scientific reports

OPEN

Distribution of controlled unitary quantum gates towards factoring large numbers on today's small-register devices

Andrei Tănăsescu, David Constantinescu & Pantelimon George Popescu

Factoring a 2048-bit number using Shor's algorithm, when accounting for error correction, reportedly requires 400,000 qubits. However, it is well known that there is yet much time before we will have this many qubits in the same local system. This is why we propose a protocol for distributed quantum computation applicable to small register devices, specifically for the distribution of controlled unitary gates, the key element in the construction of every quantum computation algorithm. We leverage quantum sharing of partial results to obtain a parallel processing scheme, allowing for the first time the quantum distribution of very large gates with thousands of inputs using only small register devices with tens of qubits. In this way, we improve all previous controlled unitary gate distribution approaches, obtaining surprising results. The impact is quantified for recent milestone hardware realizations of quantum processors.

The security of today's critical communication protocols is generally based upon three pillars: public key encryption, digital signatures and key exchange, the implementation of which is most often based on the difficulty of number theoretic problems such as integer factorization and other hidden subgroup problems[1]. In fact, integer factorization is the computational problem behind today's most famous cryptosystem, the RSA (Rivest-Shamir-Adleman) cryptosystem, and thus a lot of work has went into developing increasingly sophisticated attacks, based on everything from approximation algorithms to quantum computing. Recently, a PQCrypto 2014 talk[2] estimated that by 2030 a billion-dollar quantum computer could break 2000-bit RSA in a few hours, a figure that was taken by NIST as *a serious long-term threat to the cryptosystems currently standardized by NIST*[1], a process which ultimately kicked off a competition to determine the best candidate to replace today's most popular cryptosystem whose third round finished in late 2021[3], with a replacement to be ready by 2024. The security of the communication protocols of tomorrow is thus heavily influenced by the advent of quantum computing.

Quantum computing is the branch of computational science that aims to harness quantum phenomena such as superposition and entanglement to perform computational feats, such as the famous polynomial-time Shor factoring algorithm[4]. A recent implementation analysis of Shor's algorithm[5] shows that factoring 2048-bit numbers requires at least 400,000 qubits working at least 1 trillion qubit-hours, when error correction is accounted for. While Shor's algorithm is trivially distributed using sector search, each "thread" still has to have 400,000 qubits, whereas the world's most powerful quantum processor is reportedly the 127-qubit IBM Eagle r1[6], itself a far cry from the 5-7 qubit systems freely available in the cloud.

Motivated by similar examples, distributed quantum computation has long studied peer-to-peer coupling in quantum-classical networks using teleportation[7]. One of the first steps in this direction was the distribution of controlled unitary gates[8], the quantum equivalents of the `if` programmatic statement. Using the observation that any function can be written as a sequence of `if` statements, this protocol was recently generalized to allow the distribution at no additional cost of quantum functions[9], including the Deutsch and Grover oracles, and even the fast modular exponentiation function which is the bottle-necking factor in Shor's algorithm. While this work shows that the system requirements for distributing a quantum function is the same as for a controlled unitary, the specific implementation of the previously mentioned protocols[8,9] have the disadvantage that they require a large number of ancillary qubits at the target site.

The distribution of a quantum function $\mathbf{U}_f \in \mathrm{End}\left(\mathbb{C}^{2^N} \otimes \mathbb{C}^{2^M}\right)$[9] (and particularly N-control unitaries[8]) starts by sharing each of the "control" qubits $C_1, \ldots, C_N$ with the computer containing the "target" qubits

Computer Science and Engineering Department, University Politehnica of Bucharest, Bucharest 060042, Romania. email: pgpopescu@yahoo.com

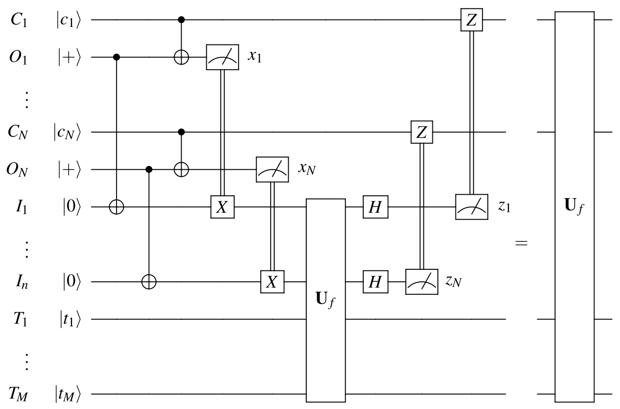

Figure 1. Multipartite distribution of N-qubit quantum functions[9].

