enter 817–818
 - in set partitioning 674–675
 - partial master problem 815
 - strategy 811–815
- demand nodes (*see* sinks)
- dependent projects 665
- depth first search
 - acyclic digraphs 515–516
 - branch and bound 755–756
 - versus best first 772–777
 - versus depth forward best back 772–777
- depth forward best back search 772
- derivatives denoted df/dx 110
- derivatives first partial denoted $\partial f/\partial x_j$
 - gradient as vector of 106
 - gradient as vector of denoted ∇f 106
 - rates of change for single variable 111
- derivatives rates of change for single variable 110
- derivatives second partial denoted $\partial^2 f/\partial x_i \partial x_j$
 - Hessian as matrix of 106
 - Hessian as matrix of denoted $\nabla^2 f$ 106
 - rates of change for two variables 938
- descriptive models 12
 - versus prescriptive 12–14
- determinants of matrices 223
 - checking convexity/concavity 952
- deterministic models 16
 - versus stochastic 18–19
- dicycles 488, 573 (*see also* negative dicycles)
 - versus cycles 571
- differentiable nonlinear programming 1021
 - (*see also* constrained nonlinear programming)
- digraphs 483, 557
- Dijkstra shortest paths one to all nonnegative 509–515
 - Algorithm 9C 509
 - computational order 515
 - permanent and temporary nodes 509
- Dijkstra, E.W. 509
- dimension (*see* vectors)

- directed graphs (*see* digraphs)
- directing a graph 483–484
- direction change in scalar x 98
 - denoted Δx ;98
- direction change in vector \mathbf{x} 98
 - denoted $\Delta \mathbf{x}$;98
- directions (*see* move directions)
- direction-step paradigm 98–100
- discrete dynamic programming (*see* dynamic programming)
- discrete improving search 888–895 (*see also* improving search)
 - Algorithm 15B 887
 - move sets 887–892
 - nonimproving moves 894–895
 - pairwise interchange 890–891
 - single complement move sets 892
- discrete optimization 56
 - assignment problems 675
 - branch and bound search 751–764
 - branch and cut search 777–778
 - capital budgeting models 662–666
 - cutting planes 777–778, 788–797
 - facility location 695–699
 - fixed charge problems 658–660
 - generalized assignment problems 680–683
 - job shop scheduling 710–713
 - knapsack problems 661–662
 - linear assignment problems 607–610, 676–677
 - lumpy linear programming 655–660
 - matching 683–684
 - modeling as continuous 157
 - network design 699–702
 - quadratic assignment problems 677–680
 - routing 689–690, 693–694
 - set packing, covering, partitioning 666–675
 - single processor scheduling 702–710
 - total enumeration 731–734
 - traveling salesman problems 679–693
- discrete variables 54
 - versus continuous 54, 144
- disjunctive constraints 704
 - big- M constants 749–751
- disjunctive variables 704
- dot product of vectors 90–91
- dual and primal LP standard forms 352
 - partitioned 354–355
- dual complementary slackness 312–313, 349–352
- dual constraints 309–310
- dual linear programs 304
- dual of the dual 317–318, 345
- dual simplex search 359–365
 - Algorithm 6A 359
 - convenience for start or restart 360
 - directions 361
 - solution update 362–363
 - step size 361–362
 - strategy for LP optimality 359–360
- dual variables 304
 - price interpretation 308–309
 - variable types 304–307
- duality
 - bounds on dual values 345
 - bounds primal value 345–346
 - by-product optimum 313
 - complementary slackness 311–313, 349–351
 - equality with primal 310–311
 - formulating 313–318
 - formulating nonnegative primal variables 314–315
 - formulating nonpositive and unrestricted primal variables 316–317
 - geometric programming 1077–1082
 - infeasible duals 347–349
 - infeasible primals 347–349
 - primal or dual finite other finite 351
 - strong 310, 351
 - unbounded duals 347–349
 - unbounded primals 347–349
 - weak 345
- due dates 706
- DuPage Land Use Planning Application 61
- Dyanmometer Ring Design
 - Application 439–440
- dynamic models (*see* time-phased models)
- dynamic programming 485
 - approach to shortest paths 485–490
 - backward solution 534–535
 - computational order 537
 - digraph for 530–531
 - functional equations 489, 532
 - functional notation 486–487
 - knapsack problems 538–540
 - Markov Decision Processes 541–545
 - modeling 532–532
 - multiple problem solution feature 537
 - principle of optimality 489–490

- solving integer programs 537–540
 - stages 533
 - states 529
- early start schedules 523
 - Algorithm 9E 525
 - versus longest paths 523
- early start times 523
- economic order quantity 9
- edges of a graph 479
- edges of the LP-feasible space 217
- Edmonds, J. 861
- efficient frontier 445–448
 - constructing 446–447
- efficient points 443–445
 - with goal programs 462–463
 - with preemptive optimization 451
 - with weighted sums 453
- efficient solutions (*see* efficient points)
- Electoral Vote Knapsack Application 538
- elite solutions genetic algorithms 903
- ellipsoid method 430
- E-Mart Application 51
- EMS Location Planning Application 666–672
- engineering design models 431, 992–996, 1007–1010, 1029–1030
- EOQ (*see* economic order quantity)
- equal returns to scale 52
- error bounds enumerative search 770–772
- Euclidean distance 176
 - versus rectilinear 176
- evolutionary metaheuristics 902–906
- exact optimal solutions 15
 - versus heuristic 16
- exact penalty functions 1032–1033
- expansion step in Nelder-Mead 977
- exponential growth 733–734
 - of total enumeration 733–734
- extensive form stochastic programming 184
- extreme points 203–205
 - adjacent 216–217
 - defined by active constraints 216
 - equivalence with basic feasible solutions 225–226
 - optimal in linear programs 207–208
- face-inducing inequalities 794
- faces of polyhedral sets 793
 - affine independence characterization 795
- facet-inducing inequalities 794
- facets of polyhedral sets 793
 - affine independence characterization 795
- facility layout models 678
- facility location problems 695–699
 - big- M constants 747–748
 - switching constraints 697
- families of valid inequalities 782–788
- fathoming (*see* termination)
- feasible completions 752–753
- feasible directions 102–104
 - active constraint conditions 118–119
 - linearly constrained NLPs 1040
 - network cycle 574–575
 - versus KKT conditions 1023–1024
- feasible models 29, 35–37
 - concluding with two-phase method 132
- feasible region (*see* feasible set)
- feasible set 27
 - convex 121–122
- feasible solutions 7
- feasible space (*see* feasible set)
- Filter Tuning Application 1039–1040
- finish node project scheduling 521
- first derivatives 110
- first partial derivatives (*see* gradients)
- first-order necessary optimality conditions 942
- first-order Taylor approximation 939–941
- fixed charge network flow (*see* network design problems)
- fixed charge problems 586–558
 - big- M constants 747–748
 - facilities location 695–699
 - fixed charge variables 658
 - network design 699–702
 - switching constraints 658
- fixed charges 658
 - versus variable 658
- fixed time horizons 170
 - versus infinite 170–171
- Fletcher, R. 966
- floor greatest integer $\leq q$ 745
 - denoted $\lfloor q \rfloor$ 745
- flow balance constraints 561 (*see also* balance constraints)
- flow time 706
- flow-distance 678
- flows with gains and losses (*see* gain/loss flows)

