Quantum Information and Computation, Vol. 23, No. 1&2 (2023) 0027–0044
© Rinton Press

DISTRIBUTED SHOR'S ALGORITHM

LIGANG XIAO
Institute of Quantum Computing and Computer Theory, School of Computer Science and Engineering,
Sun Yat-sen University, Guangzhou, 510006, China
The Guangdong Key Laboratory of Information Security Technology,
Sun Yat-sen University, Guangzhou, 510006, China

DAOWEN QIU [a]
Institute of Quantum Computing and Computer Theory, School of Computer Science and Engineering,
Sun Yat-sen University, Guangzhou, 510006, China
The Guangdong Key Laboratory of Information Security Technology,
Sun Yat-sen University, Guangzhou, 510006, China
QUDOOR Technologies Inc., Guangzhou, 510006, China

LE LUO
School of Physics and Astronomy, Sun Yat-sen University
Zhuhai, 519082, China
QUDOOR Technologies Inc., Guangzhou, 510006, China

PAULO MATEUS
Instituto de Telecomunicações, Departamento de Matemática, Instituto Superior Técnico,
Av. Rovisco Pais 1049-001 Lisbon, Portugal

Received July 17, 2022
Revised November 17, 2022

Shor's algorithm is one of the most important quantum algorithm proposed by Peter Shor [Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134]. Shor's algorithm can factor a large integer with certain probability and costs polynomial time in the length of the input integer. The key step of Shor's algorithm is the order-finding algorithm, the quantum part of which is to estimate s/r, where r is the "order" and s is some natural number that less than r. Shor's algorithm requires lots of qubits and a deep circuit depth, which is unaffordable for current physical devices. In this paper, to reduce the number of qubits required and circuit depth, we propose a quantum-classical hybrid distributed order-finding algorithm for Shor's algorithm, which combines the advantages of both quantum processing and classical processing. In our distributed order-finding algorithm, we use two quantum computers with the ability of quantum teleportation separately to estimate partial bits of s/r. The measuring results will be processed through a classical algorithm to ensure the accuracy of the results. Compared with the traditional Shor's algorithm that uses multiple control qubits, our algorithm reduces nearly $L/2$ qubits for factoring an L-bit integer and reduces the circuit depth of each computer.

Keywords: Shor's algorithm, distributed Shor's algorithm, quantum-classcial hybrid, quantum teleportation, circuit depth

[a]Corresponding author (D. Qiu). *E-mail addresses:* issqdw@mail.sysu.edu.cn (D. Qiu)

1 Introduction

Quantum computing has shown great potential in some fields or problems, such as chemical molecular simulation [1], portfolio optimization [2], large number decomposition [3], unordered database search [4] and linear equation solving [5] et al. At present, there have been many useful algorithms in quantum computing [6], but to realize these algorithms requires the power of medium or large scale general quantum computers. However, it is still very difficult to develop a medium or large scale general quantum computer, because there are important physical problems in quantum computer that have not been solved. In the NISQ (Noisy Intermediate-Scale Quantum) era, we can only perform quantum algorithms with few qubits and low circuit depth. Therefore it is necessary to consider reducing the number of qubits and other computing resources required for quantum algorithms.

Distributed quantum computing is a computing method that solves problems collaboratively through multiple computing nodes. In distributed quantum computing, we can use multiple slightly smaller quantum computers to complete a task that was originally completed by a single medium or large scale quantum computer. Distributed quantum computing not only reduces the number of qubits required, but also sometimes reduces the circuit depth of each computer. This is also important since noise is increased with circuit being deepened . Therefore, distributed quantum computing has been studied significantly (for example, [7, 8, 9, 10]).

Shor's algorithm proposed by Peter Shor in 1994 [3] is an epoch-making discovery. It can factor a large integer with certain probability and costs time polynomial in the length of the input integer, whereas the time complexity of the best known classical algorithm for factoring large numbers is subexponential but superpolynomial. Shor's algorithm can be applied in cracking various cryptosystems, such as RSA cryptography and elliptic curve cryptography. For this reason, Shor's algorithm has received extensive attention from the community. However, recently some researchers have pointed out that using Shor's algorithm to crack the commonly used 2048-bit RSA integer requires physical qubits of millions [11]. So it is vital to consider reducing the logic qubits required in Shor's algorithm. Many researchers have been working on reducing the number of qubits required for Shor's algorithm [12, 13, 14], and these results have shown that Shor's algorithm can be implemented using only one control qubit to factor a L-bit integer together with $2L + c$ qubits and circuit depth $O(L^3)$, where c is a constant. But the method requires multiple intermediate measurements.

In 2004, Yimsiriwattana et al. [10] proposed a distributed Shor's algorithm. In this distributed algorithm, it directly divides the qubits into several parts, so each part has fewer qubits than the original one. Since all unitary operators can be decomposed into single qubit quantum gates and CNOT gates [15], they only need to consider how to implement CNOT gates acting on different parts, while a CNOT gate acting on different parts can be implemented by means of pre-sharing EPR pairs, local operations and classical communication. They clarified that their distributed algorithm needs to communicate $O(L^2)$ classical bits.

In this paper, we propose a new distributed Shor's algorithm. It is a quantum-classical hybrid algorithm, which not only takes advantage of fast quantum computing, but also takes advantage of the ease of processing measuring results of classical algorithms. In our distributed algorithm, two computers execute sequentially. Each computer estimates several bits of some key intermediate quantity (the ratio of s and r, where r is the "order" and s is some natural number that less than r). In order to guarantee the correlation between the two computers' measuring results to some extent, we employ quantum communication. Furthermore, to obtain high accuracy, we can adjust the measuring result of the first computer in terms of the measuring result of the second computer through classical post-processing. Compared with the traditional Shor's algorithm that uses multiple control qubits, our algorithm reduces the cost of qubits (reduces nearly $L/2$ qubits) and the circuit depth of each computer. Although each computer in our distributed algorithm requires more qubits than the implemention of Shor's algorithm mentioned above that uses only one control qubit, our method of using quantum communication to distribute the phase estimation of Shor's algorithm may be applicable to other quantum algorithms.

The remainder of the paper is organized as follows. In Section 2, we review quantum teleportation and some quantum algorithms related to Shor's algorithm. In Section 3, we present a distributed Shor's algorithm (more specifically, a distributed order-finding algorithm), and prove the correctness of our algorithm. In Section 4, we analyze the performance of our algorithm, including space complexity, time complexity, circuit depth and communication complexity. Finally in Section 5, we conclude with a summary.

2 Preliminaries

In this section, we review the quantum Fourier transform, phase estimation algorithm, order-finding algorithm and others we will use. We assume that the readers are familiar with the liner algebra and basic notations in quantum computing (for the details we can refer to [15]).

2.1 Quantum Fourier transform

Quantum Fourier transform is a unitary operator with the following action on the standard basis states:

$$QFT|j\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n}|k\rangle, \tag{1}$$

for $j = 0, 1, \cdots, 2^n - 1$. Hence the inverse quantum Fourier transform is acted as follows:

$$QFT^{-1} \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n}|k\rangle = |j\rangle, \tag{2}$$

for $j = 0, 1, \cdots, 2^n - 1$.

Quantum Fourier transform and the inverse quantum Fourier transform can be implemented by using $O(n^2)$ single qubit gates and $O(n^2)$ CNOT gates [3, 15].