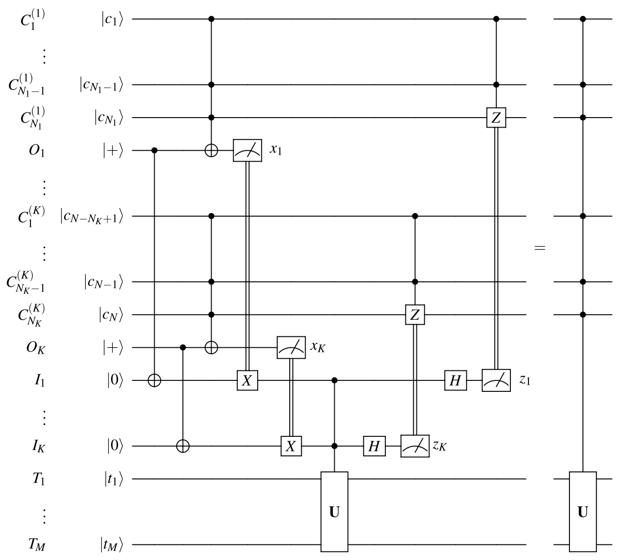
$C_1^{(1)}$ $|c_1\rangle$
$C_{N_1-1}^{(1)}$ $|c_{N_1-1}\rangle$
$C_{N_1}^{(1)}$ $|c_{N_1}\rangle$
O_1 $|+\rangle$
$C_1^{(K)}$ $|c_{N-N_K+1}\rangle$
$C_{N_K-1}^{(K)}$ $|c_{N-1}\rangle$
$C_{N_K}^{(K)}$ $|c_N\rangle$
O_K $|+\rangle$
I_1 $|0\rangle$
I_K $|0\rangle$
T_1 $|t_1\rangle$
T_M $|t_M\rangle$
x_1
x_K
z_1
z_K
X
H
Z
$\mathbf{U}$
=

Figure 2. Improved multipartite distribution of N-control unitaries[12].

$T_1, \ldots, T_M$, using N additional Bell pairs whose halves we denote by $O_1, \ldots, O_n$ (at the control site) and $I_1, \ldots, I_n$ (at the target site). The protocol then locally applies the quantum function $\mathbf{U}_f$, and finally decommissions the extra qubits, as shown in Fig. 1. This last step is required to avoid residual entanglement, and follows the transfer procedure discussed in[9,10]. Yet, the point of both protocols is not to circumvent register size limits, but rather to allow the processing of non-local data, and as such they both make use of a local copy of the gate. In brief, to distribute a N-control Toffoli gate in this way, we still need a $(N+1)$-qubit computer.

In the specific case of controlled unitaries that can be further optimized. A clever trick[11] is to use the relation between the 3-control Toffoli gate and the AND gate to share only partial results of the AND operations. In the bipartite setting, this trick allows expenditure of only 1 Bell pair rather than 2. This idea has been refined by[12] who eliminated the ancilla and also extended the approach to N-control unitary gates. In this setting, the control space is partitioned into K groups, $\mathcal{C} = \mathcal{C}_1 \otimes \cdots \otimes \mathcal{C}_K$, each with a possibly different number of qubits N_i, $\mathcal{C}_i \cong \mathbb{C}^{2^{N_i}}$, which we denote $C_1^{(i)}, \ldots, C_{N_i}^{(i)}$ for $1 \le i \le K$. Additionally, each control system shares a Bell pair with the target system. In the first step, the protocol applies a local Toffoli gate on each control system targeting the half of the Bell pair. These partial sums are then shared with the target system, where a K-control version of the unitary gate is performed. The last step of the protocol is to apply partially classically controlled $\mathbf{Z}$ gates on the control systems, as depicted in Fig. 2. Yet, the point of this protocol is to optimize for local agglomerations of qubits, so it is only a side effect that it helps our goal. In brief, to distribute a N-control Toffoli gate in this way using n-qubit control systems we have $K = \frac{N}{n-1}$ and so we still need still need a $(\frac{N}{n-1}+1)$-qubit computer. For example, taking N=400,000 and n=5, we still need a 100,000-qubit computer. Alternatively, requiring all quantum computers to be of the same size, i.e. setting $n = K + 1$, we find $n = \lceil 1 + \sqrt{N} \rceil$, i.e. when N=400,000 we need 633 computers each having 633 qubits, which is still far from current technology.

