Quantum plug n' play: modular computation in the quantum regime

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

CrossMark

OPEN ACCESS

RECEIVED
12 August 2017

REVISED
20 October 2017

ACCEPTED FOR PUBLICATION
10 November 2017

PUBLISHED
4 January 2018

PAPER

Quantum plug n' play: modular computation in the quantum regime

Jayne Thompson[1], **Kavan Modi**[2], **Vlatko Vedral**[1,3] **and Mile Gu**[1,4,5,6]

[1] Centre for Quantum Technologies, National University of Singapore, Singapore
[2] School of Physics and Astronomy, Monash University, Australia
[3] Department of Physics, University of Oxford, United Kingdom
[4] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[5] Complexity Institute, Nanyang Technological University, Singapore 639673, Singapore
[6] Author to whom any correspondence should be addressed.

E-mail: cqttjed@nus.edu.sg and gumile@ntu.edu.sg

Keywords: quantum information, quantum protocol, quantum computing

Abstract

Classical computation is modular. It exploits plug n' play architectures which allow us to use pre-fabricated circuits without knowing their construction. This bestows advantages such as allowing parts of the computational process to be outsourced, and permitting individual circuit components to be exchanged and upgraded. Here, we introduce a formal framework to describe modularity in the quantum regime. We demonstrate a 'no-go' theorem, stipulating that it is not always possible to make use of quantum circuits without knowing their construction. This has significant consequences for quantum algorithms, forcing the circuit implementation of certain quantum algorithms to be rebuilt almost entirely from scratch after incremental changes in the problem—such as changing the number being factored in Shor's algorithm. We develop a workaround capable of restoring modularity, and apply it to design a modular version of Shor's algorithm that exhibits increased versatility and reduced complexity. In doing so we pave the way to a realistic framework whereby 'quantum chips' and remote servers can be invoked (or assembled) to implement various parts of a more complex quantum computation.

Modern computing relies crucially on the philosophy that we need not redesign the wheel. Every practical programming language supplies a library of built-in functions. In executing complex computations, we call these functions at will, taking for granted that we need not understand their exact logical or physical implementation. A modern sorting algorithm, for example, generally calls on a comparison function $u(x, y)$ that outputs 0 or 1 depending on whether x should be listed before y. By supplying different comparison functions, we may easily adapt such a program to sort a sequence of data by any number of different parameters. The sorting algorithm itself need not change; just the choice of which comparison function to use. Thus, modular architecture bestows immense flexibility [1–3].

As computations become more complex, modular design holds significant practical value. The evaluation of a complex function u may require immense computational resources and only be implementable by specialized facilities, meanwhile the server may not want to disclose their method for evaluating u. A modular design allows a client to construct devices that invoke these complex functions as subroutines through a standardized interface without any knowledge of their remote implementation. This lack of knowledge has many advantages; should a server find a better way to implement u, either through a new algorithm or new physics, he need not inform his clients. A client may casually select amongst the services of many competing servers, knowing that her device need not be altered to fit the particular implementation of u. Meanwhile, the server ensures that the clients gain no more information about his computational method—outside what is available from its input–output relations.

These advantages of modular design would particularly benefit practical quantum technologies. The immense challenges of quantum processing imply that the near future will likely see various institutions each synthesizing particular quantum processes. This motivates distributive computational schemes; such as a

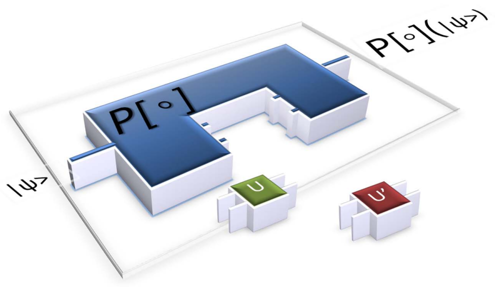

Figure 1. Modularity can be interpreted in a black-box framework: suppose that Alice can call upon a black-box that implements a unitary U, and wants to use this box to implement some U-dependent quantum process $\mathcal{P}[U]$. Modularity allows Alice to construct some U-independent 'plug n' play device' $\mathcal{P}[\circ]$, where '$\circ$' is being used as a dummy variable which ranges over the set of possible input unitaries $\{U\}$. That is colloquially '$\circ$' represents the slot where the box implementing the unitary can be plugged in. Connecting the input (output) interfaces of a black-box which implements U, to an appropriate port in $\mathcal{P}[\circ]$, allows Alice to immediately realize $\mathcal{P}[U]$. Similarly if Alice is also supplied with a black-box that implements U', then she can realize $\mathcal{P}[U']$ simply by switching the boxes.

repository of external servers, each specialized in executing specific quantum algorithms on supplied input. To what extent can a client, Alice, construct a universal device that invokes such a server —much as one invokes built-in functions on Mathematica—such that her device automatically performs a computation $\mathcal{P}[U]$, whenever the server's algorithm transforms an input $|\phi\rangle$ into $U|\phi\rangle$?

This question is particularly important for a prominent class of quantum algorithms that encompasses quantum factoring, solution of linear equations, and the DQC1 (the power of one bit of quantum information) algorithm for evaluating the normalized trace of unitary matrices [4–11]. These algorithms share the common property that their classical input x is encoded within a suitable unitary operator U_x. Quantum speed-up explicitly exploits the property that U_x can scale exponentially, and yet still be represented by a polynomial sized quantum circuit. The algorithm then operates by realizing some U_x-dependent complex quantum process $\mathcal{P}[U_x]$, whose output statistics compute some desired function $f(x)$.

Modularity is naturally desirable as each input x requires a different operator U_x. A naive synthesis of $\mathcal{P}[U_x]$ would involve creating a different quantum circuit for each possible input which is far from ideal. Could Alice adopt a more modular approach? One may envision a series of different black-boxes, each promising to output $U|\phi\rangle$ when given input $|\phi\rangle$. Can Alice then construct some fixed 'plug n' play device', such that by 'plugging in' a black-box that implements a specific U_x, her device computes $\mathcal{P}[U_x]$ (see figure 1)? If possible, such a device is clearly advantageous. Different laboratories could engineer implementations of different U_x that exploit the advantages of specific physical realizations; which can then be interchanged freely by Alice to compute $f(x)$ for different values of x.

This article introduces the framework of modular design in the quantum mechanical setting. We formulate the 'modularity constraint', as a general principle that specifies the exact conditions where modularity is unattainable. We show how this inflects surprising inflexibility on many celebrated quantum algorithms. We will see, for example, that the quantum circuit for standard factoring protocols would need to be tailored specifically for each number we wish to factor [12, 13]. This forces us to build a new circuit implementation almost from scratch every time we change numbers.

Simultaneously, we describe how the modularity constraint can guide us in developing new algorithms that do admit modular designs—indicating whether a given quantum algorithm can be made modular without sacrificing its functionality, and if not, what functionality needs to be sacrificed. We apply this methodology to design two new quantum algorithms—modular DQC1 and modular factoring. The former can evaluate the normalized modulus of the trace of a completely unknown physical process, and the latter can perform the full functionality of factoring with a polynomial reduction in the number of elementary gates over Shor's algorithm. Both algorithms exhibit the full advantages of modularity—greatly reducing the extent to which their circuits need to be tailored to specific inputs.

Finally, we discuss how the modularity constraint changes when all black-boxes are promised to only transform input information subject to certain restrictions (e.g. using a particular physical architecture). We illustrate how this additional knowledge can significantly change what can be made modular. We connect our results to existing studies on controlling an unknown unitary on a quantum degree of freedom [14–18]. We illustrate that the diversity of conclusions about whether this is possible naturally emerges from different implicit assumptions about how the unknown unitary is physically realized.