2.2 Phase estimation algorithm

Phase estimation algorithm is an application of the quantum Fourier transform. Let $|u\rangle$ be a quantum state and let U be a unitary operator that satisfies $U|u\rangle = e^{2\pi i\omega}|u\rangle$ for some real number $\omega \in [0, 1)$. Suppose we can create the quantum state $|u\rangle$ and implement controlled operation $C_m(U)$ whose control qubits is m-qubit such that

$$C_m(U)|j\rangle|u\rangle = |j\rangle U^j|u\rangle \tag{3}$$

for any positive integer m and m-bit string j, where the first register is control qubits. Then we can apply phase estimation algorithm to estimate ω. Fig. 1 shows the implementation of $C_m(U)$. For the sake of convenience, we first define the following notations. In this paper, we treat bit strings and their corresponding binary integers as the same.

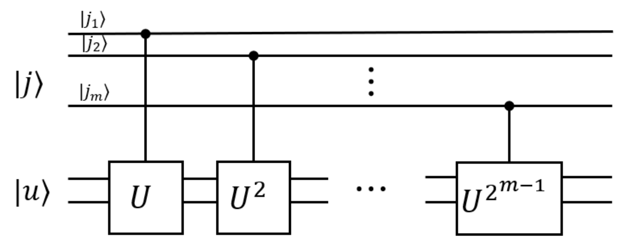

Fig. 1. Implementation for $C_m(U)$

Definition 1. For any real number $\omega = a_1 a_2 \cdots a_l.b_1 b_2 \cdots$, where $a_{k_1} \in \{0,1\}, k_1 = 1,2,\cdots,l$ and $b_{k_2} \in \{0,1\}, k_2 = 1,2,\cdots$, denote $|\psi_{t,\omega}\rangle, \omega_{\{i,j\}}, \omega_{[i,j]}$, and $d_t(x,y)$ respectively as follows:

- $|\psi_{t,\omega}\rangle$: for any positive integer t, $|\psi_{t,\omega}\rangle = QFT^{-1}\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1} e^{2\pi i j\omega}|j\rangle$.
- $\omega_{\{i,j\}}$: for any integer i,j with $1 \le i \le j$, $\omega_{\{i,j\}} = b_i b_{i+1} \cdots b_j$.
- $\omega_{[i,j]}$: for any integer i,j with $1 \le i \le j \le l$, $\omega_{[i,j]} = a_i a_{i+1} \cdots a_j$.
- $d_t(x,y)$: for any two t-bit strings (or t-bit binary integers) x,y, define $d_t(x,y) = \min(|x-y|, 2^t - |x-y|)$.

$d_t(\cdot,\cdot)$ is a useful distance to estimate the error of the algorithms in our paper and it has the following properties. We specify $a \bmod N = (kN+a) \bmod N$ for any negative integer a and positive integer N, where k is an integer and satisfies $kN + a \ge 0$.

Lemma 1. *Let t be a positive integer and let x,y be any two t-bit strings. It holds that:*
(I) *Let $B = \{b \in \{-(2^t-1),\cdots,2^t-1\} : (x+b) \bmod 2^t = y\}$. Then $d_t(x,y) = \min_{b\in B}|b|$.*
(II) *$d_t(\cdot,\cdot)$ is a distance on $\{0,1\}^t$.*
(III) *Let $t_0 < t$ be an positive integer. If $d_t(x,y) < 2^{t-t_0}$, then*

$$d_{t_0}(x_{[1,t_0]}, y_{[1,t_0]}) \le 1. \tag{4}$$

Proof. First we prove (I). It is clear for the case of $x = y$. Without loss of generality, assume $x > y$. Since $x \ne y$, we have B contains only 2 elements. Note that

$$x + (y-x) \bmod 2^t = y, \tag{5}$$
$$x + (2^t - (x-y)) \bmod 2^t = y, \tag{6}$$
$$|y-x| \le 2^t - 1, \tag{7}$$
$$|2^t - (x-y)| \le 2^t - 1 \tag{8}$$

and $y - x \ne 2^t - (x-y)$, we get that $y-x$ and $2^t-(x-y)$ are exactly two elements of B. Hence $\min_{b\in B}|b| = \min(|x-y|, 2^t - |x-y|) = d_t(x,y)$. Thus (I) holds.

Then we prove (II). We just need to show that $d_t(\cdot,\cdot)$ satisfies the triangle inequality, that is, $d_t(x,y) \le d_t(x,z) + d_t(z,y)$ holds for any t-bit string z. By (I), we know that there exists $b_1, b_2 \in \{-(2^t-1), \cdots, 2^t-1\}$ such that

$$|b_1| = d_t(x,z), |b_2| = d_t(z,y), \tag{9}$$

and

$$(x+b_1) \bmod 2^t = z, (z+b_2) \bmod 2^t = y. \tag{10}$$

Hence $(x+b_1+b_2) \bmod 2^t = y$. Then by (I) again, we have

$$d_t(x,y) \le |b_1+b_2| \le |b_1| + |b_2| = d_t(x,z) + d_t(z,y). \tag{11}$$

Thus, (II) holds.

Finally we prove (III). By (I) and $d_t(x,y) < 2^{t-t_0}$, we know that there exists an integer b with $|b| < 2^{t-t_0}$ such that

$$(2^{t-t0}x_{[1,t_0]} + x_{[t_0+1,t]} + b) \bmod 2^t = 2^{t-t0}y_{[1,t_0]} + y_{[t_0+1,t]}. \tag{12}$$

Then by (I) again we have

$$d_t(2^{t-t0}x_{[1,t_0]}, 2^{t-t0}y_{[1,t_0]}) \le |b + x_{[t_0+1,t]} - y_{[t_0+1,t]}| < 2 \cdot 2^{t-t0}. \tag{13}$$

Hence

$$d_{t_0}(x_{[1,t_0]}, y_{[1,t_0]}) < 2. \tag{14}$$

Therefore Eq. (4) holds. □

We can understand $d_t(\cdot,\cdot)$ in a more intuitive way. We place numbers 0 to 2^t evenly on a circumference where 0 and 2^t coincide. Suppose the distance of two adjacent points on the circumference is 1. Then $d_t(x,y)$ can be regarded as the length of the shortest path on the circumference from x to y. Next we review the phase estimation algorithm (see Algorithm 1) and its associated results.

Algorithm 1 Phase estimation algorithm

Procedure:

1: Create initialize state $|0\rangle^{\otimes t}|u\rangle$.

2: Apply $H^{\otimes t}$ to the first register:

$$H^{\otimes t}|0\rangle^{\otimes t}|u\rangle = \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle|u\rangle.$$

3: Apply $C_t(U)$:

$$C_t(U)\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle|u\rangle = \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle U^j|u\rangle = \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle e^{2\pi ij\omega}|u\rangle.$$

4: Apply QFT^{-1}:

$$QFT^{-1}\frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}e^{2\pi ij\omega}|j\rangle|u\rangle = |\psi_{t,\omega}\rangle|u\rangle.$$

5: Measure the first register:
obtain a t-bit string $\widetilde{\omega}$.