In summary, while the literature has optimized distribution of controlled unitaries to some extent, for fully nonlocal controlled unitaries its best solution still comes down to applying the full gate at the target site. Given the

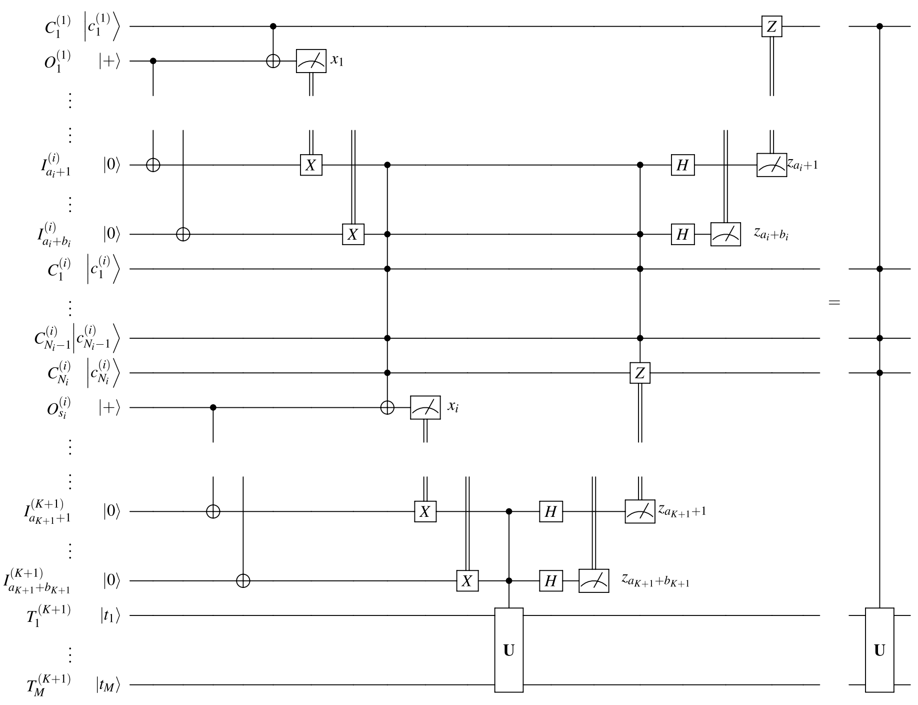
$C_1^{(1)}$ $\left|c_1^{(1)}\right\rangle$
$O_1^{(1)}$ $|+\rangle$
x_1
$I_{a_i+1}^{(i)}$ $|0\rangle$
$I_{a_i+b_i}^{(i)}$ $|0\rangle$
$C_1^{(i)}$ $\left|c_1^{(i)}\right\rangle$
$C_{N_i-1}^{(i)}$ $\left|c_{N_i-1}^{(i)}\right\rangle$
$C_{N_i}^{(i)}$ $\left|c_{N_i}^{(i)}\right\rangle$
$O_{s_i}^{(i)}$ $|+\rangle$
x_i
z_{a_i+1}
$z_{a_i+b_i}$
$I_{a_{K+1}+1}^{(K+1)}$ $|0\rangle$
$I_{a_{K+1}+b_{K+1}}^{(K+1)}$ $|0\rangle$
$T_1^{(K+1)}$ $|t_1\rangle$
$T_M^{(K+1)}$ $|t_M\rangle$
$z_{a_{K+1}+1}$
$z_{a_{K+1}+b_{K+1}}$
X
H
Z
U
=

Figure 3. Parallel cascade distribution of controlled unitaries.

lack of any specialized protocol that can work with small register devices, the only recourse is to decompose the K-control Toffoli gate into single- and two-qubit gates and then run it through a distributed quantum compiler, such as the one recently developed by IBM researchers[13] which optimizes the number of remote operations in the compiled circuit using integer programming. While serviceable, this black-box approach offers no particular insight as to how many small-register systems are required to efficiently run a workload requiring full connectivity. Thus, in this paper we set out to provide the first protocols for the distribution of N-control Toffoli gates using only small register devices.

Methods

Throughout this paper we consider a logical quantum system comprised of N control qubits, $C_1, \ldots, C_N$ and M target qubits, $T_1, \ldots, T_M$. The N control qubits are split across K local physical quantum systems, $S_1, \ldots, S_K$, while the M target qubits are consolidated in a single local physical quantum system, S_{K+1}. These $K+1$ local physical systems also include ancillary qubits used for entanglement distribution as well as computation. For convenience, we denote by N_i the number of controls at site S_i for $1 \leq i \leq K$ and relabel them $C_1^{(i)}, \ldots, C_{N_i}^{(i)}$. On top of these $K+1$ local physical systems we overlay a communication oriented tree, $G=(V,E)$, where $V=\{S_1, \ldots, S_{K+1}\}$, with the tree rooted at the target site S_{K+1}. Without loss of generality, we assume that the nodes are labeled in reverse breadth-first order. For simplicity, we reduce our construction to a single parameter, $1 \leq B \leq K$, corresponding to the branching factor, and all internal nodes will have exactly B children, except possibly the smallest indexed one, which may have fewer if the number of nodes is insufficient for them all to have B children. As an example, for a balanced binary tree with $K=5$ control systems, the target system has index S_6 and communicates with S_5, S_4, where S_5 communicates with S_3, S_2 and S_4 communicates only with S_1.