1. Framework

Modularity can be formalized in the server client framework. Consider Alice, a client, who wishes to invoke the services of Bob, a server, to perform part of a computation. To do this, Alice and Bob must first agree on a *public interface*, a mutually agreed contract between Alice and Bob that defines how quantum information is communicated between them. The interface specifies

1. The physical system S_{in}, the exact Hilbert space $\mathcal{H}_{\text{in}}$ and computational basis $\mathcal{B}_{\text{in}}$ in which Alice will deliver input quantum information, ρ_{in}, for processing to the server.
2. The physical system S_{out} the exact Hilbert space $\mathcal{H}_{\text{out}}$ and computational basis $\mathcal{B}_{\text{out}}$ in which the server will return processed quantum information, ρ_{out} to Alice.

This contract ensures that Alice can invoke Bob to perform part of a computation with no ambiguity. Note that the interface does not restrict Alice from sending in a bipartition of a larger entangled system, nor preclude S_{in} or S_{out} from having degrees of freedom beyond $\mathcal{H}_{\text{in}}$ and $\mathcal{H}_{\text{out}}$. It merely establishes an agreed basis for the exchange of quantum information.

Meanwhile, Bob's computation can be specified by a *quantum algorithm*—an explicit sequence of elementary operations that describes how Bob transforms a given S_{in} encoding ρ_{in} to some S_{out} encoding ρ_{out}. At the fundamental level, this is expressed as an ordered list of elementary physical operations specific to the physical set-up Bob makes use of (e.g. switching on a certain magnetic field for a set duration, inclusion of a beamsplitter). In many cases though, this is often abstracted—analogous to programming languages for classical computing. A common method involves the quantum circuit representation, which lists a sequence of one and two qubit gates $U_1, U_2, \ldots, U_K$ on a relevant Hilbert space, with the understanding that each U_k in this list can be translated into some subsequence of elementary physical operations (for $1 \leqslant k \leqslant K$). If application of an algorithm A on S_{in} that encodes ρ_{in} always produces a S_{out} that encodes $U\rho_{\text{in}}U^{\dagger}$, we say that A is an algorithmic realization of U.

This framework allows for a formal definition of modularity. We pose the question: can Alice construct a U-independent device that takes advantage of the public interface, such that it implements $\mathcal{P}[U]$ whenever Bob's algorithm A is an algorithmic realization of U? If this is possible, we say that $\mathcal{P}[U]$ can be made modular with respect to U. Note that in this definition, Bob makes no promises to Alice outside what is specified by the publicly agreed upon interface. This ensures that Alice has a device that satisfies the following properties:

- **Independence of realization:** If Alice finds a new server, Charlie, that computes U with a more efficient algorithm A', Alice's device can make use of Charlie's service without any modification—provided Charlie adheres to the same public interface.
- **Independence of function:** If Alice wishes to implement $\mathcal{P}[U']$, she does not need to modify her device. Alice needs only to convince Bob (or some other server) to build an algorithm that computes U' in place of U.

These advantages mirror that of modular architectures in classical computing. Bob is free to use any algorithm A, and Alice's device will continue to function. He is free to manipulate any quantum information in S_{in} stored outside $\mathcal{H}_{\text{in}}$ as he wishes, and use whatever physical process he likes to generate ρ_{out}. None of these changes should affect Alice, she can remain blissfully unaware of these details and be confident that her device implements $\mathcal{P}[U]$ for whatever U that Bob computes.

One can think of A as being encased inside a black-box, such that only its inputs and outputs are accessible. Alice wishes to make use of this box in her laboratory to synthesize $\mathcal{P}[U]$, without any knowledge of its internal configuration (see figure 1). Note that we do not assume $\mathcal{P}[U]$ is necessarily a unitary process—its input and output states can differ in both entropy and dimension.

Examples. These features are particularly relevant for quantum protocols which operate by encoding an input x within a x-dependent unitary U_x, with the end goal of retrieving information about some function $f(x)$. The DQC1 algorithm is a simple example (see figure 2(a)) [4]. In DQC1, the goal is to estimate the normalized trace of a unitary matrix M of dimension $d = 2^n$, i.e., estimate $f(M) = \text{tr}[M]/d$. This is done by encoding M within a unitary operator U_M, whose matrix representation is precisely M. The protocol then operates by realizing the quantum operation $\mathcal{P}_{\text{DQC1}}[U_M]$ which—when acting on an appropriate quantum state—results in a single qubit that can be used to estimate $f(M)$. If $\mathcal{P}_{\text{DQC1}}[U]$ can be made modular with respect to U, then Alice would be able to build a universal device, that is able to evaluate $f(M)$ when given any stand-alone unit that is promised to realize U_M. Note that in DQC1, the encoding $U_M = M$ is trivial. Thus, the usual convention to use U and M interchangeably and DQC1 is often said to evaluate $f(U)$. We will adopt this convention.

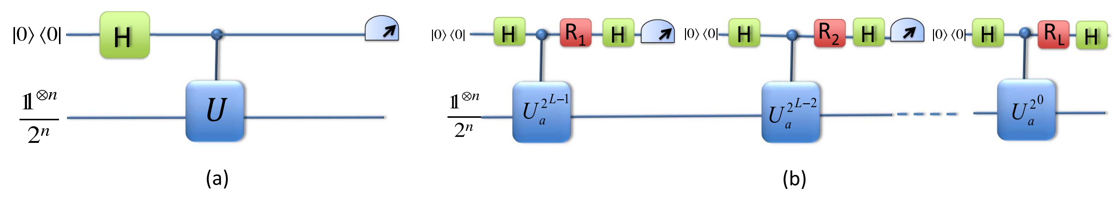

Figure 2. DQC1 (a) can be implemented by applying a unitary U to a register of completely mixed qubits, controlled on a single pure qubit in state $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, where a control unitary is defined as $C_U : |y\rangle|\phi\rangle \to |y\rangle U^y|\phi\rangle$ for $y = 0, 1$. The resulting process $\mathcal{P}_{\rm DQC1}[U]$ outputs a single qubit in state $\rho_c = \frac{1}{2d}\sum_{i,j=0,1}{\rm tr}[U^{i-j}]|i\rangle\langle j|$, whose X (Y) measurement statistics allows for the estimation of ${\rm Re}[{\rm tr}[U]]/d$ (and ${\rm Im}[{\rm tr}[U]]/d$ respectively), where we have denoted $U^\dagger$ as U^{-1}. The standard factoring circuit (b) encodes the number N inside an operation $U_a : |q\rangle \to |(q * a) \bmod N\rangle$ where all arithmetic is modulo N. We can find a factor of N by stringing together a series of controlled $U_a^{2^x}$ operations, followed by a quantum Fourier transform modulo 2^L (QFT). The latter is implemented by a series of Hadamard and R_j gates, where $R_j = |0\rangle\langle 0| + |1\rangle\langle 1| \exp(-2\pi i \sum_k M_{j-k}/2^k)$ such that $M_{j-k} = 1$ if $(j-k)^{\rm th}$ detector clicked and 0 if it did not (k indexes previous detectors). The resulting process $\mathcal{P}_F[U_a]$ outputs a binary decimal approximation of the fraction j/r, for some $j \in \{0, \ldots, r-1\}$ selected at random.