If the fractional part of ω does not exceed t bits (i.e. $2^t\omega$ is an integer), by observing Eq. (2) and the step 4 in Algorithm 1, we can see that $\widetilde{\omega}$ is a perfect estimate of ω (i.e. $\frac{\widetilde{\omega}}{2^t} = \omega$). However, sometimes ω is not approximated by $\frac{\widetilde{\omega}}{2^t}$ but is approximated by $1 - \frac{\widetilde{\omega}}{2^t}$. For example, if the binary representation of ω is $\omega = 0.11\cdots1$ (sufficiently many 1s), we will obtain the measuring result $00\cdots0$ with high probability, since at this time $e^{2\pi i\omega}$ is close to $e^{2\pi i0} = 1$. The output $\widetilde{\omega}$ of the phase estimation algorithm should satisfy that $\frac{\widetilde{\omega}}{2^t}$ is close to ω or $\omega - 1$. We have the following results.

Proposition 1 (See [15]). *In* Algorithm 1, *for any $\epsilon > 0$ and any positive integer n, if $t = n + \lceil\log_2(2 + \frac{1}{2\epsilon})\rceil$, then the probability of $d_t(\widetilde{\omega}, \omega_{\{1,t\}}) < 2^{t-n}$ is at least $1-\epsilon$.*

Lemma 2. *For any t-bit string $\widetilde{\omega}$ and real number $\omega \in [0, 1)$. If $d_t(\widetilde{\omega}, \omega_{\{1,t\}}) < 2^{t-n}$, then we have $|\frac{\widetilde{\omega}}{2^t} - \omega| \leq 2^{-n}$ or $1 - |\frac{\widetilde{\omega}}{2^t} - \omega| \leq 2^{-n}$, where $n < t$.*

Proof. Since $|2^t\omega - \omega_{\{1,t\}}| < 1$, if $d_t(\widetilde{\omega}, \omega_{\{1,t\}}) = |\widetilde{\omega} - \omega_{\{1,t\}}|$, we have

$$|\widetilde{\omega} - 2^t\omega| \leq |\widetilde{\omega} - \omega_{\{1,t\}}| + |\omega_{\{1,t\}} - 2^t\omega| \leq 2^{t-n}, \tag{15}$$

and thus $|\frac{\widetilde{\omega}}{2^t} - \omega| \leq 2^{-n}$; if $d_t(\widetilde{\omega}, \omega_{\{1,t\}}) = 2^t - |\widetilde{\omega} - \omega_{\{1,t\}}|$, we have

$$2^t - |\widetilde{\omega} - 2^t\omega| \leq 2^t - (|\widetilde{\omega} - \omega_{\{1,t\}}| - |\omega_{\{1,t\}} - 2^t\omega|) \leq 2^{t-n}, \tag{16}$$

and therefore, we have $1 - |\frac{\widetilde{\omega}}{2^t} - \omega| \leq 2^{-n}$. □

That is to say, if $\frac{\widetilde{\omega}}{2^t}$ is an estimate of $\omega_{\{1,t\}}$ with error less than 2^{-n}, then $\frac{\widetilde{\omega}}{2^t}$ is an estimate of ω with error no larger than 2^{-n}.

2.3 Order-finding algorithm

Phase estimation algorithm is a key subroutine in order-finding algorithm. Given an L-bit integer N and a positive integer a with $gcd(a, N) = 1$, the purpose of order-finding algorithm is to find the order r of a modulo N, that is, the least integer r that satisfies $a^r \equiv 1(\bmod\ N)$. An important unitary operator M_a in order-finding algorithm is defined as

$$M_a|x\rangle = |ax \mod N\rangle. \tag{17}$$

Denote

$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi i\frac{s}{r}k}|a^k \bmod\ N\rangle. \tag{18}$$

We have

$$M_a|u_s\rangle = e^{2\pi i\frac{s}{r}}|u_s\rangle, \tag{19}$$

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle, \tag{20}$$

and

$$\langle u_s|u_{s'}\rangle = \delta_{s,s'} = \begin{cases} 0 & \text{if } s \neq s', \\ 1 & \text{if } s = s'. \end{cases} \tag{21}$$

So if we expect to apply phase estimation algorithm in finding order, the key is to construct $C_m(M_a)$, that is, for any m-bit string j,

$$C_m(M_a)|j\rangle|x\rangle = |j\rangle|a^j x \bmod N\rangle. \tag{22}$$

Algorithm 2 [15] and Fig. 2 show the precedure of order-finding algorithm.

Algorithm 2 Order-finding algorithm

Input: Positive integers N and a with $gcd(N, a) = 1$.
Output: The order r of a modulo N.
Procedure:

1: Create initial state $|0\rangle^{\otimes t}|1\rangle$:
$t = 2L + 1 + \lceil \log_2(2 + \frac{1}{2\epsilon})\rceil$ and the second register has L qubits.

2: Apply $H^{\otimes t}$ to the first register:
$H^{\otimes t}|0\rangle^{\otimes t}|1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$.

3: Apply $C_t(M_a)$:
$C_t(M_a)\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle M^j(\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle) = \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} |j\rangle e^{2\pi i j \frac{s}{r}}|u_s\rangle$.

4: Apply QFT^{-1}:
$QFT^{-1}\frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} |j\rangle e^{2\pi i j \frac{s}{r}}|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t,s/r}\rangle|u_s\rangle$

5: Measure the first register:
obtain a t-bit string m that is an estimation of $\frac{s}{r}$ for some s.

6: Apply continued fractions algorithm:
obtain r.

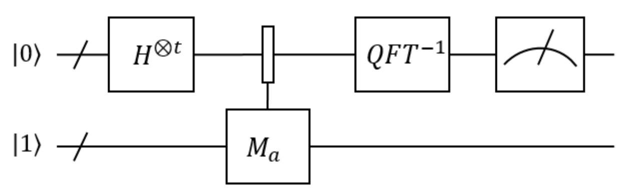

Fig. 2. Circuit for order-finding algorithm

The purpose of the quantum part of the order-finding algorithm (steps 1 to 5 in Algorithm 2) is to get a measuring result m such that m is an estimation of $\frac{s}{r}$ with error no large than $2^{-(2L+1)}$ for some $s \in \{0, 1, \cdots, r-1\}$ (i.e. $|\frac{m}{2^t} - \frac{s}{r}| \leq 2^{-(2L+1)}$), because it is one of the prerequisites to ensure the correctness of the result in step 6 [15]. Let $\{P_i\}$ be any projective measurement on $\mathbb{C}^{2^t}$ and let $|\phi_s\rangle$ be any t-qubit quantum state for $s = 0, 1, \cdots, r-1$. By Eq. (21), we have

$$\|(P_j \otimes I)\sum_{s=0}^{r-1}|\phi_s\rangle|u_s\rangle\|^2 = \sum_{s=0}^{r-1}\|(P_j|\phi_s\rangle)|u_s\rangle\|^2 \tag{23}$$

for $P_j \in \{P_i\}$. Hence by Propositon 1 and Eq. (23), we can obtain the following proposition immediately.