- Floyd, R.W. 501
- Floyd-Warshall shortest paths all to
 all 501–509
 Algorithm 9B 502
 computational order 502
 detecting negative dicycles 507–509
- FLP (*see* facilities location problems)
- for every for all
 denoted \forall
- Ford, L.R. Jr. 495
- Forest Service Allocation Application 144
- forward arc 571
- fractional parts 782
- fractional variable branching rule 761–762
- Frannie's Firewood Application 386
- free variables 752–753
- frequency histogram 17
- functional equations 489–490
 all to all shortest path 425–426
 knapsack problems 539
 one to all shortest path 489–490
 Wagner-Whitin Lot Sizing Application 532
- functional notation dynamic
 programming 486
- gain/loss flows 630–632
 tractability 632
- GAP (*see* generalized assignment problems)
- Gaussian elimination 239
- generalized assignment problems 680–683
 Lagrangian relaxations 681, 823
 tractability 685
- generalized reduced gradient methods
 1050–1051
- generations in genetic algorithms 902
- generic interpretation of LPs 288–292
- genetic algorithms 902–906
 Algorithm 15E 904
 crossover 902–903
 elite solutions 903
 generations 902
 immigrant solutions 903
 mutations 903
 populations 902
 random keys 904–905
 solution encoding 904–906
- geometric programming 1008–1010,
 1075–1084 (*see also* posynomial
 geometric programming)
- gets value variable x is assigned a value
 q denoted $x := q$;68
 q denoted $x \leftarrow q$;98
- Global Backpack (GB) Application 837
- global maxima 95 (*see also* global optima)
 concave sufficient 950–951
- global minima 95 (*see also* global optima)
 convex sufficient 950–951
- global optima 95
 from branch and bound 756
 from convex programs 1001
 from improving search 120
 versus local optima 95–97
- goal levels 454
- goal programs 458
 deficiency variables 455–456
 efficient points 462–463
 minimize deficiency objective 457–458
 modeling soft constraints 455–457
 preemptive 459–461
 versus other multiobjective 461–462
- golden ratio 926
- golden section search 925–930
 Algorithm 16A 927
- Goldfarb, D. 966
- Gomory cutting planes mixed-integer
 case 785–787
- Gomory cutting planes pure integer
 case 782–784
- Gomory, R.E. 782–778
- GP (*see* geometric programming)
- gradient norm 956
- gradient search 955–959
 Algorithm 16D 955
 zigzagging and convergence 959
- gradients 109–110
 as move directions 112–113, 252–253, 278
 conditions for improving directions 112–114
 graphic interpretation 111
- graphic solution
 constraints 27–30
 feasible solutions 28–29
 infeasible models 35–37
 objective functions and their contours 30–33
 optimal solutions 33–35
 unbounded models 37–39
- graphs 479 (*see also* digraphs; undirected
 graphs)
- greedy heuristics 880–885

- hard constraints 455
- Hazardous Waste Disposal Application 440–441
- Heart Guardian Facilities Location Application 843
- Hessian matrices 937–939
- heuristic optimal solutions 16, 678
 - from local optimum 98
 - versus exact 16
- heuristic optimization
 - constructive search 879–887
 - genetic algorithms 902–906
 - improving search 98, 678–683
 - metaheuristics 893–902
 - multistart 892–893
 - simulated annealing 898–902
 - tabu search 893–887
 - truncated branch and bound 770–772
- Highway Patrol Application 172
- hillclimbing (*see* improving search)
- holding arcs 567
- how far (*see* step sizes)
- Hungarian assignment algorithm 611–618
 - Algorithm 10D 612
 - computational order 617
 - dual solution update 616–617
 - equality subgraph 614–616
 - primal solution growth 617
 - primal-dual strategy 611–613
- identity matrix 261
 - I notation 228
- IFS (*see* Institutional Foods)
- ILP (*see* integer linear programming)
- immigrant solutions genetic algorithms 903
- implicit prices (*see* dual variables)
- improving directions 100–101
 - deflection matrix conditions 966
 - from gradients 112–113
 - gradient conditions for 112–114
 - network cycle 576–577
 - versus KKT conditions 1023–1024
- improving search 93–95, 735–736
 - active constraint conditions for feasibility 118–119
 - blocking constraints 127–128
 - cases with local optima global
 - cases with local optima global linear over convex 128
 - cases with local optima global linear programs 128
 - continuous Algorithm 3A 106
 - detecting unboundedness 108–109
 - direction-step paradigm 98–100
 - discrete 886–893
 - feasible directions 102–104
 - heuristics 886–893
 - improving directions 100–101
 - initial solutions 129, 118–127
 - neighborhoods 94–95
 - step sizes 104–105
 - stopping with local optima 107
 - tractable when local optima are
 - global 120–129
 - with convex feasible sets 127–128
- IMRT Radiation Therapy Planning Application 812
- inactive constraints (*see* active constraints)
- increase arc 583
- incumbent solutions 756–757
 - as heuristic optima 770–772, 872
- indexing 40–45
 - for time phasing 168–169
 - in summations 42
 - making models large 45–46
 - of decision variables 40–41
 - of families of constraints 43–45
 - of symbolic parameters 41–42
- Indy Car Knapsack Application 661
- infeasible models 35–37
 - concluding from two-phase method 132–133
 - detecting with simplex 249–250
 - primal to dual relationships 347–349
 - proving with relaxations 737–738
- infinite time horizons 171
 - versus fixed 170–171
- ingredient decision variables 148
- initial costs (*see* fixed charges)
- INLP (*see* integer nonlinear programming)
- Institutional Food Services Cash Flow Application 167
- integer linear programming 57, 63
 - multiobjective 442
- integer nonlinear programming 57, 63
- integer programming 56
- integer programs (*see* discrete optimization)
- integer variables (*see* discrete variables)