Similar benefits pertain to quantum factoring. To factor an input $N = pq$, one introduces a Hamiltonian H_a, with eigenspectrum $\{j/r : a^r \equiv 1 \bmod N, j = 0, \ldots, r-1\}$ for some suitable $a < N$ (see appendix B for details). The rest of the algorithm then involves determining r through quantum phase estimation. Specifically, this involves constructing a series of unitary matrices $U_a^{2^x} = \exp(2^x 2\pi i H_a)$, for $x = 0, 1, \ldots, L-1 \in O(\log_2 N)$ that effectively encode the desired N. The quantum algorithm then incorporates this information within a sophisticated quantum process $\mathcal{P}_{\rm F}[U_a^{2^x}]$. Application of this quantum process on an appropriate, fixed, initial quantum state allows for output statistics that can be used to recover the value of r (see figure 2(b)). Here, modularity implies that Alice can construct a universal factoring circuit that factors all numbers—provided circuits for applying $U_a^{2^x}$ are inserted into the right locations. This would significantly reduce how much a quantum circuit needs to be specifically tailored to factor different numbers, and—as we will see—has corollary benefits in reducing the circuit's gate complexity.

2. Constraints on modularity

We now turn to the question: when can $\mathcal{P}[U]$ be made modular with respect to U? In some cases, this question is fairly trivial. Consider the process $\mathcal{P}_V[U] : |\phi\rangle \to VU|\phi\rangle$ for any fixed unitary V. This clearly can be made modular with respect to U. Alice needs only encode her input in $S_{\rm in}$ as stipulated by the public interface, and then apply V to the state returned. Regardless of what algorithm the server employs, or unitary U it computes, Alice recovers $VU|\phi\rangle$. For more complex quantum processes $P[U]$, the answer is less obvious. Here, we introduce *the modularity constraint*, a condition that any $\mathcal{P}[U]$ must satisfy in order to be modular with respect to U.

The constraint invokes the notion of *black-box properties* [19]. Let $\mathcal{A}$ be the set of all possible algorithms the server can use (for now, this set will contain all possible algorithms). A property on $\mathcal{A}$ is a function $p : \mathcal{A} \to \{0, 1\}$ that maps each element of $\mathcal{A}$ to a binary number. Black-box properties are properties that depend only on the input-output relations of A: if any two algorithms, A and A', have statistically indistinguishable output on all coinciding input, then $p(A) = p(A')$. The name 'black-box' refers to the idea that if an algorithm is executed inside a black-box such that only the input and output is visible, an external observer can only determine its black-box properties.

For example, the truth value of the statement 'A outputs $|0\rangle$ on input $|1\rangle$' is a black-box property. Any two quantum algorithms where this property differs will have statistically distinguishable output upon appropriate input. Meanwhile, the truth value of 'A involves five two-qubit interactions' is a not a black-box property; two algorithms with differing numbers of two-qubit interactions can have statistically identical input–output relations.

Suppose now Alice outsources some algorithm A to Bob via the aforementioned public interface. How much information can Alice retrieve about which algorithm $A \in \mathcal{A}$ Bob decided to use? Given that Alice has access only to what inputs she gives Bob, and what outputs Bob returns to her, it seems unlikely that Alice can determine any information about this question beyond what is accessible through its black-box properties. This line of thought can be formalized (see appendix A for formal proof), giving constraints on when a computation $\mathcal{P}[U]$ can be made modular with respect to U:

Proposition 1 **(The Modularity Constraint).** *Let $\mathcal{A}$ be the set of all possible quantum algorithms. Suppose $\mathcal{P}[U]$ can be made modular with respect to U, then $\mathcal{P}[U]$ cannot be used to reveal information about which algorithm $A \in \mathcal{A}$ a server used to implement U beyond what is attainable from its black-box properties.*

Consequences. This constraint significantly limits what computations can be made modular. In DQC1, the it implies *any* quantum operation $\mathcal{P}[U]$ that accurately estimates $\mathrm{tr}[U]/d$ cannot be made modular with respect to U. This is because determination of $\mathrm{tr}[U]/d$ automatically reveals information to Alice about a server's choice of $A \in \mathcal{A}$ beyond what is possible through its black-box properties. Specifically consider two algorithms A_p, $p \in \{0, 1\}$ that each implement U, specified by gate sequence $U_1, U_2, \ldots, (-1)^p U_k$ on $\mathcal{H}_{\mathrm{in}}$, followed by a swap operation to transfer the resulting state into $\mathcal{H}_{\mathrm{out}}$. Note that all black-box properties of A_0 and A_1 must coincide, as all inputs to A_p will result in statistically indistinguishable output (as any two quantum states that differ by a global phase are operationally indistinguishable). Nevertheless the value of the complex number $\mathrm{tr}[U]/d$ differs by a factor of (-1) between the two cases $p = 0$ and $p = 1$. Hence the requirement that DQC1 directly estimates $\mathrm{tr}[U]/d$, implies that that the operational statistics of any such $\mathcal{P}[U]$ could be used to discriminate $p = 0$ from $p = 1$.

The modularity constraint also applies to more elementary circuit components. This allows us to determine which part(s) of the algorithm precludes its modular implementation. In the standard DQC1 circuit, the only U-dependent component is a unitary quantum process $\mathcal{P}_c[U]$, which involves the application of

$$C_U = \mathbb{1}_d \oplus U = \begin{pmatrix} \mathbb{1}_d & 0 \\ 0 & U \end{pmatrix}. \tag{1}$$

Physically, this represents applying U to a d-dimensional subspace of an extended Hilbert space of dimension $2d$. If this extension is achieved through an ancillary qubit, this corresponds to controlling U on a quantum mechanical ancilla. This process cannot be made modular with respect to U, as $\mathcal{P}_c[U]$ can also distinguish whether a server used A_0 or A_1. This retrieves a result that has received significant recent attention—it is impossible to implement the control of a completely unknown unitary [15]. We will touch on this topic again in section 4.

The latter observation precludes any quantum algorithm that makes use of $\mathcal{P}_c[U]$ from being modular with respect to U, a list that would include phase estimation, quantum factoring, and the solution of linear equations amongst many others. This presents a rather negative outlook for achieving non-trivial modular architectures in the quantum regime. We will, however, see in the next section that we can sometimes work around the modularity constraint. Certain computational tasks that currently make use of $\mathcal{P}_c[U]$ can be redesigned to use more modular components.

3. Building modular quantum algorithms

Suppose now Alice wishes to evaluate some $f(x)$ by executing some x-dependent quantum operation $\mathcal{P}[U_x]$ and yet all current constructions are non-modular, we observe two possibilities:

(i) If $f(x)$ can be used to violate the modularity constraint (it can be used to distinguish two different algorithms, A, A' with identical black-box properties), then any $\mathcal{P}[U_x]$ that can compute $f(x)$ strictly cannot be modular.

(ii) If $f(x)$ does not directly violate the constraint, then it implies that current methods for evaluating $f(x)$ have inadvertently invoked some non-modular $\mathcal{P}[U_x]$.

In case (i), we need to isolate information in $f(x)$ that distinguishes A and A'. By discarding this information, we can define some approximation of $f(x)$, say $g(x)$ that does admit a modular implementation. In case (ii), lack of modularity is a feature of existing constructions—rather than a consequence of a fundamental no-go theorem. Thus, we may be able to find alternative quantum operations that allow evaluation of $f(x)$, and remain modular.

The task of DQC1—to evaluate $f(U) = \mathrm{tr}[U]/d$—is an example of case (i), while we will see that quantum factoring falls under case (ii). We now illustrate these in detail, and in doing so, demonstrate two new algorithms, modular DQC1 and modular quantum factoring.

Modular DQC1. How close can we get to computing $f(U) = \mathrm{tr}[U]/d$ in a way that is modular with respect to U? Consider computing not $f(U)$, but rather its modulus $g(U) = |f(U)|$. The resulting function $g(U)$ no longer distinguishes any A and A' with identical black-box properties. Furthermore, one can show that this compromise is in some sense optimal—knowing any more information about $f(U)$ would necessarily violate modularity (see appendix A).