Proposition 2 (See [15]). *In* Algorithm 2*, the probability of* $d_t(m, (\frac{s}{r})_{\{1,t\}}) < 2^{t-(2L+1)}$ *for any fixed* $s \in \{0, 1, \cdots, r-1\}$ *is at least* $\frac{1-\epsilon}{r}$*. And the probability that there exists an* $s \in \{0, 1, \cdots, r-1\}$ *such that*

$$d_t(m, (\frac{s}{r})_{\{1,t\}}) < 2^{t-(2L+1)} \tag{24}$$

is at least $1-\epsilon$*.*

Proof. Denote $A_s = \{x \in \{0,1\}^t : d_t(x, (\frac{s}{r})_{\{1,t\}}) < 2^{t-(2L+1)}\}$. Let $Q_s = \sum_{i\in A_s}|i\rangle\langle i|$. For any fixed $s \in \{0, 1, \cdots, r-1\}$, the probability of $d_t(m, (\frac{s}{r})_{\{1,t\}}) < 2^{t-(2L+1)}$ is

$$\begin{aligned}
\|(Q_s \otimes I)\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\psi_{t,k/r}\rangle|u_k\rangle\|^2 &= \frac{1}{r}\sum_{k=0}^{r-1}\|(Q_s|\psi_{t,k/r}\rangle)|u_k\rangle\|^2 \text{ (by Eq. (23))} && (25)\\
&\geq \frac{1}{r}\|(Q_s|\psi_{t,s/r}\rangle)|u_s\rangle\|^2 && (26)\\
&= \frac{1}{r}\|(Q_s|\psi_{t,s/r}\rangle)\|^2 && (27)\\
&\geq \frac{1-\epsilon}{r} \text{ (by Propositon 1)} && (28)
\end{aligned}$$

Let $Q = \sum_{i\in \bigcup_{s=0}^{r-1} A_s}|i\rangle\langle i|$. And the probability that there exists an $s \in \{0, 1, \cdots, r-1\}$ such that $d_t(m, (\frac{s}{r})_{\{1,t\}})) < 2^{t-(2L+1)}$ is

$$\begin{aligned}
\|(Q \otimes I)\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|\psi_{t,k/r}\rangle|u_k\rangle\|^2 &= \frac{1}{r}\sum_{k=0}^{r-1}\|(Q|\psi_{t,k/r}\rangle)|u_k\rangle\|^2 \text{ (by Eq.(23))} && (29)\\
&\geq \frac{1}{r}\sum_{k=0}^{r-1}\|(Q_k|\psi_{t,k/r}\rangle)|u_k\rangle\|^2 && (30)\\
&= \frac{1}{r}\sum_{k=0}^{r-1}\|(Q_k|\psi_{t,k/r}\rangle)\|^2 && (31)\\
&\geq 1-\epsilon \text{ (by Equation (28))} && (32)
\end{aligned}$$

Therefore, the proposition holds. $\square$

Although it is an important part to discuss the probability of obtaining r correctly from the measuring result by applying continued fractions algorithm, the details are omitted here and we focus on considering whether the measuring result is an estimation of $\frac{s}{r}$ for some s (i.e. $|\frac{m}{2^t} - \frac{s}{r}| \leq 2^{-(2L+1)}$), since this is exactly the goal of the quantum part in the order-finding algorithm.

2.4 Quantum teleportation

Quantum teleportation is an important means to realize quantum communication [16, 17]. Quantum teleportation is effectively equivalent to physically teleporting qubits, but in fact, the realization of quantum teleportation only requires classical communication and both parties to share an EPR pair in advance. The following result is useful.

Theorem 1 ([16]). *When Alice and Bob share L pairs of EPR pairs, they can simulate transmitting L qubits by communicating $2L$ classical bits.*

3 Distributed order-finding algorithm

In [9], a distributed phase estimation algorithm was proposed. However, in the method of [9], it is not proved that the distance between the estimation and the true value can be less than the given margin of error. Their ideas deserve further consideration. In this section, by combining with quantum teleportation, we proposed a distributed order-finding algorithm and prove the correctness of our algorithm.

Without loss of generality, assume that $L = \lceil \log_2(N) \rceil$ is even. The idea of our distributed order-finding algorithm is as follows. We need two quantum computers (named A and B). We first apply order-finding algorithm in computer A and obtain an estimation of the first $\frac{L}{2}+1$ bits of $\frac{s}{r}$ for some $s \in \{0, 1, \cdots, r-1\}$, and similarly obtain an estimation of the $(\frac{L}{2}+2)$th bit to $(2L+1)$th bit of $\frac{s}{r}$ in computer B. We can realize this by using $C_t(M_a^{2^l})$, since $M_a^{2^l}|u_s\rangle = e^{2\pi i(2^l \frac{s}{r})}|u_s\rangle$ and the fractional part of $2^l\frac{s}{r}$ starts at the $(l+1)$th bit of he fractional part of $\frac{s}{r}$. Moreover, since $M_a^{2^l} = M_{a^{2^l} \bmod N}$ and we can calculate $a^{2^l} \mod N$ classically with time complexity $O(l)$, we can construct $C_t(M_a^{2^l})$ with the same way as $C_t(M_a)$. In addition, to guarantee the measuring results of A and B corresponding to the same $\frac{s}{r}$, we need quantum teleportation.

However, in order to make the distance between the overall estimation and the true value less than $\frac{1}{2^{2L+1}}$, computer B actually estimates the $\frac{L}{2}$th bit to $(2L+1)$th bit, where the estimation of the $\frac{L}{2}$th bit and the $(\frac{L}{2}+1)$th bit is used to “correct” the measuring result of A. This “correction” operation is handed over to a classical subroutine named $CorrectResults$. Our distributed order-finding algorithm is shown in Algorithm 3 and Fig. 3, and the subroutine $CorrectResults$ is shown in Algorithm 4. In addition, we give an example to show the procedure of our distrbuted order-finding algorithm in Appendix A.

Remark 1. Although Algorithm 3 is a serial algorithm, the two computers can also execute in parallel to some extent. For example, execute the algorithm in the following order: 1, (2, 6), 3, 5, 7, (4, 8), 9, 10, 11, where i represents the ith step in Algorithm 3, and (i, j) means that the ith and jth steps are executed in parallel.

Remark 2. If we initialize the quantum state of computer B to $|0\rangle_B|1\rangle_D$ (register D is L-qubit) and

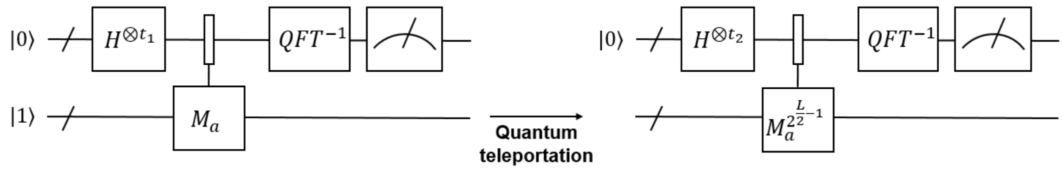

Fig. 3. Circuit for distributed order finding algorithm

do not employ quantum teleportation in Fig. 3, that is, computer A and computer B execute "partial" order-finding algorithm respectively, then the final quantum states of computers A and B will become $\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle_A|u_s\rangle_C$ and $\sum_{s=0}^{r-1}|\widetilde{s/r}'\rangle_B|u_s\rangle_D$, respectively, where $\widetilde{s/r}$ is an estimation of $(\frac{s}{r})_{\{1,\frac{L}{2}+1\}}$ and $\widetilde{s/r}'$ is an estimation of $(\frac{s}{r})_{\{\frac{L}{2},2L+1\}}$. Therefore, in this case, if computer A measures register A and computer B measures register B, their measuring results may not correspond to the same s/r.