We now use the parametric communication tree to describe a sequence of circuit equivalences. At each step we apply the equivalence in Fig. 2 to a partition of a subset of nodes, based on the subtrees to which they belong. To aid this process, we additionally denote for each $1 \leq i \leq K+1$ the following: let $P_i \subseteq \{1, \ldots, K\}$ be its children in the tree, let $1 \leq a_i \leq K$ be the smallest index of its children, let $0 \leq b_i \leq B$ be the number of its children, let $1 \leq p_i \leq K+1$ be its parent in the tree, and let $\hat{P}_i \subseteq \{1, \ldots, K\}$ be all the nodes in its subtree (including itself). By the chosen construction and numbering, it follows that $P_i = \left\{S_{a_i+1}, \ldots, S_{a_i+b_i}\right\}$.

Theorem *The circuit equivalence in Fig. 3 holds.*

We provide a proof by induction, following a chain of circuit equivalences aided by Fig. 2, as follows.

Proof In the first step we consider the entire controlled unitary gate. Its targets are the qubits $T_1, \ldots, T_M$ in system S_{K+1}, while its controls are the qubits $C_1, \ldots, C_N$ split across the systems $S_1, \ldots, S_K$. We partition the controls $C_1, \ldots, C_N$ into b_{K+1} sets based on the subtree to which their system belongs, i.e. corresponding to the controls in $\hat{P}_{a_{K+1}+1}, \ldots, \hat{P}_{a_{K+1}+b_{K+1}}$. We then place b_{K+1} Bell pairs, one between each pair of systems $S_{a_{K+1}+j}$

and S_{K+1} for $1 \leq j \leq b_{K+1}$. For each jth such Bell pair, we denotte by $O^{(a_{K+1}+j)}_{a_{K+1}+j}$ its half residing within $S_{a_{K+1}+j}$, and by $I^{(K+1)}_{a_{K+1}+j}$ its half residing within S_{K+1}. We now apply the equivalence in Fig. 2. This distributes the entire controlled unitary gate targeting S_{K+1}, decomposing it into $1 \leq b_{K+1} \leq B$ (possibly non-local) Toffoli gates each targeting one of the b_{K+1} systems $S_{a_{K+1}+j}$.

In the next steps, we perform a similar operation for each nonlocal Toffoli gate that has not yet been distributed. By induction, we assume (and, for the first step, we know) that every such gate corresponds to a subtree rooted at a system S_i, i.e. that the involved controls are exactly those belonging to the systems in this subtree, $\hat{P}_i$, and that the target is $O^{(i)}_i$. We partition these controls into $b_i + 1$ sets according to their belonging to $\hat{P}_{a_i+1}, \ldots, \hat{P}_{a_i+b_i}$ and S_i. We then place b_i Bell pairs, one between each pair of systems S_{a_i+j} and S_i for $1 \leq j \leq b_i$. For each jth such Bell pair, we denote by $O^{(a_i+j)}_{a_i+j}$ its half residing within S_{a_i+j}, and by $I^{(i)}_{a_i+j}$ its half residing within S_i. We now apply the equivalence in Fig. 2, with the first b_i sets acting non-locally, and the local $\mathbf{U}$ gate acting on the target and local controls in S_i. This distributes this nonlocal Toffoli gate controlled by the controls in $\hat{P}_i$ and S_i and targeting $O^{(i)}_i$, decomposing it into (possibly non-local) $1 \leq b_i \leq B$ Toffoli gates, each targeting one of the b_i systems S_{a_i+j}. Notice that this perpetuates the induction assumption, hence at the end of this process all of the gates are local.

Now, at the end of this process, the final circuit equivalence takes the form shown in Fig. 3, completing the proof. □

Results

The main insight of this paper is that not all K controls in a fully non-local K-control Toffoli gate need to be brought together. Instead, each site can compute and store the partial sum of one or more subsets of bits. In a chain topology, site i receives from site $i-1$ the product of the first $i-1$ bits, factors the ith bit into the product, and sends the result forward. This allows us to distribute the K-qubit Toffoli gate over a network comprised only of 3-qubit systems, at the cost of execution time numerically equal to K. Similarly, in a tree topology with uniform branching factor $B \geq 2$, site i receives from subordinate sites $i_1, \ldots, i_B$ the product of their assigned bits, factors them together with the ith bit, and sends the result forward to its superior. This allows us to distribute the K-qubit Tofffoli gate over a network comprised only of $(B+2)$-qubit systems, at the cost of execution time $\log_B K$. In particular, when every site has $B+2$ qubits, the execution time is logarithmic: $\log_{B+2} K$, and when the execution time is required to be a fixed constant t the qubit count is a fractional power, $K^{1/t}$.

