



**Distributed quantum algorithm for Simon's problem**Jiawei Tan <sup>\*</sup>, Ligang Xiao, and Daowen Qiu <sup>†</sup>*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, 510006, China;  
and QUDOOR Technologies Inc., Guangzhou, China*Le Luo *School of Physics and Astronomy, Sun Yat-sen University, Zhuhai 519082, China  
and QUDOOR Technologies Inc., Guangzhou, China*

Paulo Mateus

*Instituto de Telecomunicações, Departamento de Matemática, Instituto Superior Técnico, Avenida Rovisco Pais, 1049-001 Lisbon, Portugal*

(Received 5 May 2022; revised 19 August 2022; accepted 31 August 2022; published 13 September 2022)

Limited by today's physical devices, quantum circuits with a long depth are usually noisy and difficult to realize in practice. The novel computing architecture of distributed quantum computing is expected to reduce the noise and depth of quantum circuits. In this paper, we study Simon's problem in distributed scenarios and design a distributed quantum algorithm to solve the problem. The algorithm proposed by us has the advantage of exponential acceleration compared with classical distributed computing and has the advantage of square acceleration compared with the best distributed quantum algorithm proposed before in query complexity. In particular, the previous distributed quantum algorithm for Simon's problem cannot be extended to the case of more than *two computing nodes* (i.e., two subproblems), but our distributed quantum algorithm can be extended to the case of *multiple computing nodes* (i.e., multiple subproblems).

DOI: [10.1103/PhysRevA.106.032417](https://doi.org/10.1103/PhysRevA.106.032417)**I. INTRODUCTION**

Quantum computing has been proved to have great potential in factorizing large numbers [1], unordered database searches [2], and chemical molecular simulations [3,4]. However, due to the limitations of today's physical devices, large-scale general quantum computers have not been realized. At present, quantum technology has entered the noisy intermediate-scale quantum (NISQ) era [5]. So it is possible that we can implement quantum algorithms on middle-scale circuits.

Distributed quantum computing is a novel computing architecture which combines quantum computing with distributed computing [6–9]. In distributed quantum computing architecture, multiple quantum computing nodes are allowed to communicate information through channels and cooperate to complete computing tasks. Compared with centralized quantum computing, the circuit size and depth can be reduced by using distributed quantum computing, which helps to reduce the noise of circuits. In the current NISQ era, the adoption of distributed quantum computing technology is likely conducive to the successful implementation of quantum algorithms.

Simon's problem is an important problem in the history of quantum computing [10]. In Simon's problem, quantum algorithms have the advantage of exponential acceleration over the best classical algorithms [11]. Simon's algorithm had a great enlightening effect on the subsequent proposal of Shor's algorithm, which can decompose large numbers and compute discrete logarithms in polynomial time [1]. Avron *et al.* proposed a distributed quantum algorithm to solve Simon's problem [12]. Their algorithm has the advantage of exponential acceleration compared with the classical algorithm, but in the worst case, it needs  $O(n^2)$  queries to solve Simon's problem (here  $n$  represents the length of the string input into the oracle in Simon's problem). Because their algorithm directly runs Simon's algorithm in multiple distributed nodes without using communication between nodes, the query complexity of their algorithm is high.

In this paper, we study Simon's problem in distributed scenarios and design a distributed quantum algorithm with query complexity  $O(n)$  to solve Simon's problem. We consider designing distributed quantum algorithms specifically for the particular structure of Simon's problem. We use quantum teleportation technology to realize quantum gates spanning multiple computing nodes. After the oracles' queries of multiple quantum computing nodes, we use the sorting network to process the query results of multiple quantum computing nodes. After performing the above operations, we find that we can obtain a quantum state containing a structure similar

<sup>\*</sup>912574652@qq.com<sup>†</sup>Corresponding author: issqdw@mail.sysu.edu.cn

to the original Simon's problem. Using this structure, we can solve Simon's problem with  $O(n-t)$  depth of the query complexity.

The algorithm proposed by us has the advantage of exponential acceleration compared with classical distributed computing and has the advantage of square acceleration compared with the best distributed quantum algorithm proposed before [12]. In particular, the previous distributed quantum algorithm cannot be extended to the case of more than two computing nodes (i.e., two subproblems), but our distributed quantum algorithm can be extended to the case of multiple computing nodes (i.e., multiple subproblems), which is a different technical method in distributed quantum computing.