Next we prove the correctness of our algorithm, that is, we can obtain an output m such that $|\frac{m}{2^{(2L+1+p)}}-\frac{s}{r}|\leq 2^{-(2L+1)}$ holds for some $s\in\{0,1,\cdots,r-1\}$ with high probability. Let $r,L,t_1,t_2,p,m_1,m_2,m_{prefix},m,\epsilon',|\phi_{final}\rangle$ be the same as those in Algorithm 3 and Algorithm 4. We first prove that if m_1 and m_2 are both estimations of some bits of $\frac{s_0}{r}$ with $\frac{s_0}{r}=0.a_1a_2\cdots a_{\frac{L}{2}+1}$, then the output m is perfect (i.e. $m=a_1a_2\cdots a_{\frac{L}{2}}a_{\frac{L}{2}+1}0\cdots0$), and the probability of this case is not less than $\frac{1}{r}$.

Proposition 3. *Let* $s_0\in\{0,1,\cdots,r-1\}$ *satisfy that* $2^{\frac{L}{2}+1}\cdot\frac{s_0}{r}$ *is an integer, that is,* $\frac{s_0}{r}=0.a_1a_2\cdots a_{\frac{L}{2}+1}$ *where* $a_i\in\{0,1\}$, $i=1,2,\cdots,\frac{L}{2}+1$. *Then in* Algorithm 3*, it holds that*

$$\text{Prob}(m=a_1a_2\cdots a_{\frac{L}{2}+1}0\cdots0)\geq\frac{1}{r}. \tag{33}$$

Proof. Since the fractional part of $\frac{s_0}{r}$ is at most $(\frac{L}{2}+1)$-bit, in Algorithm 3, we have

$$|\psi_{t_1,s_0/r}\rangle=|a_1a_2\cdots a_{\frac{L}{2}+1}0\cdots0\rangle \tag{34}$$

and

$$|\psi_{t_2,2^{\frac{L}{2}-1}s_0/r}\rangle=|a_{\frac{L}{2}}a_{\frac{L}{2}+1}0\cdots0\rangle. \tag{35}$$

Denote $x=a_1a_2\cdots a_{\frac{L}{2}}a_{\frac{L}{2}+1}0\cdots0$ and $y=a_{\frac{L}{2}}a_{\frac{L}{2}+1}0\cdots0$. By Eq. (23), we have

$$\begin{aligned}\text{Prob}(m_1=x\text{ and }m_2=y)&=\||x\rangle\langle x|\otimes|y\rangle\langle y|\otimes I\;|\phi_{final}\rangle\|^2 &(36)\\ &\geq\||x\rangle\langle x|\otimes|y\rangle\langle y|\otimes I\;\frac{1}{\sqrt{r}}|\psi_{t_1,s_0/r}\rangle|\psi_{t_2,2^{\frac{L}{2}-1}s_0/r}\rangle|u_{s_0}\rangle\|^2 &(37)\\ &=\frac{1}{r}. &(38)\end{aligned}$$

Since $CorrectResults(x,y)=a_1a_2\cdots a_{\frac{L}{2}}a_{\frac{L}{2}+1}0\cdots0$, the lemma holds. $\square$

Algorithm 3 Distributed order-finding algorithm

Input: Positive integers N and a with $gcd(N, a) = 1$.
Output: The order r of a modulo N.
Procedure:

1: Computer A creates initial state $|0\rangle_A|1\rangle_C$. Computer B creates initial state $|0\rangle_B$:
Here registers A, B and C are t_1-qubit, t_2-qubit and L-qubit, respectively. We take $t_1 = \frac{L}{2} + 1 + p$ and $t_2 = \frac{3L}{2} + 2 + p$, where $p = \lceil \log_2(2 + \frac{1}{2\epsilon'}\rceil$ and $\epsilon' = \frac{\epsilon}{2}$.
Computer A:
2: Apply $H^{\otimes t_1}$ to register A: $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} (H^{\otimes t_1}|0\rangle_A|u_s\rangle_C)|0\rangle_B$.
3: Apply $C_{t_1}(M_a)$ to registers A and C: $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} (\frac{1}{\sqrt{2^{t_1}}} \sum_{j=0}^{2^{t_1}-1} e^{2\pi i j \frac{s}{r}}|j\rangle_A|u_s\rangle_C)|0\rangle_B$.
4: Apply $QFT^{-1}$to register A: $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t_1,s/r}\rangle_A|u_s\rangle_C|0\rangle_B$
5: Teleport the qubits of register C to computer B: $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t_1,s/r}\rangle_A|0\rangle_B|u_s\rangle_C$
Computer B:
6: Apply $H^{\otimes t_2}$ to register B: $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t_1,s/r}\rangle_A H^{\otimes t_2}|0\rangle_B|u_s\rangle_C$
7: Apply $C_{t_2}(M_a^{2^{\frac{L}{2}-1}})$ to registers B and C:
$\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t_1,s/r}\rangle_A (\frac{1}{\sqrt{2^{t_2}}} \sum_{j=0}^{2^{t_2}-1} e^{2\pi i j(2^{\frac{L}{2}-1}\frac{s}{r})}|j\rangle_B)|u_s\rangle_C$
8: Apply $QFT^{-1}$to register B: $\rightarrow |\phi_{final}\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_{t_1,s/r}\rangle_A|\psi_{t_2,2^{\frac{L}{2}-1}s/r}\rangle_B|u_s\rangle_C$
9: Computer A measures register A and computer B measures register B:
A obtains a t_1-bit string m_1 and B obtains a t_2-bit string m_2.
10: $m \leftarrow CorrectResults(m_1, m_2)$: m is a $(2L + 1 + p)$-bit string.
11: Apply continued fractions algorithm: obtain r.

Algorithm 4 CorrectResults subroutine

Input: Two measuring results: t_1-bit string m_1 and t_2-bit string m_2.
Output: An estimation m such that $|\frac{m}{2^{(2L+1+p)}} - \frac{s}{r}| \leq 2^{-(2L+1)}$ for some $s \in \{0, 1, \cdots, r-1\}$.
Procedure:

1: Choose $CorrectionBit \in \{-1, 0, 1\}$ such that
$((m_1)_{[\frac{L}{2},\frac{L}{2}+1]} + CorrectionBit) \bmod 2^2 = (m_2)_{[1,2]}$.
2: $m_{prefix} \leftarrow ((m_1)_{[1,\frac{L}{2}+1]} + CorrectionBit) \bmod 2^{\frac{L}{2}+1}$
3: $m \leftarrow m_{prefix} \circ (m_2)_{[3,t_2]}$ ("$\circ$" represents catenation)
4: return m

Then we prove that if m_2 is an estimation of the $\frac{L}{2}$th to $(2L+1)$th bit of $\frac{s_0}{r}$, we can get $(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2},\frac{L}{2}+1\}}$.

Lemma 3. *Let* $s_0 \in \{0, 1, \cdots, r-1\}$ *satisfy that* $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ *is not an integer and let* m_2 *satisfy*

$$d_{t_2}(m_2, (\frac{s_0}{r})_{\{\frac{L}{2}, 2L+1+p\}}) < 2^p. \tag{39}$$

Then $(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2}, \frac{L}{2}+1\}}$.