Algorithm 1 Sequential cascade algorithm for distribution of controlled unitary gates

```
1:  procedure SeqInternalNode($i$)
2:      Establish Bell halves with child system $S_{i-1}$: $I^{(i)}_{i-1}$ and parent system $S_{i+1}$: $O^{(i)}_i$
3:      Wait classical correction $x_{i-1}$ from $S_{i-1}$
4:      If $x_i = 1$ apply classical correction $\mathbf{X}$ to qubit $I^{(i)}_{i-1}$
5:      Apply Toffoli gate with controls $I^{(i)}_{i-1}$ and $C^{(i)}_1, \ldots, C^{(i)}_{N_i}$ and target $O^{(i)}_i$
6:      Measure qubit $O^{(i)}_i$ in the computational basis (i.e. observable $\mathbf{Z}$)
7:      Send observed result $x_i$ to $S_{i+1}$
8:      Wait classical correction $z_i$ from $S_{i+1}$
9:      If $z_i = 1$ apply controlled $\mathbf{Z}$ correction targeting qubit $C^{(i)}_{N_i}$ controlled by qubits $C^{(i)}_1, \ldots, C^{(i)}_{N_i-1}$ and $I^{(i)}_{i-1}$
10:     Measure qubit $I^{(i)}_{i-1}$ in the angular basis (i.e. observable $\mathbf{X}$)
11:     Send observed result $z_{i-1}$ to $S_{i-1}$
12: end procedure
13: procedure SeqFinalNode
14:     Establish Bell halves with child system $S_K$: $I^{(K+1)}_K$
15:     Wait classical correction $x_K$ from $S_K$
16:     If $x_K = 1$ apply classical correction $\mathbf{X}$ to qubit $I^{(K+1)}_K$
17:     Apply controlled $\mathbf{U}$ gate with control $I^{(K+1)}_K$ and targets $T^{(K+1)}_1, \ldots, T^{(K+1)}_M$
18:     Measure qubit $I^{(K+1)}_K$ in the angular basis (i.e. observable $\mathbf{X}$)
19:     Send observed result $z_K$ to $S_K$
20: end procedure
```

Cascading Toffoli gates. The most qubit-efficient rendition of this argument comes in the form of a cascade of Toffoli gates. Based on the equivalence between the statement `if` $(\wedge_{j=1}^{N c_j}$==1) `then` $|\psi\rangle$=$\mathbf{U}|\psi\rangle$; and the cascade statement `if` $(\wedge_{j=1}^{N_1} c_j$==1) `then {... if` $(\wedge_{j=N-N_{K+1}}^{N} c_j$==1) `then` $|\psi\rangle$=$\mathbf{U}|\psi\rangle$`; ...}`, we note that the unitary gate in the equivalence in Fig. 2 can itself be a controlled unitary. We apply this observation K times, each time considering only the controls located at the ith site, and targeting not only the target qubits at

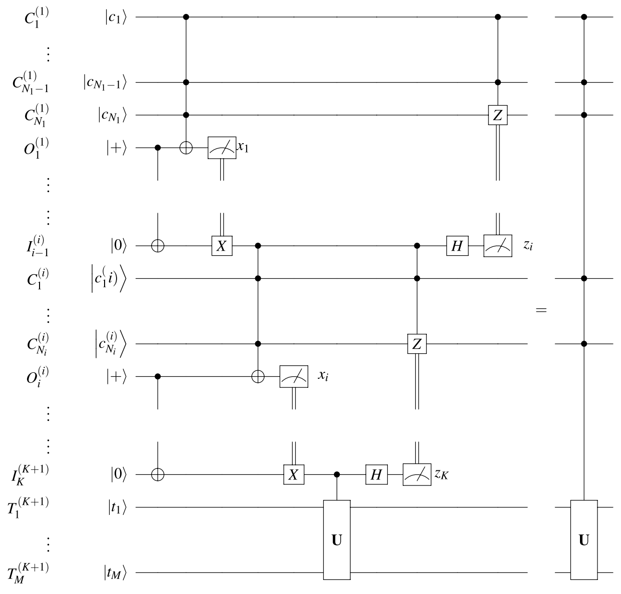

Figure 4. Cascade distribution of controlled unitaries.

site $K + 1$, but also the qubits in sites $i + 1, \ldots, K$. This produces the equivalence in Fig. 4, a simplified version of Fig. 3 adapted to the path topology.

Corollary *The circuit equivalence in Fig. 4 holds.*

Proof This immediately follows from the equivalence in Fig. 3 setting $B = 1$. □

As can be observed, the ith site must have $N_i + 2$ qubits for $1 < i \leq K$, the 1st site must have $N_1 + 1$ qubits, and the target site must have $M + 1$ qubits. In particular, to implement a non-local N-control Toffoli gate when all systems have n qubits we find $K = \lceil \frac{N-1}{n-2} \rceil$ computers. We can also present this protocol in the form of a distributed algorithm, specifying the actions at each of the local sites S_i, as shown in Algorithm 1 where all internal nodes $S_1, \ldots, S_K$ execute procedure SeqInternalNode and the root S_{K+1} executes SeqFinalNode. Using this protocol we can implement such a gate controlled by N=400,000 qubits using 133,333 5-qubit systems, but at a cost of a linear execution time, required by the fact that site $i + 1$ necessarily awaits for the input from site i before performing its computation. Alternatively, one can use 15,385 28-qubit IBM Falcon r1 systems, etc.