The algorithm we design is closely related to the structure of Simon's problem. We have not found evidence that our algorithm can be directly extended to some quantum query algorithms, such as the Deutsch-Jozsa problem and Grover's search problem. However, we believe that the design of distributed quantum algorithms with advantages for Deutsch-Jozsa problem and the Grover's search problem also needs to explore the structure of the problem. On the other hand, the algorithm we designed may provide some inspiration for the distributed quantum algorithm of the hidden-subgroup problem obtained by the generalization of Simon's problem.

The remainder of this paper is organized as follows. In Sec. II, we recall Simon's problem and quantum teleportation. Then in Sec. III, we introduce Simon's problem in distributed scenarios by dividing it into multiple subproblems. After that, in Sec. IV, we describe the distributed quantum algorithm which we have designed. The correctness of the algorithm we have designed is proved in Sec. V. In Sec. VI, we compare the efficiency and scalability of our algorithm with the distributed classical algorithm and another distributed quantum algorithm proposed in [12]. Finally, in Sec. VII, we conclude with a summary.

## II. PRELIMINARIES

In this section, we review Simon's problem and quantum teleportation, which are useful in this paper.

### A. Simon's problem

Simon's problem is a special kind of hidden-subgroup problem [13]. We can describe Simon's problem as follows: Consider a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , where we have the promise that there is a string  $s \in \{0, 1\}^n$ , such that  $f(x) = f(y)$  if and only if  $x = y$  or  $x \oplus y = s$ . The symbol  $\oplus$  here stands for "binary bitwise exclusive or." We have an oracle that can query the value of function  $f$ . In classical computing, for any  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}^m$ , if we input  $(x, y)$  into the oracle, we will get  $(x, y \oplus f(x))$ . In quantum computing, for any  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}^m$ , if we input  $|x\rangle|y\rangle$  into the oracle, we will get  $|x\rangle|y \oplus f(x)\rangle$ . Our goal is to find the hidden string  $s$  by performing the minimum number of queries to  $f$ .

On a classical computer, the best algorithm for solving Simon's problem requires  $\Theta(\sqrt{2}^n)$  queries [11]. However, the best quantum algorithm to solve Simon's problem requires  $O(n)$  queries [10]. In the history of quantum computing, Simon's problem is the first to show that the quantum algorithm

has exponential acceleration compared with the classical probabilistic algorithm.

### B. Quantum teleportation

Quantum teleportation is an amazing discovery [14]. By sharing a classical information channel and a pair of entangled states, one can teleport an unknown quantum state to another distant location without actually transmitting physical qubits. This process protects the qubits from being destroyed during transport.

In distributed quantum computing, quantum gates that cross multiple computing nodes may be used. We can teleport a quantum state from one computing node to another [15,16] and then apply a multiqubit gate on the combined state so that teleportation can be carried out. In this way, we can implement qubit gates across multiple computing nodes.

## III. SIMON'S PROBLEM IN THE DISTRIBUTED SCENARIO

In order to better compare our algorithm with the classical distributed algorithm, we describe Simon's problem in the distributed scenario in this section.

In the distributed case, the original function  $f$  is divided into two parts, which can be accessed by two subfunctions of domain  $\{0, 1\}^{n-1}$ :  $f_e$  and  $f_o$  [ $\forall u \in \{0, 1\}^{n-1}$ ,  $f_e(u) = f(u0)$ ,  $f_o(u) = f(u1)$ ]. Consider the following scenario: Alice has an oracle  $O_{f_e}$  that can query all  $f_e(u)$  for all  $u \in \{0, 1\}^{n-1}$ , and Bob has an oracle  $O_{f_o}$  that can query all  $f_o(u)$  for all  $u \in \{0, 1\}^{n-1}$ . Here  $u0$  (or  $u1$ ) represents the connection between string  $u$  and character 0 (or 1). So Alice and Bob's oracles split the domain of function  $f$  into two parts at the last bit of the domain. Alice and Bob each know half of the information about function  $f$ , but neither knows all of the information about  $f$ . They need to find the hidden string  $s$  by querying their own oracles as few times as possible and exchanging information. The method in [12] solves this problem with  $O(n^2)$  queries.