Proof. Since $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ is not an integer, we have

$$2^{-L} < \frac{1}{r} \leq \frac{2^{\frac{L}{2}+1} s_0 \bmod r}{r} \leq \frac{r-1}{r} < 1 - 2^{-L}. \tag{40}$$

So we get $(\frac{s_0}{r})_{\{\frac{L}{2}+2, \frac{3L}{2}+1\}}$ is not $00\cdots0$ or $11\cdots1$. Hence, $(\frac{s_0}{r})_{\{\frac{L}{2}+2, 2L+1\}}$ is not $00\cdots0$ or $11\cdots1$. That is to say, if we add or subtract 1 to $(\frac{s_0}{r})_{\{\frac{L}{2}, 2L+1\}}$, its first two bits are not changed. Thus by Eq. (39), we have

$$(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2}, \frac{L}{2}+1\}} \tag{41}$$

Therefore the lemma holds. □

If $(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2}, \frac{L}{2}+1\}}$, that is, the first two bits of m_2 are correct, then we can use these two bits of m_2 to "correct" m_1. The following lemma can be used to show the correctness of Algorithm 4.

Lemma 4. *Let* $t > 2$ *be a positive integer and let* x, y *be two* t*-bit strings with* $d_t(x, y) \leq 1$*. Then there only exists one element* b_0 *in* $\{-1, 0, 1\}$ *such that* $(x + b_0) \bmod 2^t = y$*, and for any* $b \in \{-1, 0, 1\}$*,* $(x + b) \bmod 2^t = y$ *if and only if* $(x_{[t-1,t]} + b) \bmod 2^2 = y_{[t-1,t]}$*.*

Proof. By Lemma 1, we know that there exists such a b_0. It is clear that such a b_0 is unique. Next we prove that for any $b \in \{-1, 0, 1\}$, $(x + b) \bmod 2^t = y$ if and only if $(x_{[t-1,t]} + b) \bmod 2^2 = y_{[t-1,t]}$. For any $b \in \{-1, 0, 1\}$, suppose $(x + b) \bmod 2^t = y$, then we have

$$(x + b) \bmod 2^2 = y \bmod 2^2. \tag{42}$$

That is,

$$(x_{[t-1,t]} + b) \bmod 2^2 = y_{[t-1,t]}. \tag{43}$$

On the other hand, for any $b \in \{-1, 0, 1\}$, suppose $(x_{[t-1,t]} + b) \bmod 2^2 = y_{[t-1,t]}$. Since there only exists one elements b_1 in $\{-1, 0, 1\}$ such that $(x_{[t-1,t]} + b_1) \bmod 2^2 = y_{[t-1,t]}$, b is equal to b_0, that is, b satisfies $(x + b) \bmod 2^t = y$. Consequently, the lemma holds. □

We can inspect Lemma 4 from another aspect. If $d_{\frac{L}{2}+1}(m_1, (\frac{s_0}{r})_{\{1, \frac{L}{2}+1\}}) \leq 1$ and $(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2}, \frac{L}{2}+1\}}$ hold for some s_0, then the $CorrectionBit$ in Algorithm 4 exists, and $m_{prefix} = (\frac{s_0}{r})_{\{1, \frac{L}{2}+1\}}$ holds as well.

Finally, we give the following results, which completes the proof of the correctness of our algorithms.

Proposition 4. *Let m_2 satisfy $d_{t_2}(m_2, (\frac{s_0}{r})_{\{\frac{L}{2}, 2L+1+p\}}) < 2^p$ for some $s_0 \in \{0, 1, \cdots, r-1\}$ with $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ being not an integer. Suppose $d_{t_1}(m_1, (\frac{s_0}{r})_{\{1,t_1\}}) < 2^p$. Then $|\frac{m}{2^{2L+1+p}} - \frac{s_0}{r}| \leq 2^{-(2L+1)}$.*

Proof. Since $d_{t_2}(m_2, (\frac{s_0}{r})_{\{\frac{L}{2}, 2L+1+p\}}) < 2^p$ and $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ is not an integer, by Lemma 3, we have

$$(m_2)_{[1,2]} = (\frac{s_0}{r})_{\{\frac{L}{2}, \frac{L}{2}+1\}}. \tag{44}$$

Since $d_{t_1}(m_1, (\frac{s_0}{r})_{\{1,t_1\}}) < 2^p$ and $t_1 = \frac{L}{2} + 1 + p$, by Lemma 1, we have

$$d_{\frac{L}{2}+1}((m_1)_{[1,\frac{L}{2}+1]}, (\frac{s_0}{r})_{\{1,\frac{L}{2}+1\}}) \leq 1. \tag{45}$$

As a result, in Algorithm 4, the $CorrectionBit$ exists. By Eq. (44), Lemma 4, and the steps 1 to 2 in Algorithm 4, we get

$$m_{prefix} = (\frac{s_0}{r})_{\{1,\frac{L}{2}+1\}}. \tag{46}$$

Since $m = m_{prefix} \circ (m_2)_{[3,t_2]}$, by Eq. (44) and Eq. (46), we get

$$d_{2L+1+p}(m, (\frac{s_0}{r})_{\{1,2L+1+p\}}) = d_{\frac{3L}{2}+2+p}(m_2, (\frac{s_0}{r})_{\{\frac{L}{2},2L+1+p\}}) < 2^p. \tag{47}$$

Since $\frac{s_0}{r}$ is not an integer, similar to Eq. (40), we know that $(\frac{s_0}{r})_{\{1,2L+1\}}$ is not $00\cdots0$ or $11\cdots1$. Then by Eq. (47), we get $d_{2L+1+p}(m, (\frac{s_0}{r})_{\{1,2L+1+p\}}) = |m - (\frac{s_0}{r})_{\{1,2L+1+p\}})|$. Therefore, by Eq. (47) and Lemma 2, we obtain

$$|\frac{m}{2^{2L+1+p}} - \frac{s_0}{r}| \leq 2^{-(2L+1)}. \tag{48}$$

□

Theorem 2. *In* Algorithm 3*, for any fixed $s_0 \in \{0, 1, \cdots, r-1\}$, the probability of $|\frac{m}{2^{2L+1+p}} - \frac{s_0}{r}| \leq 2^{-(2L+1)}$ is at least $\frac{1-\epsilon}{r}$. The probability that there exists an $s \in \{0, 1, \cdots, r-1\}$ such that $|\frac{m}{2^{2L+1+p}} - \frac{s}{r}| \leq 2^{-(2L+1)}$ is at least $1-\epsilon$.*

Proof. By Proposition 3, for any fixed $s_0 \in \{0, 1, \cdots, r-1\}$ with $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ being an integer, we have

$$\text{Prob}(\frac{m}{2^{2L+1+p}} = \frac{s_0}{r}) \geq \frac{1}{r}. \tag{49}$$

For any fixed $s_0 \in \{0, 1, \cdots, r-1\}$ with $2^{\frac{L}{2}+1} \cdot \frac{s_0}{r}$ being not an integer, by Proposition 1 and Eq. (23), we get that the probabilty of

$$d_{t_2}(m_2, (\frac{s_0}{r})_{\{\frac{L}{2},2L+1+p\}})) < 2^p \tag{50}$$

and

$$d_{t_1}(m_1, (\frac{s_0}{r})_{\{1,t_1\}}) < 2^p \tag{51}$$

is at least $\frac{1}{r}(1-\epsilon')^2 = \frac{1}{r}(1-\frac{\epsilon}{2})^2 > \frac{1-\epsilon}{r}$. Consequently, by Proposition 4, we obtain

$$\text{Prob}(|\frac{m}{2^{2L+1+p}} - \frac{s_0}{r}| \le 2^{-(2L+1)}) > \frac{1-\epsilon}{r}. \tag{52}$$

Similar to the proof of Proposition 2, we can obtain the probability that there exists an $s \in \{0, 1, \cdots, r-1\}$ such that $|\frac{m}{2^{2L+1+p}} - \frac{s}{r}| \le 2^{-(2L+1)}$ is at least $1-\epsilon$. Finally, the theorem has been proved. $\square$

4 Complexity analysis

The complexity of the circuit of (distributed) order-finding algorithm depends on the construction of $C_t(M_a)$. There are two kinds of implementation of $C_t(M_a)$ proposed by Shor [18]. The first method (denoted as method (I)) needs time complexity $O(L^3)$ and space complexity $O(L)$, and the second method (denoted as method (II)) needs time complexity $O(L^2 \log L \log\log L)$ and space complexity $O(L \log L \log\log L)$. In this section, we compare our distributed order-finding algorithm with the traditional order-finding algorithm. For a more concrete comparison, we consider that $C_t(M_a)$ is implemented by method (I). There is a concrete implementation of order-finding algorithm by using method (I) in [10]. However, the advantages of our distributed order-finding algorithm in space and circuit depth are independent of whether method (I) or method (II) is used.