Indeed there exists a quantum operation $\mathcal{P}[U]$ that computes $g(U)$ in a modular fashion (see figure 3). This involves using a single pure control qubit and *two* completely mixed registers of n qubits. Two controlled-SWAP operations interspersed by the application of U on one register—followed by a measurement of the control qubit—completes the algorithm. Note that the implementation of U can be freely outsourced to a third party via

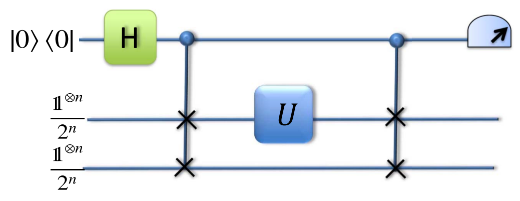
$|0\rangle\langle 0|$
H
$\frac{\mathbb{1}^{\otimes n}}{2^n}$
$\frac{\mathbb{1}^{\otimes n}}{2^n}$
U

Figure 3. The modular DQC1 protocol is based on coherently swapping *two* registers of completely mixed qubits, controlled on a single pure control qubit in state $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. The protocol uses two such control swap operations, interspersed by application of U on a single reservoir. The output is a single (control) qubit whose X-measurement statistics allow estimation of $|\mathrm{tr}[U]|/d$.

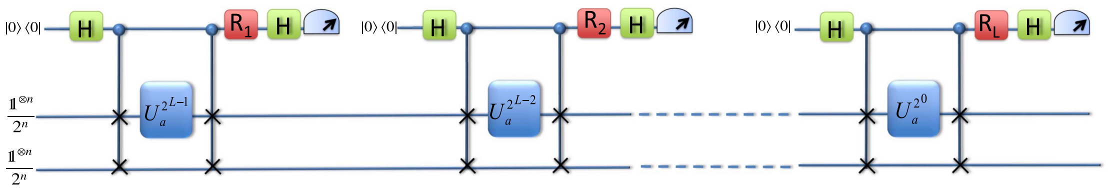

Figure 4. It is possible to factor using a polynomial series of application of the modular DQC1 protocol. Acting on an input constituted by pure control qubits in state $|0\rangle$ and two reservoirs of completely mixed qubits, this basic building block functions identically to a controlled- $U_a^{2^x} \otimes U_a^{2^x \dagger}$. When coupled to a QFT modulo 2^L this algorithm outputs a binary decimal approximation to the difference $(j-k)/r$, for some $j, k \in \{0, \ldots, r-1\}$, between two eigenvalues of H_a.

a public interface (see figure 3), and the resulting device then allows for the estimation of $|\mathrm{tr}[U]|$ without knowing which U the third party applied.

While evaluation of $|\mathrm{tr}[U]|$ is a more limited computation than the evaluation of $\mathrm{tr}[U]$, modular DQC1 can have significant benefits. The algorithm can treat U as an arbitrary input—in the true spirit of a plug n' play interface. One can use this circuit as a 'probe', where exposed quantum wires are connected to the (input) output of the (second) first swap gate. These wires can then be directly connected to an arbitrary unknown unitary processes, allowing for the measurement of the absolute value of its trace.

Modular quantum factoring. Unlike DQC1, the desired output information in quantum factoring does not conflict with the modularity constraint. Recall that in quantum factoring, the number $N = pq$ that we wish to factor is encoded within a unitary U_a, whose Hamiltonian generator H_a has eigenspectrum $\{j/r : a^r \equiv 1 \bmod N, j = 0, \ldots, r-1\}$. Success in determining r to sufficient accuracy allows us to compute a factor of N. Standard methods of quantum factoring involve using quantum phase estimation to determine the eigenvalues of U, i.e., each $\lambda_j = j/r$. This information, however, violates the modularity constraint. The algorithmic constructions of U_a and $e^{2\pi i/r}(U_a)$ share identical black-box properties, and nevertheless differing eigenvalues. This precludes these standard approaches from being modular—implying the quantum circuit must be adapted for factoring each specific N.

This conflict is avoidable. Collecting information about each individual λ_j is unnecessary. Information about r can be entirely retrieved by evaluating the gap $\lambda_j - \lambda_{j-1} = 1/r$ between consecutive energy eigenvalues of U_a's Hamiltonian generator. Unlike the eigenvalues themselves, this gap is completely fixed by the black-box properties of an algorithm and hence its evaluation cannot violate the modularity constraint. This suggests that factoring can be made modular without any compromise to efficiency or output information.

In appendix C, we demonstrate an explicit construction. The core idea involves the use of a polynomial sequence of modular DQC1 circuits—connected in a serial configuration (see figure 4). The resulting algorithm recovers the differences between eigenvalues; in general these are also of the form k/r for $k = 0, \ldots, r-1$. For the purposes of factoring, these differences contain the same amount of useful information as the spectrum itself. In the appendix, we prove our construction factors successfully in approximately

$$O\left(\frac{pq}{(p-1)(q-1)}\log\log r\right) \tag{2}$$

runs. This is comparable to Shor's algorithm which typically succeeds in $O(\log\log r)$ runs.

This results in a significantly more modular implementation of Shor's protocol. Such a construction has dual advantage. Recall that each U_a must be tailored specifically to the number N we wish to factor. A modular architecture thus allows each individual laboratory to optimize the synthesis of each $U_a^{2^x}$, using whichever physical realization they fancy (ion traps, photonic qubits, etc.) and a third party can 'mix n' match' these realizations to factor the number they wish.

Furthermore, the modular design has an immediate side benefit in reducing the complexity of quantum factoring. In conventional quantum factoring, each unitary $U_a^{2^x}$ is supplied via its algorithmic construction—an explicit sequence $U_1, \ldots, U_K$ of elementary one and two qubit gates. The factoring circuit then synthesizes $C_{U_a^{2^x}}$ by adding a control to each individual gate in the sequence, i.e., implementation of $C_{U_1}, \ldots, C_{U_K}$. This would require $O(n^3)$ controls per operator [20]. In contrast, in this enhanced factoring algorithm, the number of controls can be significantly reduced—we need only $O(n)$ controlled-SWAP gates, all of which can be reused regardless of which number we factor.

4. Modularity with partial knowledge

Our work has focused on the most general form of modularity—where the server has complete freedom in how and what U is implemented, and Alice's device must synthesize $\mathcal{P}[U]$ regardless. What about more restricted situations, where Alice limits herself to dealing with specific types of servers, and thus has extra knowledge about what algorithms her device is required to work with? This is formalized by an additional agreement in the public interface, where the server promises any algorithm A he applies will be restricted to some strict subset, $\bar{\mathcal{A}} \subset \mathcal{A}$ of all possible algorithm. Armed with this promise, many additional $\mathcal{P}[U]$ may become modular. The intuition is that while a given $\mathcal{P}[U]$ may distinguish two algorithms A_0 and A_1 in $\mathcal{A}$ that share identical black-box properties, one or both of these algorithms may not lie in $\bar{\mathcal{A}}$.

Before making a general statement, we illustrate this situation by example. Consider a public interface where Alice communicates with complying servers via the (non-degenerate) energy levels $\{|k\rangle\}_k$ of an atom $S = S_{\text{in}} = S_{\text{out}}$. They agree to use, $\mathcal{B} = \{|0\rangle, \ldots, |d-1\rangle\}$, the first d energy levels, to encode relevant quantum information. Let $\mathcal{H}$ be the Hilbert space spanned by this basis. When Alice supplies an atom S that encodes $|\phi\rangle$ in $\mathcal{H}$, Bob promises to manipulate the atom using some algorithm $A \in \bar{\mathcal{A}}$ such that the resulting state encodes $U|\phi\rangle$ for some U. Alice now wants to harness this interface to build a universal device that estimates the normalize trace, $f(U) = \text{tr}[U]/d$. In the sections above, we established this violates the modularity constraint. No quantum operation $\mathcal{P}[U]$ that accurately estimates $f(U)$, can be made modular with respect to U.