- integrality property
 - lacking in gain/loss flows 632
 - lacking in multicommodity flows 629
 - minimum cost flows 601–602
- Intensity Modulated Radiation Therapy 812
 - beamlets 812
 - voxels 812
- interior point methods
 - for LP 385–428
 - affine scaling search 396–408
 - computational burden 402
 - log barrier search 408–420
 - Newton step barrier search 412–419
 - primal-dual (*see also* primal-dual search interior-point)
 - Algorithm 7C 424
 - strategy 422
 - projection on equality constraints 390–394
 - strategy 387–389
- interior point solutions 386
 - convenience for feasible directions 384
 - in linear program standard form 389
 - strictly interior 389
- interior points 203–205, 252
- inverse of a matrix 261–262
- IP (*see* integer programming)
- job shop scheduling 712–715 (*see also* scheduling)
 - conflict constraints 712–713
 - precedence constraints 712
- joint probability density function 919
- Karmarkar, N. 430
- Karp, R.P. 861
- Karush, W. 351
- Karush-Kuhn-Tucker conditions
 - active set methods 1055
 - necessary and sufficient for LPs 351–352
 - necessary and sufficient for LPs 351–352
 - necessity 1027–1028
 - NLPs 1020–1021
 - partitioned standard form LPs 354
 - proof of necessity for LPs 358–359
 - proof of sufficiency for LPs 351
 - quadratic programs 1053–1054
 - standard form LPs 353–354
 - sufficiency for convex programs 1027
 - versus improving feasible directions 1023–1027
- Khachiyan 430
- KI Routing Application 693–694
- KKT conditions (*see* Karush-Kuhn-Tucker conditions)
- KKT point 1020
- Klee-Minty perverse simplex instances 429
- knapsack problems 661–662
 - as dynamic programs 539–541
 - minimal cover valid inequalities 787–788
- Kuhn, H.W. 351
- Lagrange multiplier techniques 822–824, 1011–1019
 - all equality standard form 1011–1012
 - limitations to use 1018–1019
 - optimal solutions from stationary points 1014–1017
 - stationary point conditions 1013–1014
- Lagrange multipliers (*see also* dual variables)
 - interpretation 1017–1018
 - sign restrictions 1020
- Lagrangian duals 828–830
 - objective function concavity 832–833
 - subgradient search Algorithm 13C 833
- Lagrangian functions 1012–1013
- Lagrangian relaxations 822–824
 - bound versus LP relaxation 831
 - bounds on overall optimum 826
 - multipliers 822
 - solutions provably overall optimum 826
 - strategy 822–824
 - tractability 824–825
- large-scale optimization methods (*see* decomposition methods)
- late start schedules 526–527
- late start times 526
- lateness 706
- lead times 5
- learning curves 980
- left-hand sides interpretation 290
- lexicographic goal programming (*see* preemptive goal programming)
- lexicographic optimization (*see* preemptive optimization)
- LHS (*see* left-hand sides)
- likelihood functions 919–920
- line search (*see* 1-dimensional search)
- line search required in reduced gradient 1046–1047

- line segments 123
 - algebraic characterization 123–124
- linear assignment problems 607–610, 676–677
 - CAM Application 609
 - Hungarian Algorithm 10D 612
 - integrality 610
 - standard form 608
 - tractability 685
 - versus matching 683
- linear combinations 224 (*see also* linearly independent vectors; weighted sums)
- linear constraints 48, 107
 - conditions for feasible directions 118–119
 - convexity of feasible set 126–127
- linear equations 223–224
- linear functions 48
 - convex and concave 952
 - equal returns to scale 52
- linear objective functions 125
- linear program-based branch and bound (*see* LP-based branch and bound)
- linear programming 50, 64, 131 (*see also* duality)
 - relaxations 737–738
 - affine-scaled standard form 399
 - allocation models 144–147
 - blending models 147–151
 - degeneracy 253–257
 - dual bounds 345–346
 - formulation of duals 313–318
 - formulation of duals nonnegative primal variables 314–316
 - formulation of duals nonpositive and URS primal variables 316–317
 - generic interpretation 288–292
 - inequalities as supplies and demands 288–290
 - linearizable nonlinear objectives 171–178
 - lower- and upper-bounded simplex search 272–279
 - modeling integer quantities 157
 - multiobjective 60–62, 373–375
 - objectives as costs and benefits 288
 - operations planning models 152–161
 - primal versus dual 304
 - qualitative sensitivity 293–303
 - quantitative sensitivity 304–310
 - relaxations of ILPs 737–738
 - revised simplex search 260–272
 - shift scheduling and staff planning models 162–166
 - simplex search 227–237
 - standard forms 209–215, 238–239
 - testing infeasible 249–250
 - testing unbounded 252
 - time-phased models 166–171
 - tractability 203, 424
 - variables as activities 302
- linear programs feasible sets polyhedral 126
- linear regression (*see* curve fitting)
- linear Taylor approximation (*see* first-order Taylor approximation)
- linearly dependent vectors 224
 - impossible in a basis 222
 - relation to singularity 224
- linearly independent sets of points versus affinely independent 795
- linearly independent vectors 224
 - relation to nonsingularity 224
 - versus bases 221
- links (*see* edges)
- Littleville Application 477–478
- local improvement (*see* improving search)
- local maxima 95 (*see also* local optima)
- local minima 95
- local optima 95
 - caused by constraints 122
 - first-order necessary conditions 942
 - from improving search 95–98
 - KKT conditions 1024–1026
 - second-order necessary conditions 946
 - second-order sufficient conditions 946–947
 - single variable search 924–925
 - versus global optima 96–97
- local search (*see* improving search)
- location models 91, 678
- location-allocation models 988–990
- log barrier functions 409–410
- log barrier methods (*see* barrier methods)
- logistics curve 980
- longest path problems 519–520 (*see also* shortest path problems)
 - tractability 519–520
- lot sizing 529
- lower- and upper-bounded simplex search 272–279
 - Algorithm 5D 278
 - basic solutions 274–275
 - step sizes 276–277