Space complexity. The implementation of the operator $C_t(M_a)$ in method (I) needs $t+L$ qubits plus b auxiliary qubits for any positive integer a, where b is $O(L)$. By Theorem 1, to teleport L qubits, computers A and B need to share L pairs of EPR pairs and communicate with $2L$ classical bits. As a result, A needs $\frac{5L}{2} + 1 + \lceil \log_2(2+\frac{1}{\epsilon}\rceil) + b$ qubits and B needs $\frac{5L}{2} + 2 + \lceil \log_2(2+\frac{1}{\epsilon})\rceil + b$ qubits. As a comparison, order-finding algorithm needs $3L + 1 + \lceil \log_2(2+\frac{1}{2\epsilon})\rceil + b$ qubits. So, our distributed order-finding algorithm can reduce nearly $\frac{L}{2}$ qubits.

Time complexity. The operator $C_t(M_a)$ can be implemented by means of $O(tL^2)$ elementary gates in method (I). Hence the gate complexity (or time complexity) in both our distributed order-finding algorithm and order-finding algorithm is $O(L^3)$.

Circuit depth. By Fig. 1, we know that the circuit depth of $C_t(M_a)$ depends on the circuit depth of controlled-$M_a^{2^x}$ $(x = 0, 1, \cdots, t-1)$ and t. The circuit depth of controlled-$M_a^{2^x}$ is $O(L^2)$ in method (I). By observing the value "t" in order-finding algorithm and our distributed order-finding algorithm, we clearly get that the circuit depth of each computer in our distributed order-finding algorithm is less than the traditional order-finding algorithm, even though both are $O(L^3)$.

Communication complexity. In our distributed Shor's algorithm, we need to teleport L qubits (in step 5 of Algorithm 3). Therefore, the communication complexity of our distributed Shor's algorithm is $O(L)$. As a comparison, the communication complexity of the distributed order-finding algorithm proposed in [10] is $O(L^2)$. In [10] they directly divide the circuit into several parts. However, the CNOT gates acting on different parts cannot be directly implemented. In order to solve this difficulty, they use some operations called cat-entangler and cat-disentangler to implement non-local CNOT gates (the implemention of each non-local CNOT gate needs to communicate 2 classical bits and previously share an EPR pair). They demonstrated that their division makes it necessary to implement $O(L^2)$ non-local CNOT gates and thus concluded that the communication complexity of their distributed Shor's algorithm is $O(L^2)$.

5 Conclusions

In this paper, we have proposed a new distributed Shor's algorithm. More specifically, we have proposed a new quantum-classical hybrid distributed order-finding algorithm, which uses quantum computing to obtain results quickly, while using classical algorithms to guarantee the accuracy of the results. In this distributed quantum algorithm, two computers work sequentially via quantum teleportation. Each of them can obtain an estimation of partial bits of $\frac{s}{r}$ for some $s \in \{0, 1, \cdots, r-1\}$ with high probability, where r is the "order". It is worth mentioning that they can also be executed in parallel to some extent. We have shown that our distributed algorithm has advantages over the traditional order-finding algorithm in space and circuit depth, which is vital in the NISQ era. Our distributed order-finding algorithm can reduce nearly $\frac{L}{2}$ qubits and reduce the circuit depth to some extent for each computer. However, unlike parallel execution, the way of serial execution that has been used in our algorithm leads to noise in both computers.

We have proved the correctness of this distributed algorithm on two computers, a natural problem is whether or not this method can be generalized to multiple computers or to other quantum algorithms. We would further consider the problem in subsequent study.

Acknowledgements

The authors are grateful to the anonymous referee for proposing very important comments and suggestions that help us improve the quality of the paper. This work is partly supported by the National Natural Science Foundation of China (Nos. 61572532, 61876195) and the Natural Science Foundation of Guangdong Province of China (No. 2017B030311011).

References

1. A. Aspuru-Guzik, A.D. Dutoi, P. J. Love and M. Head-Gordon (2005), *Simulated quantum computation of molecular energies*, Science, Vol. 309(5741), pp. 1704–1707.
2. G. Rosenberg, P. Haghnegahdar, P. Goddard, P. Carr, K. Wu and M. L. De Prado (2016), *Solving the optimal trading trajectory problem using a quantum annealer*, IEEE J. Sel. Top. Signal Process., Vol. 10(6), pp. 1053–1060.
3. P. W. Shor (1994), *Algorithms for quantum computation: discrete logarithms and factoring*, in: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134.
4. L. K. Grover (1996), *A fast quantum mechanical algorithm for database search*, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212–219.
5. A. W. Harrow, A. Hassidim and S. Lloyd (2009), *Quantum algorithm for linear systems of equations*, Phys. Rev. Lett., Vol. 103(15), pp. 150502.
6. A. Montanaro (2016), *Quantum algorithms: an overview*, npj Quantum Inf., Vol. 2, pp. 15023.
7. J. Avron, O. Casper and I. Rozen (2021), *Quantum advantage and noise reduction in distributed quantum computing*, Phys. Rev. A, Vol. 104(5), pp. 052404.
8. R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd and M. Stather (2013), *Efficient distributed quantum computing*, Proc. Math. Phys. Eng. Sci., Vol. 469(2153), pp. 20120686.
9. K. Li, D. Qiu, L. Li, S. Zheng and Z. Rong (2017), *Application of distributed semi-quantum computing model in phase estimation*, Inform. Process. Lett., Vol. 120, pp. 23–29.
10. A. Yimsiriwattana and S.J. Lomonaco (2004), *Distributed quantum computing: a distributed Shor algorithm*, Quantum Inf. Comput. II, Vol. 5436, pp. 360–372.
11. C. Gidney and M. Ekera (2021), *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*, Quantum, Vol. 5, pp. 433.

12. S. Beauregard (2003), *Circuit for Shor's algorithm using $2n+3$ qubits*, Quantum Inf. Comput., Vol. 3(2), pp. 175–185.
13. T. Haner, M. Roetteler and K. M. Svore (2017), *Factoring using $2n+2$ qubits with Toffoli based modular multiplication*, Quantum Inf. Comput., Vol. 17(7-8), pp. 673–684.
14. S. Parker and M. B. Plenio (2000), *Efficient factorization with a single pure qubit and* $\log N$ *mixed qubits*, Phys. Rev. Lett., Vol. 85(14), pp. 3049–3052.
15. M. A. Nielsen and I. L. Chuang (2000), *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge.
16. C. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres and W. K. Wootters (1993), *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*, Phys. Rev. Lett., Vol. 70(13), pp. 1895–1899.
17. N. Gisin and R. Thew (2007), *Quantum communication*, Nat. Photonics, Vol. 1(3), pp. 165–171.
18. P. W. Shor (1999), *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, Siam Rev., Vol. 41(2), pp. 303–332.