This changes with appropriate extra knowledge. First define $\mathcal{B}' = \{|d\rangle, |d+1\rangle, \ldots, |2d-1\rangle\}$ as the next d energy levels of S. Now suppose that the server makes one additional promise: All algorithms A it applies, will lie in $\bar{\mathcal{A}}$, the set of algorithms that leaves invariant (i) any quantum information in $\mathcal{B}'$ (ii) all quantum coherence between $\mathcal{B}$ and $\mathcal{B}'$. i.e., if Alice were to supply Bob a state $|\phi'\rangle$ with support on $\mathcal{B} \cup \mathcal{B}'$, Bob promises to return the state

$$(U \oplus \mathbb{1}_d)|\phi'\rangle = \begin{pmatrix} U & 0 \\ 0 & \mathbb{1}_d \end{pmatrix}|\phi'\rangle, \tag{3}$$

where $\mathbb{1}_d$ denotes an identity operator. The output of the above equation now exhibits different statistics for each possible choice of U—including those that differ by a global phase (i.e., U and $e^{i\phi}U$). Thus any information $f(U)$ reveals about U is no longer a black-box property, and therefore estimation of $\text{tr}[U]/d$ no longer violates the modularity constraint. Indeed, we can relabel the $\mathcal{B} \cup \mathcal{B}'$ basis elements as $|b\rangle_{r1}|k\rangle_{r2}$ for $b \in \{0, 1\}$ and $k \in \{0, \ldots, d-1\}$. Thus Bob's promise can be reinterpreted as a contract to implement U on target register 'r2' controlled on the virtual qubit 'r1'. On reception of S_{out}, Alice can estimate $\text{tr}[U]/d$ by measuring 'r1' in the Pauli X (and Y) bases on repeated runs.

In contrast, if the server's choice of algorithms is unrestricted, the above argument breaks down. Consider for example, two algorithms A_p, $p = \{0, 1\}$, each consisting of first a measurement projecting the state into $\mathcal{B}$, followed by application of a gate sequence $V_0, V_1, \ldots (-1)^p V_k$ acting only on the basis $\mathcal{B}$ (i.e. each $V_j = U_j \oplus \mathbb{1}_d$). Clearly A_0 and A_1 have identical black-box properties. Yet, if Alice could accurately estimate estimate $(-1)^p\text{tr}[U]/d$—a quantity that depends on p—she would be able to discern between A_0 and A_1, and thereby violating the modularity constraint. Therefore while computing $f(U)$ violates the modularity constraint in the general scenario—it can still field a modular implementation provided Alice has additional information regarding Bob's algorithmic construction. We can capture the above observations formally by adapting the modularity constraint for general $\bar{\mathcal{A}}$.

Proposition 2 (modularity with partial knowledge). *Let $\bar{\mathcal{A}}$ be a subset of all possible algorithms. We say that $\mathcal{P}[U]$ can be made modular with respect to U and $\bar{\mathcal{A}}$ only if Alice can build a U-independent device that implements $\mathcal{P}[U]$ whenever the server implements U using only algorithms in $\bar{\mathcal{A}}$. Suppose $\mathcal{P}[U]$ can be made modular with respect to U and $\bar{\mathcal{A}}$, then $\mathcal{P}[U]$ cannot be used to reveal information about which algorithm $A \in \bar{\mathcal{A}}$ a server used to implement U beyond what is attainable from its black-box properties.*

The proof is identical to that of the standard modularity constraint, substituting $\overline{\mathcal{A}}$ for $\mathcal{A}$. In the example above, both A_0 and A_1 have identical black-box properties, yet any $\mathcal{P}[U]$ that computes $f(U)$ can differentiate A_0 from A_1. Thus such a $\mathcal{P}[U]$ would violate the general modularity constraint. However, both A_0 and A_1 lie outside $\overline{\mathcal{A}}$, and thus $\mathcal{P}[U]$ need not violate the modularity constraint with respect to $\overline{\mathcal{A}}$. In particular, equation (3) implies that any two algorithms $\overline{A}_0$ and $\overline{A}_1$ that lie within $\overline{\mathcal{A}}$ with the same black-box properties must synthesize the exact same U with the exact same global phase on $\mathcal{B}$. Therefore when Alice has preknowledge that all algorithms implemented by the server lie inside $\overline{\mathcal{A}}$, the modularity constraint no longer prohibits computation of DQC1.

Indeed, this problem has been explored in significant detail for the special case where $\mathcal{P}[U] = \mathcal{P}_c[U]$. This aligns with the longstanding problem of whether it is possible to control an unknown U on a quantum mechanical degree of freedom. The motivation was the seemingly paradoxical observation that proposals to add controls to unknown unitaries have been experimentally demonstrated [16]—despite formal proofs showing its mathematical impossibility [15]. A resolution of the puzzle was given by Friis *et al* [14], noting that all existing methods assumed knowledge of specific physical architectures to realize U. Here, we see how these discussions fit within the general context of modularity. $\mathcal{P}_c[U]$ indeed cannot be made modular with respect to U, in agreement with no-go results on controlling unknown unitaries. However, one can side-skirt this problem with preknowledge, as demonstrated in [14, 16, 17].

5. Discussion

Our research can be summarized in two parts. In the first, we formalized modularity in the context of quantum computation within the server client framework—where modularity ensures a client can construct a U-independent device that implements a quantum operation $\mathcal{P}[U]$ by blindly invoking a server to implement U. We proposed necessary constraints on such modular architectures, and explored their impact on existing quantum algorithms—encompassing quantum factoring and DQC1—that operate by probing the properties of some input dependent unitary operator U. The resulting 'modularity constraint' indicates that the aforementioned algorithms are generally non-modular, making it impossible to prefabricate devices that work for each possible input—forcing their circuits to be tailored for each specific input.

The second part explored ways to circumvent this constraint. We ascertain what sacrifices in functionality, if any, are required to refine existing algorithms to restore modular implementation. This resulted in two new algorithms: (i) a universal device that evaluates the normalized modulus of the trace of any exponentially large unitary U, even when this unitary is completely unknown and supplied within a black-box. (ii) a modular factoring algorithm that can factor numbers while recycling a far larger portion of its circuit architecture, allowing for a significant reduction in the number of control gates required for implementation.

One avenue of future research is to investigate applications of modular architectures outside pure computation. A natural scenario, for example, originates in the fields of metrology and sensing. Here universal devices that probe the properties of different physical processes, without making assumptions about what unitary operations they implement, can streamline and simplify both the state preparation and read-out stages of the metrology protocol [21]. Meanwhile, a second natural direction is to better understand the mathematical properties of such architectures, and thus better identify what plug n' play devices are achievable in quantum mechanics. One way to proceed is to consider whether the class of $\mathcal{P}[\circ]$ that field a modular implementation exactly coincides with quantum supermaps—a mathematical formalism that defines transformations between quantum maps [22].

On a more fundamental level, combinatorial evolution postulates that advanced technology evolves by combining technologies that already exist, and in turn becomes new building blocks for potential future technology [23]. This modular approach has demonstrated success in the context of designing classical circuits [24]. Similarly, more complex quantum technologies will likely evolve from taking existing quantum technologies as their basic building blocks.

Acknowledgments

The authors would like to thank A Brodutch, O Dalhston, K Kim and C Weedbrook for discussions. We are grateful for funding from the John Templeton Foundation Grant 53914 *'Occam's Quantum Mechanical Razor: Can Quantum theory admit the Simplest Understanding of Reality?'*; the Oxford Martin School; the Ministry of Education in Singapore, the National Research Foundation of Singapore and in particular NRF Award No. NRF–NRFF2016–02 and NRF-CRP14-2014-02, the Foundational Questions Institute (FQXi).