- LP (*see* linear programming)
- LP-based branch and bound 762–766
 - (*see also* branch and bound)
 - Algorithm 12A 761
 - branching rules 761–762
- LP-feasible sets
 - edges 217
 - extreme points 216
 - polyhedral 126
- Luenberger, David G. 141
- lumpy linear programming 655–660
 - all or nothing constraints 656–658
 - fixed charge problems 658–660
- main constraints 25
- major iterations reduced gradient
 - search 1049–1050
- makespan (*see* maximum completion time)
- Mall Layout Quadratic Assignment
 - Application 678
- marginal prices (*see* dual variables)
- Marine Mobilization Transportation Problem
 - Application 606–607
- Markov Decision Processes 541–546
 - actions 543
 - dynamic programming solution 542
 - model elements 541–542
 - rewards 542
 - states 542
 - transitions 542
- matching problems 683–684
 - tractability 685
 - versus linear assignment 683
- material balance constraints (*see* balance constraints)
- mathematical model 1
- mathematical program general form 47
- mathematical programs 4
 - standard form 26, 46–47
- matrices 213–214
 - denoted by boldface upper case notation 213
 - inverses 261
 - inverses denoted \mathbf{M}^{-1} 261
 - multiplication 214–215
 - nonsymmetric 214
 - symmetric 214
 - transposes 214–215
 - transposes denoted \mathbf{M}^T 214–215
- maximal sets of independent vectors 223–224
- maximin objectives 173–174
 - linearizing 173–174
- maximum completion time 706
- maximum flow problems 618
 - Algorithm 10E 622
 - computational order 625
 - equality of max flow and min cut 624
 - flow augmenting paths 620–621
 - minimum cuts 619
 - network flow formulations 621
 - residual digraph 622–624
- maximum flow time 706
- maximum lateness 706
- maximum likelihood estimation 919–921
- maximum tardiness 706
- maximum/minimum spanning trees 633–639
 - Algorithm 10F 634
 - composition tree 635
 - computational order 638–639
 - equivalence of primal and dual 637
 - greedy algorithm strategy 633–634
 - ILP formulation 636
 - LP relaxation and dual 636
- maxisum objectives 173
- MDP (*see* Markov Decision Processes)
- Meade, R 974
- mean completion time 706
- mean flow time 706
- mean lateness 706
- mean tardiness 706
- metaheuristics discrete optimization
 - heuristics 893–902
- method of steepest ascent (*see* gradient search)
- minimal cover knapsack inequalities 787–788
- minimax objectives 173
 - linearizing 173–174
 - scheduling 706–708
- minimum cost flows 559–563
 - basic solutions 596–597
 - cycle canceling search 583–591
 - cycle direction search 580–582
 - cycle directions 571–572
 - formulation of shortest paths 647
 - integer optima 601–602
 - linear dependence of cycles 591–592
 - network simplex search 591–601
 - node-arc incidence matrices 568–570
 - residual digraphs 583–584
 - single commodity 629

- spanning tree bases 594–595
- standard form 559
- time-expanded models 565–568
- total supply equals total demand 562–563
- tractability versus LP 499–500
- tractability versus shortest paths 647
- minimum cut problems 619
- minimum deviation objectives 176
 - linear modeling 176
- minimum mean length dicycles (*see* Cycle Cancelling)
- minimum ratio rule (*see* step sizes)
- minisum objectives 173
- MINLP (*see* mixed-integer nonlinear programming)
- minor iterations reduced gradient search 1049–1050
- MINOS solver 70
- MIP (*see* mixed-integer programming)
- mixed-integer nonlinear programming (*see* integer nonlinear programming)
- mixed-integer programming 57 (*see also* discrete optimization)
- modeling languages vs. solvers 66
- Monte Carlo analysis (*see* stochastic simulation)
- Mortimer Middleman Application 2
- move directions 98, 735
 - BFGS 965–968
 - gradient search 955–959
 - gradient 112–113, 252–253, 278
 - Nelder-Mead 975–976
 - Newton step barrier 412–414
 - Newton’s method 960
 - reduced gradient 1044–1045
 - simplex cycle 597–598
 - simplex LP 228–230, 240–241
- move sets discrete improving search 887–892
- multicommodity flows 625–629
 - tractability 629
 - versus single commodity 626–627
- multidimensional knapsack problems (*see* capital budgeting problems)
- multiobjective optimization 60, 437–464
 - efficient frontier 445–448
 - efficient points 443–445
 - engineering design models 439
 - finance models 437
 - goal programming 454–464
 - preemptive 450–451
 - public sector models 60–62, 440–443
 - versus single-objective 64
 - weighted sums 451–453
- multistart search 894–895 (*see also* discrete improving search)
- mutations genetic algorithms 903
- mutually exclusive choices 664–665
- myopic rules (*see* greedy rules)
- NASA Capital Budgeting Application 662–663, 764–765
- NCB Circuit Board TSP Application 685–686, 888–889
- nearest child rule 773
- negative definite matrices 945
 - local optimality conditions 946–947
 - test for concave functions 952
- negative dicycles 488
 - Bellman-Ford algorithm 500–501
 - difficulty for shortest paths 488–489
 - Floyd-Warshall algorithm 507–509
 - improving cycle directions 585–586
- negative semidefinite matrices 945
 - local optimality conditions 946–947
 - test for concave functions 952
- neighborhood search (*see* improving search)
- neighborhoods 94–95, 887–892
 - discrete 887–892
- Nelder, R 974
- Nelder-Mead search 972–979
 - Algorithm 16G 973
 - direction 975–976
- Nelder-Mead shrinking 978–979
- Nelder-Mead step sizes 976–977
- network cycle cancelling search Algorithm 10B (*see* cycle cancelling search)
- network design problems 699–702
 - big- M constants 747–748
 - switching constraints 701
- network flow gain/loss flows 630–632
- network flows (*see also* each model class)
 - linear assignment 607–610
 - maximum flow 611–620
 - minimum cost flows 559–563
 - multicommodity 625–629
 - network design 699–702
 - shortest path problems 647
 - time-expanded models 565–568
 - transportation problems 502–506

- network rudimentary search Algorithm 10A
(*see* cycle direction search)
- network simplex search 591–601
Algorithm 10C 599
simplex cycle directions 597–598
- networks 479, 559 (*see also* digraphs)
- Newton step barrier search 412–419
Algorithm 7B 419
directions 413–414
multiplier management 417–419
step sizes 415–417
- Newton step 412–414, 960
- Newton's method 959–964
Algorithm 16E 962
starting close 964
versus gradient search 963
- Nifty Notes Machine Scheduling
Application 702–703
- NLP (*see* nonlinear programming)
- node-arc incidence matrices 568–570
linear dependence of cycles 591–592
spanning tree bases 594–595
total unimodularity 603–604
- nodes 479
- nonbasic variables 219, 836
superbasic 1041
- nondominated solutions (*see* efficient points)
- nonimproving moves 894–895
- nonlinear constraints 48
- nonlinear functions 48
- nonlinear objective functions 49, 64
- nonlinear programming 50 (*see also*
constrained nonlinear programming;
unconstrained nonlinear
programming)
multiobjective 440
- nonlinear regression (*see* curve fitting)
- nonnegativity constraints 25
- nonpositive variables 211
converting to nonnegative 211
- nonsingular matrices 223 (*see also* singular
matrices)
- nonsmooth functions 922
Nelder-Mead search 972–979
versus smooth 922–924
- nonsymmetric matrices 214
- norms of vectors 90
- NP** complexity class 864
- NP-Complete** complexity class 869
- NP-Hard** complexity class 869
- numerical search 14
dependence on starting point 15
- objective function coefficients
ranges 326–328
sensitivity analysis 299–300
sensitivity computer outputs 326–329
what if's 330–333
- objective function modifying relaxations 738
- objective functions 25–26
as interior move directions 386–387
contours 32–33
- objectives 4
- Ohio National Bank Application 162–163
- ONB Application 162
- one-dimensional search (*see* 1-dimensional
search)
- operations planning models 152–161
balance constraints 158–159
- operations research approach 3
dependence on time and resources 19
- Operations Research 1
- Opt**[≤] threshold version of problem Opt 863
- Opt**[≥] threshold version of problem Opt 863
- Opt**^{feas} feasibility version of problem Opt 863
- optima (*see* global optima)
- Optimal Ovens, Inc (OOI) Application 558
- optimal solutions 7
denoted with * 8
exact 15
from constraint relaxations 742
from Lagrangian stationary points 1014–1017
from relaxations modifying objective
functions 743
heuristic 16
unique versus alternative 34–35
values denoted with ν^* 8
- optimal values 35
- optimization models 4 (*see also* mathematical
programs)
- OR (*see* operations research)
- order quantity 5
- output variables 9
- Oxygen System Engineering Design
Application 992–994
- P** ≠ **NP** conjecture 869–871
- P** complexity class 864