Appendix A: An example

We will give an example to show the procedure of our distributed order-finding algorithm. For convenience, we omit some details and modify some parameters.

Give a 10-bit composite number $N = 2^{10} - 1 = 1023$ and an integer $a = 2$. The order r of a modulo N is 10 (i.e. $r = 10$). In this example, the purpose of the quantum part of the order-finding algorithm is to output an estimation of s/r for some s with error no larger than 2^{-21}. In addition, Alice is to estimate the first 6 bits of s/r and Bob is to estimate the 5th to the 21th bit of s/r (17 bits for Bob). The procedure of our distributed Shor's algorithm is as follows:

(1) Alice initializes qubits as $|0\rangle_A|1\rangle_C$ and Bob initializes qubits as $|0\rangle_B$.
Registers A, B, and C have 6-qubit, 17-qubit, and 10 qubit, respectively.

(2) Alice applies "partial" order-finding algorithm.
Alice's qubits becomes $\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle_A|u_s\rangle_C$, where $\widetilde{s/r}$ is an estimation of the first 6 bits of s/r for some s.

(3) Teleport the qubits of register C to Bob.
At this time, the global quantum state is $\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle_A|0\rangle_B|u_s\rangle_C$. Also, Bob owns qubits of register B and C.

(4) Bob applies "partial" order-finding algorithm.
The global quantum state becomes

$$\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle_A|\widetilde{s/r}'\rangle_B|u_s\rangle_C,$$

where $\widetilde{s/r}'$ is an estimation of the 5th to the 21th bit of s/r.

(5) Alice measures register A and Bob measures register B.
In this step, Alice and Bob will obtain an estimation of partial bits of some s/r, respectively. Suppose $s = 7$ (we should remember that in the actual algorithm process we do not know s and r). It is worth mentioning that order-finding algorithm can be regarded as a phase estimation

algorithm. Note that $\frac{s}{r} = \frac{7}{10} = (0.1011\ 0011\ 0011\cdots)_2$ where $(0.1011\ 0011\ 0011\cdots)_2$ indicates that $0.1011\ 0011\ 0011\cdots$ is a binary representation. We can see that the Alice's measurement is most likely to be

$$1011\ 01,$$

since $0.1011\ 01$ is the nearest 6-bit binary decimal to $\frac{7}{10} = (0.1011\ 0011\ 0011\cdots)_2$. Similarly, we know that Bob's measurement is most likely to be

$$0011\ 0011\ 0011\ 0011\ 0$$

(17 bits), since the bits after the 5th bit (include the 5th bit) of $\frac{7}{10}$ is

$$0.0011\ 0011\ 0011\ 0011\ 0011\cdots.$$

Fig. A.1 shows the relationship between these estimations.

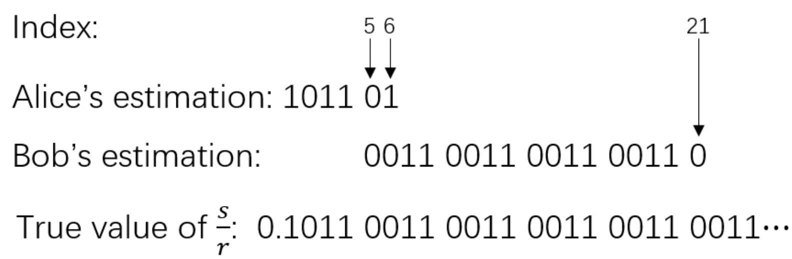

Fig. A.1. The relationship of estimations

(6) Apply steps 1 and 2 of the $CorrectResults$ subroutine to correct Alice's measurement. Alice's estimate of the 5th and the 6th bits of $\frac{7}{10}$ is 01. Bob's estimate of the 5th and the 6th bits of $\frac{7}{10}$ is 00. Since $((01)_2 - 1) \bmod 2^2 = (00)_2$, we get that the $CorrectionBit$ is -1. We use the $CorrectionBit$ to correct the result of Alice, and it becomes $((1011\ 01)_2 - 1) \bmod 2^6 = (1011\ 00)_2$. Fig. A.2 shows the relationship between these results (the overall result is obtained in the next step).

(7) By directly concatenating the first 6th bits of Alice's result and the bits after the 2th bit of Bob's result, we obtain the overall estimation is $0.1011\ 0011\ 0011\ 0011\ 0011\ 0$ (21 bits). It satisfies

$$(0.1011\ 0011\ 0011\ 0011\ 0011\ 0)_2 - \frac{7}{10} < \frac{1}{2^{21}}. \tag{A.1}$$

It means that we output an estimation with error no large than 2^{-21} and thus our algorithm works for this example.

(8) Apply continued fractions algorithm. Although this step is not considered in our paper, for completeness, we still show this step.

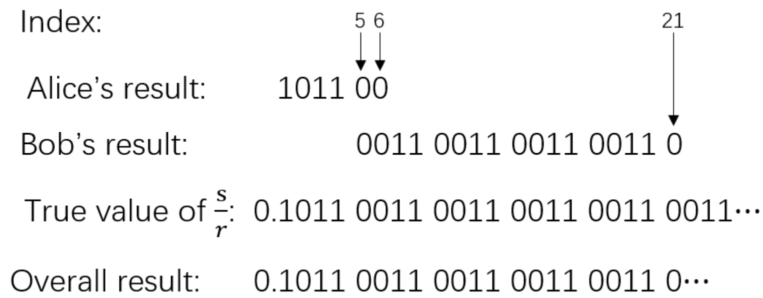

Fig. A.2. The relationship of results

Note that our estimation of $\dfrac{s}{r}$ is $(0.1011\ 0011\ 0011\ 0011\ 0011\ 0)_2 = \dfrac{734003}{1048576}$. After applying continued fractions algorithm, we have

$$\frac{734003}{1048576} = 0 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{52428 + \cfrac{1}{2}}}}}. \tag{A.2}$$

Hence we know that

$$0 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{3}}} = \frac{7}{10} \tag{A.3}$$

is the closest to $(0.1011\ 0011\ 0011\ 0011\ 0011\ 0)_2$ of all numbers with the form $\frac{p}{q}$ where $\gcd(p, q) = 1$ and $q < N = 1023$. Thus, we get that $\dfrac{s}{r} = \dfrac{7}{10}$ for some s and obtain that 10 is a factor of r. If repeating steps (1)-(8) several times, we are likely to get all prime factors of the order r, and finally conclude that $r = 10$.

We have obtained the order r by means of our distributed order-finding algorithm. Let us continue with the rest of the Shor's algorithm. Since $r = 10$ is even, we compute $\gcd(2^{r/2} + 1, N)$ and $\gcd(2^{r/2} - 1, N)$. We get that

$$\gcd(2^{r/2} + 1, N) = \gcd(33, 1023) = 33 \tag{A.4}$$

and

$$\gcd(2^{r/2} - 1, N) = \gcd(31, 1023) = 31. \tag{A.5}$$

Finally, we have $N = 1023 = 33 * 31$ and we can continue to try to factor 33 and 31 similarly if necessary.