Appendix A. Constraining modularity

Proof of modularity constraint: Here, we provide a formal proof of the modularity constraint. First, let $\mathcal{A}$ be the set of all possible algorithms available to the server. Each property p then divides $\mathcal{A}$ into two equivalence classes. Thus given a particular A in $\mathcal{A}$, knowledge of $p(A)$ progressively better isolates which algorithm was used. We can then define the equivalence relation $\sim$ on $\mathcal{A}$ such that two algorithms $A,\ A' \in \mathcal{A}$ satisfy $A \sim A'$ if and only if all of their black-box properties coincide.

Consider the following statements.

(i) $\mathcal{P}[U]$ **can be made modular with respect to** U: Alice can construct a U-independent device that implements the quantum operation $\mathcal{P}[U]$, where all U dependence comes from its call to a public interface: The device can send Bob a quantum system S_{in} that encodes a quantum state ρ_{in} in a pre-agreed Hilbert space $\mathcal{H}_{\mathrm{in}}$. If Bob returns a quantum system S_{out} that encodes $U\rho_{\mathrm{in}}U^{\dagger}$, then Alice's device automatically performs the quantum operation $\mathcal{P}[U]$.

(ii) $\mathcal{P}[U]$ **can reveal information regarding which** A **Bob used beyond its black-box properties**: We can find some A_q, $q = \{0,\ 1\}$ such that $A_0 \sim A_1$, and nevertheless the operational behavior of $\mathcal{P}[U]$ depends on whether Bob used A_0 or A_1.

The modularity constraint states that (i) and (ii) cannot both be true. We will assume the truth of both and derive a contradiction. Let A_q, $q = \{0,\ 1\}$ be two algorithms that satisfy (ii). Suppose now that Alice constructs the U-independent device outlined in (i). Let $\mathcal{P}_q$ be the resulting quantum operations Alice's device performs when Bob applies A_q. Now (ii), implies the output statistics of $\mathcal{P}_0$ and $\mathcal{P}_1$ must differ, and thus depend on the value of q.

As the only q dependence lies in Bob's choice of A_q, this implies that there exist some ρ_{in} that Alice's device can send to Bob, such that the output state ρ_{out} is dependent of q. Let this state be denoted $\rho_{\mathrm{out}}^{(q)}$ depending on Bob's choice of q. Since $\mathcal{P}_q$ depends on q, this implies that Alice's device provides a method to discern between $\rho_{\mathrm{out}}^{(0)}$ and $\rho_{\mathrm{out}}^{(1)}$. Thus $\rho_{\mathrm{out}}^{(0)}$ and $\rho_{\mathrm{out}}^{(1)}$ are statistically distinguishable despite having coinciding input ρ_{in}. This implies that A_0 and A_1 do not have coinciding input-output statistics. In particular, we can define a property p that evaluates the truth value of the statement 'does A output $\rho_{\mathrm{out}}^{(0)}$ on input ρ_{in}?'. As p is clearly a black-box property, $A_0 \nsim A_1$ and we arrive at a contradiction.

Discarding minimal information to restore modularity. Here we elaborate further on the modular DQC1 algorithm, and show that it is the optimal modular approximation to the DQC1 algorithm. Specifically, suppose $f = f(U)$ is some function of U that we wish to evaluate. If $f(U)$ violates the modularity constraint, it implies that f can be used to distinguish two algorithms A and A' that lie in the same equivalence class (as defined above), i.e., Alice's universal device that computes $f(U)$ whenever the server it invokes applies U will give differing values, depending on whether the server implements A or A'.

We can rephrase this in terms of unitaries. Let A be named an algorithmic realization of U if it implements that unitary operator U. That is, if the server receives a state ρ encoded within basis $\mathcal{B}_{\mathrm{in}}$ in S_{in}, then application of A on S_{in} will result in a system S_{out} that encodes $U\rho U^{\dagger}$ in basis $\mathcal{B}_{\mathrm{out}}$. First note that if A and $\bar{A}$ both realize the same U, then $A \sim \bar{A}$. Then we can define an equivalence relation on the set of all unitaries, such that $U \sim U'$ if all algorithmic realizations of U and U' lie in the same equivalence class. Clearly, there necessarily exists two algorithms $A \sim A'$ for which $f(U)$ takes on different values iff $f(U) \neq f(U')$ for some $U \sim U'$.

Given f such that $f(U) \neq f(U')$ for some $U \sim U'$, consider some approximation $g(U)$ such that (i) $g(U) = g(U')$ whenever U and U' lie in the same equivalence class and (ii) there exists a U in each equivalence class such that $f(U) = g(U)$. Then by from above $g(U)$ does not reveal any more information about which $A \in \mathcal{A}$ the server used beyond its black-box properties. Thus the modularity constraint does not prohibit the existence of some $\mathcal{P}[U]$ that can accurately estimate $g(U)$, and simultaneously be modular with respect to U. Furthermore, it is an *optimal* candidate—in the sense that if $g(U)$ contained any more information about $f(U)$, then it would violate (i) and hence no longer admit modular realization. One can check that the modular DQC1 satisfies these properties, by setting $f(U) = \mathrm{tr}[U]/d$ and $g(U) = |f(U)|$.

Appendix B. Intuition behind modular factoring

To efficiently factor, it is sufficient to have an efficient algorithm that solves the order finding problem [12, 13, 25]: Given an input $a \in \mathbb{N}, 1 < a < N$, output the first value of r such that $a^r \equiv 1 \bmod N$. If a is chosen at random, the value of r will, with good probability, reveal the factors of N.

Quantum factoring algorithms function by determining the eigenvalues of a modular exponentiation operator

$$U_a|x\rangle = |(x * a) \bmod N\rangle, \tag{4}$$

which encode the value of r. Note that $(U_a)^r$ is the N-dimensional identity matrix. This last constraint forces the eigenvalues of this operator to be the rth roots of unity; these are complex numbers ω^{-j} which carry information about r through $\omega = \exp 2\pi i/r$ and $j = 1, \ldots, r - 1$. If we can measure the phase of an eigenvalue for which j/r is an irreducible fraction then we can find r.

Due to the closure of the rth roots of unitary under multiplication the eigenvalues of $U_a \otimes U_a^\dagger$ are also rth roots of unity. Hence it is functionally equivalent to find the phase associated with an eigenvalue of $V_a = U_a \otimes U_a^\dagger$. Thus, noting that the modular DQC1 protocol with input U_a is equivalent to DQC1 with input V_a, we may replace each control U_a with its more modular variant with negligible loss in efficiency.

Appendix C. Proof of correctness

Firstly we characterize the modular exponentiation operator defined in equation (4). Every eigenvector of this $N \otimes N$ unitary operator can be expressed in terms of some natural number $g_d < N$, as:

$$|\psi_{j_d}\rangle = \frac{1}{\sqrt{r_d}}(\omega_d{}^{-j_d(1)}|g_d * a\rangle + \ldots + \omega_d{}^{-j_d(r_d)}|g_d * a^{r_d}\rangle),$$

where r_d is an exponent satisfying $g_d^* a^{r_d} \equiv g_d \bmod N$ while the coefficients are defined through $\omega_d = \exp 2\pi i/r_d$ and $j_d \in \{0, \ldots, r_d - 1\}$. The associated eigenvalue is $\omega_d^{j_d}$. Note that the case $g_d = 1$ has r eigenvectors and associated eigenvalues of the form $\omega^j = \exp 2\pi ij/r$ for $j = 0, \ldots, r - 1$, while in general $r_d|r$ (i.e. r_d divides r) because $a^r \equiv 1 \bmod N$. Furthermore whenever $N = pq$ is coprime with g_d the relation $g_d(a^{r_d} - 1) \equiv 1 \bmod N$ implies $r_d = r$; these conditions are met by $(p - 1)(q - 1)$ natural numbers less than N. Implying that at most $p + q - 1$ possible values of g_d correspond to eigenrelations for U_a where the phase of ω_d has denominator $r_d \neq r$ [13].