- pairwise interchanges discrete improving search 890–891
- parameters of models 9
 - indexed symbolic 41–42
- parametric programming 338–344 (*see also* sensitivity analysis)
 - inadequacies of range outputs 338
 - multiple coefficient changes 340–344
 - one coefficient 338–339
- parent bounds 769–770
 - terminating by 769–770
- parent of a tree node 768
- Pareto optimal points (*see* efficient points)
- partial derivatives 110–111 (*see also* gradients)
- partial solutions 752–753
 - active 754
 - branched 754
 - completions 752–753
- partitioned LP standard form
 - complementary dual solution 355–356
 - primal basic solution 355
- partitioning constraints 667
- paths 480, 570
- paths shortest versus shortest subpaths 487–488
- paths versus chains 570–571
- penalty functions 1029
 - differentiability 1031–1032
 - exactness 1032–1033
- penalty methods 1028–1034
 - concluding optimality 1031
 - multiplier management 1033
 - sequential unconstrained penalty techniques 1033–1034
- penalty multipliers 1028
 - management 1033
- penalty objective function 1028
- permanently labeled node Dijkstra shortest paths 510–511
- PERT (*see* program evaluation and review technique)
- PERT Maximum Likelihood Application 920
- Pfizer Pharmaceuticals Lot Sizing Application 995
- Phase I 129 (*see also* two-phase method)
 - artificial model 129–130
 - artificial variables 129–130
 - in two-phase method 129–134
 - minimum cost flows 563–564
 - objective function 131
 - outcomes 132–133
 - starting solutions 129–130, 212–215
 - testing infeasibility 132–133
- Phase II 129 (*see also* two-phase method)
 - proceeding from Phase I 132
- Pi Hybrids Application 40
- piecewise-linear approximation of separable programs 1002–1003, 1065–1073
 - correctness 1065–1073
- pivots simplex 264
- plant location problems (*see* facility location problems)
- points (*see* solutions)
- polyhedral combinatorics (*see* cutting plane theory)
- polyhedral sets 126, 203–204
 - convexity of 126
 - extreme directions 204
 - extreme points 202
 - LP feasible sets as 126
- polynomial reduction 866–867
- polynomial-time algorithm complexity 861
- populations in genetic algorithms 902
- portfolio management 1003–1004
- positive definite matrices 945
 - local optimality conditions 946–947
 - test for convex functions 952
- positive dicycles 519
- positive semidefinite matrices 945
 - local optimality conditions 946–947
 - test for convex functions 952
- post-optimality analyses (*see* sensitivity analysis)
- posynomial functions 1008
- posynomial geometric programming 1009–1011, 1073–1082
 - degrees of difficulty 1079–1080
 - dual program 1077–1082
 - logarithmic change of variables 1075–1076
 - standard form 1073–1074
- precedence constraints 521
 - in job shop scheduling 712
- predecessor 520
- preemption in scheduling 703
- preemptive goal programming 459–461
- preemptive optimization efficient points 451
- preemptive optimization multiobjective optimization 450–451

- prescriptive models 12
 - versus descriptive 12–14
- President's Library Application 533–534
- pricing columns in delayed generation 811–821
- pricing columns in revised simplex 267–269
- primal and dual LP standard forms 352
 - partitioned 354–355
- primal complementary slackness 311–312, 349–351
- primal linear programs 304
- primal simplex search strategy for LP optimality 357–358
- primal-dual search
 - interior point Algorithm 7C 424
 - interior point complexity 430
 - interior point strategy 424
 - interior point 421–428
 - interior point Algorithm 7C 424
 - interior point management of complementary slackness 423
 - interior point move directions 422–423
 - interior point step size 423
 - interior point strategy (non simplex) 422
 - interior-point 421
 - interior-point directions 422
 - interior-point step size 423
 - linearly assignment Hungarian Algorithm 10D 612
 - simplex 365–371
 - simplex Algorithm 6B 369
 - simplex dual change directions 368
 - simplex dual step size 368
 - simplex restricted primal 366–367
 - simplex strategy for LP optimality 365–366
- principal minors 945
- principal submatrices 945
- principle of optimality 489–490
- probabilistic models (*see* stochastic models)
- probability density function 918
- problems vs. instances (*see* complexity theory)
- process times 703
- processor scheduling (*see* scheduling)
- program evaluation and review technique 920–921
- project activity 520
- project duration 520
- project management 520
- project networks 521–522
 - acyclic property 528
- projection on equality constraints
 - affine-scaled form 398–399
 - burden of interior point methods 390–394
 - least squares curve fitting 391–392
 - obtaining feasible directions 392–394
 - obtaining improving directions 394–395
 - projection matrices 392
- projective transformation method 430
- pseudo-polynomial time algorithm
 - complexity 871–872
- PTime** complexity class 865
- Purdue Final Exam Scheduling Application 59
- pure integer programs 57 (*see also* discrete optimization)
- QA application (*see* Quick Aid Application)
- QAP (*see* quadratic assignment problems)
- QP (*see* quadratic programming)
- quadratic assignment problems 677–680
 - flow-distance 678
 - formulation of TSPs 694–693
 - tractability 685
- quadratic fit search 932–935
 - Algorithm 16C 934
- quadratic form 945
- Quadratic Portfolio Management Application 1003–1004
- quadratic programming 1051–1061
 - active set methods 1055–1061
 - direct solution of KKT conditions 1054
 - equality constrained 1053–1055
- quadratic Taylor approximation (*see* second-order Taylor approximation)
- quasi-Newton condition 965
 - satisfied by BFGS 971
- quasi-Newton methods 964–972
 - versus gradient search 965
 - versus Newton's method 965
- Quick Aid Application 179
- random keys 904–905
- random variables 16
- rank one matrices 967
- ratio constraints 150–151
- realizations of a random variables 17
- reciprocals of convex/concave functions 952
- rectilinear distance 176
 - linear modeling 176–177
 - versus Euclidean 176