Parallel cascades.

To circumvent the linear execution time, we can set up a different communication tree, $G = (V, E)$, as described in the Methods section, where the vertices correspond to the sites $V = \{1, \ldots, K + 1\}$ with $K + 1$ being the root and the directed edges E are such that the degree of almost all vertices except the leaves is $B + 1$. The resulting circuit is shown in Fig. 3 and the corresponding equivalence is proven in the Methods section.

In this protocol, the time required for the execution of the gate is equal to the depth of the graph. If each site except the sources has an inner degree $B \geq 2$, this depth is $\lceil \log_B K \rceil$. We can also present this protocol in the form of a distributed algorithm, specifying the actions at each of the local sites S_i, as shown in Algorithm 2 where all internal nodes $S_1, \ldots, S_K$ execute procedure ParInternalNode and the root S_{K+1} executes ParFinalNode. In particular, to implement a non-local N-control Toffoli gate when all systems have n qubits we find $K = \lceil \frac{N+1-B}{n-B-1} \rceil$. Thus, using this protocol we can implement such a gate controlled by $N = 400,000$ qubits using 200, 000 5-qubit systems, in execution time 18τ where τ is the time required for a node's computation (i.e. the local Toffoli and Hadamard gates and communication with neighbors). Alternatively, one can use 16,667 27-qubit IBM Falcon systems, etc.

	Ancillary Qubits	Maximum Qubits Locally	EPR pairs consumed	Time
Existing Algorithms[8,9,12] (Fig. 1)	$2N$	$N+M$	N	$\mathcal{O}(1)$
Sequential Cascade (Fig. 4)	$2N$	$\max\{3,\, M+1\}$	N	$\mathcal{O}(N)$
Binary Parallel Cascades (Fig. 3)	$2N$	$\max\{4,\, M+2\}$	N	$\mathcal{O}(\log N)$
$\sqrt{N}$ Parallel Cascades (Fig. 3)	$2N$	$\lceil\sqrt{N}+M\rceil$	N	$\mathcal{O}(1)$

Table 1. Comparison of distribution protocols for a fully non-locally N-controlled unitary.

Algorithm 2 Parallel cascade algorithm for distribution of controlled unitary gates

```
1: procedure ParInternalNode($i$)
2:     Establish Bell halves with child systems $S_{a_i+1},\dots,S_{a_i+b_i}$: $I^{(i)}_{a_i+1},\dots,I^{(i)}_{a_i+b_i}$ and parent system $S_{p_i}$: $O^{(i)}_i$
3:     Wait classical corrections $x_{a_i+1},\dots,x_{a_i+b_i}$ from $S_{a_i+1},\dots,S_{a_i+b_i}$ respectively
4:     for $1\le j\le b_i$ do
5:         If $x_{a_i+j}=1$ apply classical correction $\mathbf{X}$ to qubit $I^{(i)}_{a_i+j}$
6:     end for
7:     Apply Toffoli gate with controls $I^{(i)}_{a_i+1},\dots,I^{(i)}_{a_i+b_i}$ and $C^{(i)}_1,\dots,C^{(i)}_{N_i}$ and target $O^{(i)}_i$
8:     Measure qubit $O^{(i)}_i$ in the computational basis (i.e. observable $\mathbf{Z}$)
9:     Send observed result $x_i$ to $S_{p_i}$
10:    Wait classical correction $z_i$ from $S_{p_i}$
11:    If $z_i=1$ apply controlled $\mathbf{Z}$ correction targeting qubit $C^{(i)}_{N_i}$ controlled by qubits $C^{(i)}_1,\dots,C^{(i)}_{N_i-1}$ and $I^{(i)}_{a_i+1},\dots,I^{(i)}_{a_i+b_i}$
12:    Measure qubits $I^{(i)}_{a_i+1},\dots,I^{(i)}_{a_i+b_i}$ in the angular basis (i.e. observable $\mathbf{X}$)
13:    Send observed results $z_{a_i+1,\dots,a_i+b_i}$ to $S_{a_i+1},\dots,S_{a_i+b_i}$ respectively
14: end procedure
15: procedure ParFinalNode
16:    Establish Bell halves with child systems $S_{a_{K+1}+1},\dots,S_{a_{K+1}+b_{K+1}}$: $I^{(K+1)}_{a_{K+1}+1},\dots,I^{(K+1)}_{a_{K+1}+b_{K+1}}$
17:    Wait classical corrections $x_{a_{K+1}+1},\dots,x_{a_{K+1}+b_{K+1}}$ from $S_{a_{K+1}+1},\dots,S_{a_{K+1}+b_i}$ respectively
18:    for $1\le j\le b_{K+1}$ do
19:        If $x_{aK+1i+j}=1$ apply classical correction $\mathbf{X}$ to qubit $I^{(i)}_{a_{K+1}+j}$
20:    end for
21:    Apply controlled $\mathbf{U}$ gate with controls $I^{(K+1)}_{a_{K+1}+1},\dots,I_{a_{K+1}+b_{K+1}}$ and targets $T^{(K+1)}_1,\dots,T^{(K+1)}_M$
22:    Measure qubits $I^{(K+1)}_{a_{K+1}+1},\dots,I_{a_{K+1}+b_{K+1}}$ in the angular basis (i.e. observable $\mathbf{X}$)
23:    Send observed results $z_{a_{K+1}+1},\dots,z_{a_{K+1}+b_{K+1}}$ to $S_{a_{K+1}+1},\dots,S_{a_{K+1}+b_{K+1}}$
24: end procedure
```