With respect to the eigenbasis $|\psi_{j_d}\rangle$ we write the operator U_a as

$$U_a = \sum_d \sum_{[j_d=0,\ldots,r_d-1]} w_d^{j_d}|\psi_{j_d}\rangle\langle\psi_{j_d}|, \tag{5}$$

where the first sum, indexed by d, runs over the set $\{g_d\}$ and the nested sum runs over $j_d = 0, \ldots, r_d - 1$.

We now use this information to analyze the circuit in figure 4 in the main text. We simplify the calculation by using the binary decimal expansion

$$0.c_l c_{l+1} \ldots c_m = \frac{1}{2}c_l + \frac{1}{4}c_{l+1} + \ldots + \frac{1}{2^{m-l+1}}c_m. \tag{6}$$

In this convention a measurement of the control register at the end of the circuit yields a number

$$c = \sum_{i=0}^{L-1} 2^i c_i, \tag{7}$$

where the binary digit c_i is 1 if the ith detector clicked and 0 otherwise, while the binary decimal $c/2^L$ is the best estimate to the phase of some eigenvalue of $U_a \otimes U_a^\dagger$. To achieve sufficient accuracy we require $L = \log_2 t$ ancillary qubits where t is the power of 2 satisfying $N^2 \leqslant t \leqslant 2N^2$ [13].

The probability of obtaining a specific binary number c when measuring the circuit in figure 4 in the main text is:

$$P(c) = \frac{1}{N^2 t^2}\sum_{d,d'}\sum_{j_d,j_d'}|G|^2, \tag{8}$$

where

$$G = \sum_{b=0}^{t-1} \exp\left(2\pi bi\left(\frac{j_d}{r_d} - \frac{j_d'}{r_d'} - \frac{c}{t}\right)\right). \tag{9}$$

We deliberately chose the number of control qubits so that our measurement c/t can resolve $j_d/r_d - j_d'/r_d'$ to an accuracy sufficient for determining r: this implies their exists an eigenvalue for which our estimate has a bounded amount of error:

$$\left|\frac{j_d}{r_d} - \frac{j_d'}{r_d'} - \frac{c}{t}\right| \leqslant \frac{1}{2t}. \tag{10}$$

Under these conditions we inherit a lower bound on $|G|^2 \geqslant 4t^2/\pi^2$ [13], see also [12] for a more detailed argument.

If we are going to be successful in retrieving any information about r from c/t then (a) we need $j_d/r_d - j'_d/r'_d$ to have denominator r and (b) we need the numerator to be coprime with r.

Firstly there are $(p-1)(q-1)$ values of g which are coprime with N permitting at least $(p-1)(q-1)/r$, values of $r_d = r$ [13]. For each value of $r_d = r$ the number of eigenvalues corresponding to irreducible fractions j/r where $j \in \{0, \ldots, r-1\}$ is defined through Euler's totient function $\phi(r)$; which follows the relation $\phi(r)/r > \delta/\log\log r$ for a constant δ [12, 13, 26].

In the next section we demonstrate that for every j_d/r_d satisfying $r_d = r$ and $gcd(j_d, r) = 1$ there is a faction $j_d/r_d - j'_d/r'_d$ satisfying both (a) and (b); by symmetry this argument should apply equally to j'_d/r'_d. Hence the number of eigenvalues $j_d/r_d - j'_d/r'_d$ from which we can successfully determine r is:

$$\mathrm{Num}_c = N^2 - (N-\chi)^2 = \chi(2N-\chi), \tag{11}$$

where $\chi = \frac{\phi(r)(p-1)(q-1)}{r}$. And the probability our circuit succeeds (that is estimates a fraction with denominator r and numerator coprime with r) is

$$P'(c) = \mathrm{Num}_c * P(c) \geqslant \frac{4t^2}{N^2t^2\pi^2}\chi(2N-\chi). \tag{12}$$

For a direct comparison with Shor's result [12] we give the lower bound on the success probability:

$$\frac{4}{N\pi^2}(p-1)(q-1)\frac{\phi(r)}{r}. \tag{13}$$

This scales as the same order in N as standard factoring algorithms [12, 13]; in fact, asymptotically the probability of success using the modular DQC1 protocol goes like $P*(2-P)$ where P is the probability of success for Parker and Plenio's factoring routine [13]; so to first order in P (which tends to 0 as $N\to\infty$) we get a doubling in the success probability of the modular DQC1 protocol over that of Parker and Plenio, which recovers the cost of the extra register qubits used in our construction.

Appendix D. The number of fraction $j_d/r_d - j'_d/rd'$ which have denominator r and a coprime numerator

This section contains information required to derive equation (11).

Firstly fix the eigenvalue j'_d/r'_d and assume $r_d = r$ then

$$j_d/r_d - j'_d/r'_d = \frac{j - k'_d j'_d}{r}, \tag{14}$$

where we have let $r'_d = r/k'_d$ for some integer k'_d (which is always possible because r'_d divides r).

Now for a fixed value of j'_d/r'_d there are r possible numerators in equation (14) corresponding to the possible values of $j = 0, \ldots, r-1$. We want to establish a one to one correspondence between values of j which are coprime with r and values of the numerator of equation (14) which are coprime with r (for a fixed j'_d/r'_d).

Since we have fixed $k'_d j'_d$ we know $j - k'_d j'_d \equiv 0, \ldots, r-1 \bmod r$ (i.e., when $j = 0, \ldots, r-1$ so does $j - k'_d j'_d \bmod r$).

Additionally for any $\alpha, \beta \in \mathbb{Z}$ we have: $\alpha + \beta * r$ is coprime with r if and only if α is coprime with r (this follows very quickly from the contrapositive).

We show the relation in one direction $\alpha + \beta * r$ coprime with $r \longrightarrow \alpha$ coprime with r using the contrapositive. First assume α shares a common factor with r; that is let $\alpha = \lambda * \tau$ and $r = \lambda * \kappa$ (for integers κ, τ, λ). This implies $\alpha + \beta * r = \lambda(\tau + \beta * \kappa)$ and therefore $\alpha + \beta * r$ is not coprime with r.

It follows that for a fixed j'_d/r'_d; if the conditions: (a) fraction has denominator r and (b) numerator is coprime with r, are satisfied by j_d/r_d then there is a corresponding value of $j_d/r_d - j'_d/r'_d$ also satisfying (a) and (b). This argument is symmetric and can also be applied to j'_d/r'_d.

So the number of eigenvalues $j_d/r_d - j'_d/r'_d$ which cannot be used to determine r is the number of pairs $(j_d/r_d, j'_d/r'_d)$ for which is it impossible to determine r from either j_d/r_d, or j'_d/r'_d. Equation (11) is simply the total number of eigenvalues of $U_a \otimes U_a^\dagger$ minus the number that can not be used to determine r.

Appendix E. Equivalence of the modular DQC1 subroutine to a control unitary on a completely mixed register

We characterize the resulting action of the modular DQC1 protocol for the general case where the two reservoirs are initialized in arbitrary states ρ and σ. We compare this to the use of a control $U \otimes U^\dagger$ operation. We demonstrate equivalence when the case of the factoring protocol.