- recursions (*see* functional equations)
- reduced costs 231
 - cycle direction computation 576
 - revised simplex computation 267–269
- reduced gradient search 1038–1051
 - Algorithm 17C 1048
 - basic, nonbasic, superbasic 1041–1042
 - major and minor iterations 1049–1050
 - move directions 1044–1045
 - nondegeneracy assumption 1042–1043
- reduced gradients 1043–1044
- reduces to polynomial reductions from
 - (P) to (Q) 866
 - denoted $(P) \propto (Q)$ 866
- reductions among problems (*see* complexity theory)
- reflecting step in Nelder-Mead 977
- regression (*see* curve fitting)
- relaxations bounds 739–742
- relaxations choosing big- M s 747–751
- relaxations constraint 735–736
- relaxations LP 737–738
- relaxations modifying objective 738
- relaxations optimal solutions 742–743
- relaxations proving infeasibility 738–739
- relaxations rounded solutions 744–747
- relaxations strengthening 747–751
 - with valid inequalities 777–778
- relaxing constraints (*see* sensitivity analysis)
- release times 703
- reoptimization (*see* parametric programming)
- reorder point 5
 - residual digraphs 583–584
 - feasible directions 584–585
 - improving directions 585–586
- residual errors 918
- resources interpretation of constraints 288–290
- reverse arc 571
- revised simplex search 260–272 (*see also* simplex search)
 - Algorithm 5C 278
 - computing reduced costs 267–269
 - pricing vector 267
 - updating basis inverse 264–266
 - use of basis inverse 261–266
- rewards of Markov Decision Processes 542
- RHS (*see* right-hand sides)
- right-hand sides 47
 - ranges 322–324
 - sensitivity analysis 294–296, 323–324
- Risk and Detection of Breast Cancer
 - Application 543–545
- River Power Application 751–752
- root node 754
- rounded solutions
 - for incumbent solutions 765–768
 - from relaxations 744–747
- routing problems
 - multiple routes 693–695
 - traveling salesman 679–693
- saddle points 943
 - versus local optima 943–947
- safety stock 5
- scalars distinguished by subscripts 89
- scalars single real numbers 89
- scalars variables denoted in italics 89
- scaling (*see* affine scaling)
- scenarios in stochastic programming 179–180
- schedule conflict 704–705
- schedule slack 527–528
- scheduling (*see also* early start schedules; late start schedules)
 - conflict constraints 704–705
 - disjunctive variables 704–705
 - equivalences among objectives 710
 - handling of due dates 706
 - job shop 710–713
 - objective functions 706–708
 - single processor 702–710
 - time decision variables 703
- second derivatives 937–939
- second partial derivatives (*see* Hessian matrices)
- second-order necessary optimality conditions 946
- second-order sufficient optimality conditions 946–947
- second-order Taylor approximation 939–941
- sensitivity analysis 11 (*see also* parametric programming)
 - adding or dropping constraints 297–298, 322–323
 - adding or dropping variables 303, 333–334
 - constraint coefficients rates of change 298–299
 - LHS changes 296

- sensitivity analysis (*continued*)
 - objective function coefficients 299–300
 - objective function coefficients rates of change 301–303
 - qualitative for LPs 293–303
 - quantitative for LPs 304–310
 - relaxing versus tightening constraints 293
 - RHS changes 294–296, 322–323
 - only for single parameter changes 320–335
- separable functions 1001
- separable programming 1002–1003, 1065–1073
 - convex 1071–1072
 - piecewise-linear approximation 1002–1003, 1065–1073
 - standard form 1065–1066
- sequential decision making 528–529
- sequential quadratic programming (SQP) 1061–1065
 - Algorithm 17E 1063
 - strategy 1062
 - subproblem 1063
- sequential unconstrained barrier technique 1037–1038
 - Algorithm 14B 1038
- sequential unconstrained penalty technique 1033–1034
 - Algorithm 17A 1034
- Service Desk Design Application 1029–1030
- set covering constraints 667
- set covering problems 667–672
- set packing constraints 667
- set packing problems 668
- set partitioning constraints 667
- set partitioning problems 668, 672–675
 - with column generation 674–675
- setup costs (*see* fixed charges)
- Shanno, D 966
- shift scheduling models
 - air crews 673–674
 - covering constraints 164–165
 - LPs 162–166
- shortest path problems Acyclic
 - Algorithm 9D 518
 - computational order 518
- shortest path one to all nonnegative problems
 - Dijkstra Algorithm 9C 509
- shortest path problems 481
 - acyclic digraphs 515–519
 - all to all 493–494
 - Bellman-Ford Algorithm 9A 496–498
 - Bellman-Ford Algorithm 9A computational order 499
 - classification 481, 485
 - Dijkstra Algorithm 9C computational order 515
 - Floyd-Warshall Algorithm 9B 501–509
 - Floyd-Warshall Algorithm 9B computational order 502
 - functional equations 489–490
 - network flow formulation 647
 - one to all nonnegative costs 509–515
 - one to all 489–493
 - one to one 485
 - solving by LP 494
 - tractability 494
- shrinking in Nelder-Mead 978–979
- signomial geometric programming 1082
- simple lower bound 272
- simple upper bound 272
- simplex algorithm (*see* simplex search)
- simplex cycle directions 596–598
- simplex dictionaries 239–241
 - relationship to improving search 243
- simplex directions 228–230
 - feasibility 229–230
 - minimum cost flows 597–598
 - test for improving 231–232
- simplex search (*see also* dual simplex search; lower- and upper-bounded simplex search; network simplex search; primal-dual simplex search; revised simplex search)
 - computational order 429
 - degeneracy 253
 - dual Algorithm 6A 363
 - primal basis update 234–235
 - primal directions 229–230
 - primal finite convergence 257–258
 - primal global optimality 236–237
 - primal lower- and upper-bounded form Algorithm 5D 278
 - primal revised form Algorithm 5C 271
 - primal Rudimentary Algorithm 5A 235
 - (primal) starting basic solution 227
 - primal step sizes 232
 - primal two-phase Algorithm 5B 247
 - primal two-phase approach 245–250
 - primal with dictionaries 242–243