Discussion

In this subsection we will compare the existing distribution algorithms with the presented one in terms of execution time, number of required maximum local qubits, as well as number of entanglement pairs consumed. Next, we will discuss the applications and implications of the improved distribution algorithm.

We firstly consider the case of a fully non-locally N-controlled unitary having M target qubits. In this scenario, the protocols of[8,9,12] reduce to the circuit in Fig. 1. We compare these protocols with the cascade method in Fig. 4 and two variations of the parallel cascade method in Fig. 3: considering a balanced binary tree topology, and a balanced $\sqrt{N}$-tree topology. As it can be seen from Table 1, our protocols greatly reduce the requirement on local subsystem dimension, either to its square root while maintaining constant time, or even to a constant but at the cost of logarithmic time.

Now, we consider the case of a non-local N-controlled unitary where N controls are spread across K local subsystems and all M target qubits are consolidated into one subsystem. We denote the largest number of controls assigned to either of these subsystems as n. In this scenario, the protocols of[8,9] no longer coincide with that of[12]. We again compare the existing protocols with the cascade method in Fig. 4 and the same two variations of the parallel cascade method in Fig. 3. As it can be seen from Table 2, our protocols once again greatly reduce the requirement on local subsystem dimension even beyond[12], either to its square root while maintaining constant time, or even to a constant but at the cost of logarithmic time.

We visually represent the number of systems required to distribute a 400,000-controlled unitary gate using Algorithm 2 in Fig. 5, emphasizing the timeline of recent milestone hardware realization of quantum processors. For example, we can see that we would need a network of 200,000 5-qubit Canary r1 processors (or other

	Ancillary Qubits	Maximum Qubits Locally	EPR pairs consumed	Time
Full Clustering[8,9] (Fig. 1)	$2N$	$N+M$	N	$\mathcal{O}(1)$
Local Clustering[12] (Fig. 2)	$2K$	$\max\{n+1,\ K+M\}$	K	$\mathcal{O}(1)$
Sequential Cascade (Fig. 4)	$2K$	$\max\{n+2,\ M+1\}$	K	$\mathcal{O}(K)$
Binary Parallel Cascades (Fig. 3)	$2K$	$\max\{n+3,\ M+2\}$	K	$\mathcal{O}(\log K)$
$\sqrt{N}$ Parallel Cascades (Fig. 3)	$2K$	$\max\left\{\lceil n+\sqrt{K}\rceil,\ \lceil M+\sqrt{K}\rceil\right\}$	K	$\mathcal{O}(1)$

Table 2. Comparison of distribution protocols for a fully non-local N-controlled unitary.