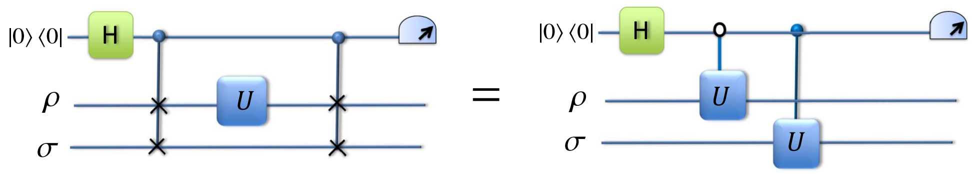

Figure 5. A modular DQC1 circuit (a) is formally equivalent to a pair of controlled unitary transformations: $|0\rangle\langle 0| \otimes U \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \mathbb{1} \otimes \mathbb{1}$ and $|0\rangle\langle 0| \otimes \mathbb{1} \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \mathbb{1} \otimes U$, respectively shown in (b). This equivalence holds for an arbitrary ρ, σ, and U. The measurement of the control qubit at the end yields the value of $\mathrm{tr}[U\rho] \times \mathrm{tr}[\sigma U^\dagger]$.

Consider two states $|\psi\rangle = \sum_{ijk} \alpha_{ijk}|i, j, k\rangle$ and $|\phi\rangle = \sum_{ijk} \beta_{ijk}|i, j, k\rangle$ which are each composed of three qubits. The tensor product of these states is

$$|\psi\rangle \otimes |\phi\rangle = \sum_{ijk}\sum_{lmn} \alpha_{ijk}\beta_{lmn}|il, jm, kn\rangle. \tag{15}$$

We define the operator S which swaps the mth qubit of ψ with mth qubit of ϕ:

$$S|\psi\rangle \otimes |\phi\rangle = S\sum_{ijk}\sum_{lmn} \alpha_{ijk}\beta_{lmn}|il, jm, kn\rangle \tag{16}$$

$$= \sum_{ijk}\sum_{lmn} \alpha_{ijk}\beta_{lmn}|li, mj, nk\rangle \tag{17}$$

$$= |\phi\rangle \otimes |\psi\rangle. \tag{18}$$

This furnishes a SWAP operator which interchanges two m qubit registers ρ and σ

$$\rho \otimes \sigma \to S\rho \otimes \sigma S = \sigma \otimes \rho. \tag{19}$$

When the registers are initialized as two arbitrary m qubit states, ρ and σ, due to the relation $U \otimes \mathbb{1}_m \rho \otimes \sigma S U^\dagger \otimes \mathbb{1}_m S = U \rho \otimes \sigma U^\dagger$, the state of the modular DQC1 circuit after the second SWAP in figure 5 is

$$\tau_{BB} = \frac{1}{2^{2m+1}}\begin{pmatrix} \rho \otimes \sigma & U\rho \otimes \sigma U^\dagger \\ \rho U^\dagger \otimes U\sigma & \rho \otimes U\sigma U^\dagger \end{pmatrix}. \tag{20}$$

When ρ and σ are eigenstates of U with eigenvalues $e^{i\lambda_\rho}$ and $e^{i\lambda_\sigma}$ respectively then the final state of the circuit is

$$\tau_{BB} = \frac{1}{2^{2m+1}}\begin{pmatrix} \rho \otimes \sigma & e^{i(\lambda_\rho - \lambda_\sigma)}\rho \otimes \sigma \\ e^{i(\lambda_\sigma - \lambda_\rho)}\rho \otimes \sigma & \rho \otimes \sigma \end{pmatrix}. \tag{21}$$

By comparison the state of a circuit implementing a controlled-$U \otimes U^\dagger$ on two registers initialized as ρ and σ is:

$$\tau_{U\otimes U^\dagger} = \frac{1}{2^{2m+1}}\begin{pmatrix} \rho \otimes \sigma & \rho U \otimes \sigma U^\dagger \\ U^\dagger\rho \otimes U\sigma & U^\dagger\rho U \otimes U\sigma U^\dagger \end{pmatrix}. \tag{22}$$

In general the final state of these circuits are the same when the registers are initialized as eigenstates of U:

$$\tau_{U\otimes U^\dagger} = \frac{1}{2^{2m+1}}\begin{pmatrix} \rho \otimes \sigma & e^{i(\lambda_\rho - \lambda_\sigma)}\rho \otimes \sigma \\ e^{i(\lambda_\sigma - \lambda_\rho)}\rho \otimes \sigma & \rho \otimes \sigma \end{pmatrix}. \tag{23}$$

Due to the linearity of quantum mechanics, the two circuits are equal for any input state that is an improper mixture of eigenstates of U, i.e. any density operator that is diagonal in the eigenbasis of U. This clearly includes complete mixed states, and all inputs during the operation of the modular factoring algorithm.

In the most general case, the modular DQC1 circuit as represented in equation (20) is formally equivalent to a pair of controlled unitary transformations as outlined in figure 5.

References

[1] Baldwin C Y and Clark K B 2000 *Design Rules: The Power of Modularity* vol 1 (Cambridge, MA: MIT Press)
[2] Knuth D E 1998 *The Art of Computer Programming: Sorting and Searching* vol 3 (Reading, MA: Addison-Wesley)
[3] Parnas D L 1972 *Commun. ACM* **15** 1053
[4] Knill E and Laflamme R 1998 *Phys. Rev. Lett.* **81** 5672
[5] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
[6] Ekert A K, Alves C M, Oi D K L, Horodecki M, Horodecki P and Kwek L C 2002 *Phys. Rev. Lett.* **88** 217901
[7] Datta A, Flammia S T and Caves C M 2005 *Phys. Rev.* A **72** 042316

[8] Poulin D, Laflamme R, Milburn G J and Paz J P 2003 *Phys. Rev.* A **68** 022302
[9] Emerson J, Lloyd S, Poulin D and Cory D 2004 *Phys. Rev.* A **69** 050305
[10] Temme K, Osborne T J, Vollbrecht K G, Poulin D and Verstraete F 2011 *Nature* **471** 87
[11] Poulin D, Blume-Kohout R, Laflamme R and Ollivier H 2004 *Phys. Rev. Lett.* **92** 177906
[12] Shor P W 1997 *SIAM J. Comput.* **26** 1484
[13] Parker S and Plenio M 2000 *Phys. Rev. Lett.* **85** 3049
[14] Friis N, Dunjko V, Dür W and Briegel H J 2014 *Phys. Rev.* A **89** 030303
[15] Araújo M, Feix A, Costa F and Brukner Č 2014 *New J. Phys.* **16** 093026
[16] Zhou X-Q, Ralph T C, Kalasuwan P, Zhang M, Peruzzo A, Lanyon B P and O'Brien J L 2011 *Nat. Commun.* **2** 413
[17] Friis N, Melnikov A A, Kirchmair G and Briegel H J 2015 *Sci. Rep.* **5** 18036
[18] Rambo T M, Altepeter J B, Kumar P and D'Ariano G M 2016 *Phys. Rev.* A **93** 052321
[19] Rice H G 1953 *Trans. Am. Math. Soc.* **74** 358
[20] Vedral V, Barenco A and Ekert A 1996 *Phys. Rev.* A **54** 147
[21] Friis N, Orsucci D, Skotiniotis M, Sekatski P, Dunjko V, Briegel H J and Dür W 2017 *New J. Phys.* **19** 063044
[22] Chiribella G, D'Ariano G M and Perinotti P 2008 *Europhys. Lett.* **83** 30004
[23] Arthur W B 2009 *The Nature of Technology: What It Is and How it Evolves* (New York: Simon and Schuster)
[24] Arthur W B and Polak W 2004 *Complexity* **11** 23
[25] Kitaev A Y 1995 arXiv:quant-ph/9511026
[26] Hardy G G H and Wright E M 1979 *An Introduction to the Theory of Numbers* (New York: Oxford University Press)