- primal with tableaux 242–243
 - primal-dual Algorithm 6B 369
- simplex tableaux 241
- simulated annealing 898–902
 - Algorithm 15D 898
 - temperature 899
- simulation models 12, 17–18
- simultaneous linear equations 223–224
- single commodity flows 629
 - versus multicommodity 626–627
- single complement discrete improving search 892
- single machine scheduling (*see* single processor scheduling)
- single objective optimization 59–60
 - versus multiobjective 64
- single processor scheduling 704–712 (*see also* scheduling)
- singular matrices 223
 - relation to linear dependence 223
 - testing by determinants 223
- sink nodes 560–561
- slack variables 209
 - converting main inequalities to nonnegativities 209–211
- Small Lagrangian Numerical Application 828–829
- smooth functions 109, 722
 - versus nonsmooth 922–924
- soft constraints 455
 - goal program modeling 455
- solutions 88
 - as vectors 88–91
 - encoding in genetic algorithms 904–906
- solver software 67
- solvers vs. modeling languages 66
- source nodes
 - in network flows 560–561
 - in shortest paths 481
- Spanning Trees Complexity Application 856–857
- spanning trees 594
 - max/min weight (*see* maximum/minimum spanning trees)
 - network flow bases 594–595
- staff planning models (*see* shift scheduling models)
- stages of a dynamic program 533
- standard forms
 - affine-scaled LPs 399–401
 - convex programs 998
 - differentiable NLP 1019
 - linear assignment 618
 - linear program interior points 389
 - linear programs 209–215, 352
 - linearly constrained NLP 1038–1039
 - lower- upper-bounded LPs 272
 - mathematical programs 26, 46–47
 - network flow problems 568–570
 - partitioned linear programs 260–262, 354–355
 - posynomial geometric programming 1073–1074
 - primal and dual LPs 352
 - quadratic programs 1051
 - separable programs 1065–1066
 - transportation problems 605
- star notation (*) 5
- start node project scheduling 521
- start times
 - early 523
- starting feasible solutions 129
 - basic 227
 - big- M methods 135–138
 - minimum cost flow 563–565
 - two-phase method 129–134, 212–214
- states of a dynamic programs 529
- states of Markov Decision Processes 542
- static models 166
- stationary points 941
 - necessary optimality conditions 941
 - of Lagrangians 1013–1014
- steepest ascent directions 114–115
- steepest descent directions 114–115
- Steiner tree problems 856
- step sizes
 - active set QP 1056–1057
 - in improving search 104–105
 - in simplex 232
 - minimum ratio rule 232
 - Nelder-Mead 976–977
 - Newton step barrier 415–417
 - reduced gradient methods 1046–1047
 - with cycle directions 577–578
 - with lower and upper bounds 276–277
 - zero due to degeneracy 255–257

- stochastic models 16–19
 - Markov Decision Processes 541–545
 - stochastic programming 179–184
 - versus deterministic 18–19
- stochastic programming 179
 - extensive form 184
 - extensive vs. large-scale forms 184
 - scenarios 179–180
 - two-stage 179
 - versus deterministic 181
 - with recourse 179
- stochastic simulation 17–18
- stopping early in branch and bound 770–772
 - error bounds 770–772
- strong duality 310, 351
- subgradient search (*see* Lagrangian duals)
- subgradient search Lagrangian dual 833–836
- subpaths 487–488
- subscripts 40 (*see also* indexing)
 - for components of vectors 89
- subtour elimination constraints 688
- subtours 688
- SUMT (*see* sequential unconstrained min/maximization technique)
- superbasic variables 1041
- Superfi Speaker Matching Application 683–684
- superscripts for distinguishing vectors 89
- supply nodes (*see* sources)
- surplus variables (*see* slack variables)
- Swedish Steel Application 147–148, 656
- switching constraints 658–660
 - big- M constants 749–751
- symmetric matrices 214
 - quasi-Newton requirement 965
- symmetric quadratic program form 1051
- symmetric traveling salesman problems (*see* traveling salesman problems)
- system boundary 9
- tabu list 894
- tabu search 893–887
 - Algorithm 15C 895
- tardiness 706
- target levels (*see* goal levels)
- Taylor series approximations 939–941
- temperature simulated annealing 899
- temporarily labeled node Dijkstra shortest paths 510
- termination 754
 - by bound 759
 - by infeasible 759
 - by parent bound 769–770
 - by solving 759
- Texaco Gasoline Blending Application 989–990
- Texas Transfer Application 481–482
- three-point pattern (*see* 3-point pattern)
- tight constraints (*see* active constraints)
- tightening constraints (*see* sensitivity analysis)
- time horizons 170
- time-expanded networks 565–568
- time-phased models
 - balance constraints 168–169
 - LPs 166–171
 - time horizons 170–171
- Tinyco Cash Flow Application 630
- Tmark Facilities Location Application 695–696
- Top Brass Trophy Application 201–202
- total enumeration 731–734
 - exponential growth 733–733
- total supply equals total demand 562–563
- total unimodularity of network flow
 - matrices 603–604
 - implications for solution integrality 603–604
- tours in traveling salesman problems 685
- tractability 11
 - assignment and matching 685
 - constructive search 884–885
 - continuous versus discrete variables 53–54, 144
 - convex programs 1000–1001
 - engineering design NLPs 992
 - gain/loss flows 632
 - improving search 98, 109–118
 - linear versus nonlinear functions 52
 - linearly constrained NLPs 989
 - longest path problems 519–520
 - LP versus network flows 570
 - LPs 203
 - multicommodity flows 629
 - network flows versus shortest path 647
 - posynomial geometric programs 1010–1011
 - quadratic programs 1006
 - separable programs 1002–1003
 - shortest path problems 494