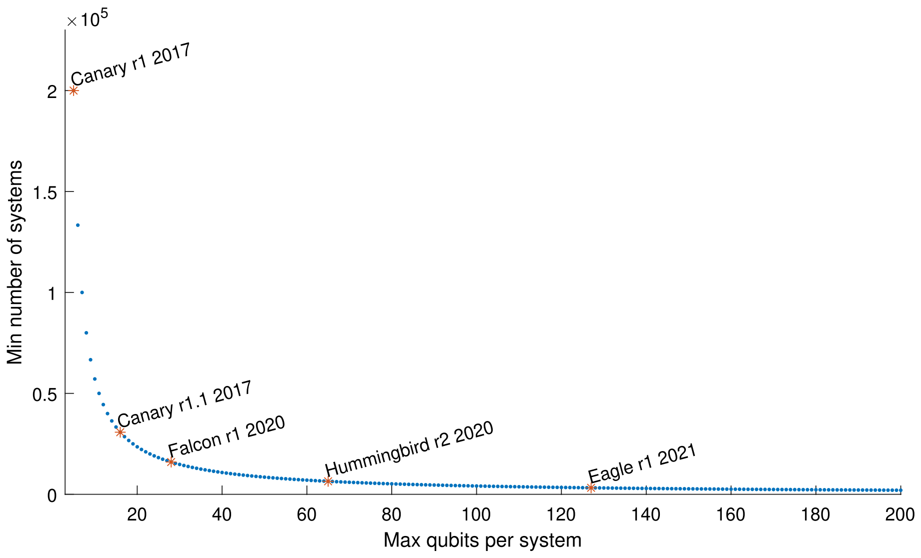

Figure 5. Minimal number of required systems to distribute a 400,000-controlled unitary using Algorithm 2 as required to factor a 2048-bit number using Shor's algorithm, emphasizing the timeline of IBM quantum processors[14].

equivalents, as commonly available on the market) or, equivalently instead, a network of 3,226 127-qubit IBM Eagle r1 processors.

In conclusion, the protocols presented in this paper compare favorably to the state of the art, greatly reducing restrictions on local system dimension. Consequently, if all quantum operations in Shor's algorithm such as modular exponentiation were to be decomposed as products of controlled unitaries, with a logarithmic (18x) increase in execution time, we could run Shor's algorithm to factor 2048-bit numbers using 200,000 5-qubit systems. This highlights the impact of future work related to the distribution of not just controlled unitaries, but actual quantum functions such as fast modular exponentiation which could lead to the experimental implementations of Shor's algorithm distributed across a network of small registry devices.

For future work we propose the analysis of the noise introduced by entanglement distribution across a network with tens of thousands of quantum computers, for example considering computation fidelity when using only diluted EPR states. In fact, it is not clear how this compares to how noise scales in large-scale quantum computers with hundreds of thousands of qubits.

Data availability

The datasets generated during the current study will be made available from the corresponding author on reasonable request.

Received: 27 September 2022; Accepted: 5 December 2022
Published online: 09 December 2022

References

1. Chen, L. *et al.* Report on post-quantum cryptography, vol. 12 (US Department of Commerce, National Institute of Standards and Technology 2016).
2. Building a superconducting quantum computer. Invited Talk PQCrypto 2014, October 2014 Waterloo, Canada. https://www.youtube.com/watch?v=wWHAs--HA1c Accessed: 2022-10-25.
3. Moody, D. *Nist status update on the 3rd round* (Cryptography Technology Group, National Institute of Standards and Technology, 2021).

4. Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science (IEEE, 1994).
5. Ha, J., Lee, J. & Heo, J. Resource analysis of quantum computing with noisy qubits for Shor's factoring algorithms. *Quantum Inf. Process.* **21**, 1–19 (2022).
6. Chow, J., Dial, O. & Gambetta, J. IBM quantum breaks the 100-qubit processor barrier. IBM Research Blog (2021).
7. Jiang, L., Taylor, J. M., Sørensen, A. S. & Lukin, M. D. Distributed quantum computation based on small quantum registers. *Phys. Rev. A* **76**, 062323 (2007).
8. Eisert, J., Jacobs, K., Papadopoulos, P. & Plenio, M. B. Optimal local implementation of nonlocal quantum gates. *Phys. Rev. A* **62**, 052317 (2000).
9. Tănăsescu, A., Mina, M.-Z. & Popescu, P. G. Non-local quantum functions and the distributed Deutsch-Jozsa algorithm. *Phys. Lett. A* **383**, 2168–2171 (2019).
10. Lee, S. M., Lee, S.-W., Jeong, H. & Park, H. S. Quantum teleportation of shared quantum secret. *Phys. Rev. Lett.* **124**, 060501 (2020).
11. Luo, M.-X. & Li, H.-R. Distributed quantum computation assisted by remote Toffoli gate. In International Conference on Cloud Computing and Security, 475–485 (Springer, 2016).
12. Sarvaghad-Moghaddam, M. & Zomorodi, M. A general protocol for distributed quantum gates. *Quantum Inf. Process.* **20**, 1–14 (2021).
13. Cuomo, D. *et al.* Optimized compiler for distributed quantum computing. arXiv preprint arXiv:2112.14139 (2021).
14. IBM quantum processor types. https://quantum-computing.ibm.com/composer/docs/iqx/manage/systems/processors. Accessed: 2022-09-20.

Author contributions

Conceptualization: A.T., D.C. and P.G.P.; methodology, A.T. and P.G.P.; validation, P.G.P.; writing—original draft preparation, A.T., D.C. and P.G.P.; writing—review and editing, A.T., and P.G.P.; supervision, P.G.P.; All authors have read and agreed to the published version of the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to P.G.P.

Reprints and permissions information is available at www.nature.com/reprints.