- single versus multiobjective 64, 397
- smooth versus nonsmooth functions 922–924
- total enumeration 733–734
- unconstrained versus constrained NLPs 915
- versus validity 11
- transitions of Markov Decision Processes 542
- transportation problems 604–607
 - standard form 605
- transshipment nodes 560–561
- traveling salesman problems 679–693
 - formulating asymmetric 690–692
 - formulating symmetric 687–690
 - heuristic methods 873–874, 888–893
 - quadratic assignment formulation 692–693
 - subtour elimination constraints 688
 - subtours 688
 - symmetric versus asymmetric 687
- trees 594
- TSP (*see* traveling salesman problems)
- Tubular Products Operations Planning
 - Application 152
- Tucker, A. W. 351
- Turing, A. 859
- twice differentiable functions 938
- twice-around heuristic application for TSPs 873–874
- Two Crude Petroleum Application 24
- Two Ring Circus Application 482
- two-phase method 129–134 (*see also* Phase I; Phase II)
 - Algorithm 3B 129
 - simplex Algorithm 5B 247
- U.S. Forest Service Application 144
- unbounded models 37–39
 - detection with improving search 108–109
 - detection with simplex 252
 - primal to dual relationships 347–349
- unconstrained global maxima (*see* unconstrained global optima)
- unconstrained global minima (*see* unconstrained global optima)
- unconstrained global optima 122
 - convex/concave objectives 950–951
- unconstrained local maxima (*see* unconstrained local optima)
- unconstrained local minima (*see* unconstrained local optima)
- unconstrained local optima 122
 - first-order necessary conditions 942
 - second-order necessary conditions 946
 - second-order sufficient conditions 946–947
- unconstrained nonlinear programming 913
 - 1-dimensional search 924–935
 - BFGS search 966–972
 - curve fitting 916–918
 - global optimality conditions 950–951
 - golden section search 925–929
 - gradient search 955–959
 - local optimality conditions 941–947
 - maximum likelihood estimation 919–921
 - Nelder-Mead search 972–979
 - Newton’s method 959–964
 - quadratic fit search 932–935
 - quasi-Newton methods 964–972
 - single-variable 915
- unconstrained optima (*see* unconstrained global optima)
- Undecidable** complexity class 865
- undirected graph 482–483 (*see also* graph)
 - directing 483–484
- unimodal objective functions single variable 924–925
- unique optimal solutions 34–35
- unrestricted variables 212
 - converting to nonnegative 212
- URS (*see* unrestricted variables)
- USPS Single Variable Application 914
- valid equalities partial convex hull
 - dimension 796
- valid inequalities 777
 - face-inducing 794
 - facet-inducing 794
 - families of 782–788
 - Gomory mixed integer 785–787
 - Gomory pure integer 782–785
 - in branch and cut 777–782
 - minimal cover 787–788
- validity of models 11
 - simulation models 12
 - versus tractability 11
- variable costs 658
 - versus fixed 658

- variable elimination methods (*see* reduced gradient methods)
- variables (*see* decision variables)
- variable-type constraints 24–25
- variance of return 1004
- vectors 89–91
 - addition and subtraction 90
 - components of denoted with subscripts 89
 - denoted boldface lower case notation 89
 - dimension 89
 - distinguished by superscripts 89
 - dot product 90
 - geometric interpretation 89
 - norm or length 90
 - representing one-dimensional array of scalars 89
 - scalar multiple 90
- verbal models 4
- vertices (*see* nodes)
- Virginia Prestress Location Application 175
- VP Application(*see* Virginia Prestress) 175

- Wagner, H. 529
- Wagner-Whitin lot sizing Application 529–532
 - dynamic programming solution 529–532
- warehouse location problems (*see* facility location problems)
- Warshall, S. 501
- Wastewater Network Design Application 700
- We Build Construction Application 521
- weak duality 345
- weighted sums 90
 - linear functions 125
 - of convex/concave functions 952
 - of deficiencies 458–459
 - of objectives 451–453
 - with dot product notation 90–91
- what ifs (*see* sensitivity analysis)
- Whitin, T. 529
- Wilderness Energy (WE) Application 633
- WLP (*see* warehouse location problems)
- Wolfe, P. 836
- work in process 707
- worst-case heuristic algorithms 872–874

- zigzagging in gradient search 96

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank

APPLICATIONS

1.1	Mortimer Middleman	2	10.8	Wilderness Energy (WE)	633
2.1	Two Crude Petroleum	24	11.1	Indy Car Knapsack	661
2.2	Pi Hybrids	40	11.2	NASA Capital Budgeting	662
2.3	E-Mart	51	11.3	EMS Location Planning	666
2.4	Bethlehem Ingot Mold	53	11.4	AA Crew Scheduling	673
2.5	Purdue Final Exam Scheduling	59	11.5	Mall Layout Quadratic Assignment	678
2.6	DuPage Land Use Planning	61	11.6	CDOT Generalized Assignment	681
3.1	DClub Location	91	11.7	Superfi Speaker Matching	683
4.1	Forest Service Allocation	144	11.8	NCB Circuit Board TSP	685
4.2	Swedish Steel	147	11.9	KI Truck Routing	693
4.3	Tubular Products Operations Planning	152	11.10	Tmark Facilities Location	695
4.4	Canadian Forest Products Limited (CFPL) Operations Planning	154	11.11	Wastewater Network Design	700
4.5	Ohio National Bank (ONB) Shift Scheduling	162	11.12	Nifty Notes Single-Machine Scheduling	702
4.6	Institutional Food Services (IFS) Cash Flow	167	11.13	Custom Metalworking Job Shop	711
4.7	Highway Patrol	172	12.1	Bison Boosters	734
4.8	Virginia Prestress (VP) Location	175	12.2	River Power	751
4.9	Quick Aid (QA)	179	13.1	IMRT Planning for Radiation Therapy Optimization	812
5.1	Top Brass Trophy	201	13.2	Lagrangian Relaxation of CDOT Generalized Assignment	823
5.2	Clever Clyde	244	13.3	Small Lagrangian Numerical Example	828
7.1	Frannie's Firewood	386	13.4	Global Backpack (GB) Numerical Example	837
8.1	Bank Three Investment	437	13.5	Heart Guardian Facilities Location	843
8.2	Dynamometer Ring Design	439	14.1	Spanning Tree Examples of the Complexity Challenge	856
8.3	Hazardous Waste Disposal	440	14.2	Set Partitioning and Decision Problems	863
9.1	Littleville	477	14.3	Nondeterministic Solvability of ILP Threshold	864
9.2	Texas Transfer	481	14.4	Reduction of Set Partition to Steiner Tree	867
9.3	Two Ring Circus	482	14.5	Twice-Around Heuristic for TSP with Triangle Inequality	873
9.4	We Build Construction	521	16.1	USPS Single Variable	914
9.5	Wagner-Whitin Lot Sizing	529	16.2	Custom Computer Curve Fitting	916
9.6	President's Library	533	16.3	PERT Maximum Likelihood	920
9.7	Electoral Vote Knapsack Application	538	17.1	Beer Belge Location Allocation	987
9.8	Risk and Detection of Breast Cancer	543	17.2	Texaco Gasoline Blending	989
10.1	Optimal Ovens (OOI)	558	17.3	Oxygen System Engineering Design	992
10.2	Agrico Chemical Time-Expanded Network Flow	566	17.4	Pfizer Optimal Lot Sizing	995
10.3	Marine Mobilization Transportation Problem	606	17.5	Quadratic Portfolio Management	1003
10.4	CAM Assignment	609	17.6	Cofferdam Design	1006
10.5	Building Evacuation Maximum Flow	619	17.7	Service Desk Design	1029
10.6	Bay Ferry Multicommodity Flow	626	17.8	Filter Tuning	1039
10.7	Tinyco Cash Flow with Gains and Losses	630	17.9	Modified Version of Pfizer Application 17.4	1061