In this paper, we propose a distributed quantum algorithm to solve this problem with  $O(n)$  queries. In particular, our method can deal with a more general case; that is, there are  $2^t$  people, each of whom has an oracle  $O_{f_w}$  ( $w \in \{0, 1\}^t$  is each person's unique identifier) that can query all  $f_w(u) = f(uw)$  for all  $u \in \{0, 1\}^{n-t}$  ( $uw$  here represents the connection between string  $u$  and string  $w$ ). So each person can access  $2^{n-t}$  values of  $f$ . Note that there is no common element between the values of function  $f$  that each person can access. They need to find the hidden string  $s$  by querying their own oracle as few times as possible and exchanging information. Actually, this problem cannot be solved by the method in [12].

Next, we further introduce some notations that will be used in this paper.

*Definition 1.* For all  $u \in \{0, 1\}^{n-t}$ , let multiset  $G(u) = \{f(uw) | w \in \{0, 1\}^t\}$ .

Notice that there could be multiple identical elements in  $G(u)$ . An example of  $G(u)$  is shown in the Appendix.

*Definition 2.* For all  $u \in \{0, 1\}^{n-t}$ , let  $S(u)$  represent a string of length  $2^t m$  by concatenating all strings  $f(uw)$  ( $w \in \{0, 1\}^t$ )

according to lexicographical order, that is,

$$S(u) = f(uw_0)f(uw_1) \cdots f(uw_{2^t-1}), \quad (1)$$

where  $f(uw_0) \leq f(uw_1) \leq \cdots \leq f(uw_{2^t-1}) \in \{0, 1\}^m$ , with  $w_i \in \{0, 1\}^t (i = 0, 1, \dots, 2^t - 1)$ , where  $w_i \neq w_j$  for any  $i \neq j$ .

An example of  $S(u)$  is given in the Appendix.

**Definition 3.** For any binary string  $u, v \in \{0, 1\}^n$ ,  $uv$  denotes  $(\sum_{i=1}^n u_i v_i) \bmod 2$ .

**Definition 4.** For any binary string  $u \in \{0, 1\}^n$ ,  $u^\perp$  denotes  $\{v \in \{0, 1\}^n | uv = 0\}$ .

Let  $s$  be the target string to be found, and denote  $s = s_1 s_2$ , where the length of  $s_1$  ( $s_2$ ) is  $n - t$  ( $t$ ). We will introduce our algorithm in the next section. Our algorithm is divided into two subalgorithms: Algorithms 2 and 3. We use Algorithm 2 to figure out  $s_1$  and Algorithm 3 to figure out  $s_2$ .

Note that the  $2^t$  oracles are not necessarily decomposed in the last  $t$  bits of  $f$ 's domain but can be any  $t$  bits in the  $n$  bits of  $f$ 's domain. However, the problem can be equivalent to the final  $t$ -bit decomposition in the domain of  $f$  by logical relabeling.

It should be noted that our algorithm also works for the original Simon's problem. The purpose of this section is to better compare our algorithm with other classical or quantum distributed algorithms and to illustrate the cases that our distributed quantum algorithm can handle that the original Simon's algorithm cannot.

The following theorem concerning  $S(u)$  is useful and important.

**Theorem 1.** Suppose function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , satisfies that there is a string  $s \in \{0, 1\}^n$  with  $s \neq 0^n$ , such that  $f(x) = f(y)$  if and only if  $x = y$  or  $x \oplus y = s$ . Then  $\forall u, v \in \{0, 1\}^{n-t}$ ,  $S(u) = S(v)$  if and only if  $u \oplus v = 0^{n-t}$  or  $u \oplus v = s_1$ , where  $s = s_1 s_2$ .

*Proof.* Based on the properties of a multiset,  $\forall u, v \in \{0, 1\}^{n-t}$ ,  $S(u) = S(v)$  if and only if  $G(u) = G(v)$ . So our goal is to prove  $\forall u, v \in \{0, 1\}^{n-t}$ ,  $G(u) = G(v)$  if and only if  $u \oplus v = 0^{n-t}$  or  $u \oplus v = s_1$ .

(1)  $\Leftarrow$ . (i) If  $u \oplus v = 0^{n-t}$ , we clearly have  $G(u) = G(v)$ . (ii)  $u \oplus v = s_1$ . Since for (i) we have  $u \oplus v = 0^{n-t}$ , we can now assume that  $s_1$  is not  $0^{n-t}$ . Then we can prove that  $\forall u \in \{0, 1\}^{n-t}$ ,  $\forall w_1 \neq w_2 \in \{0, 1\}^t$ ,  $f(uw_1) \neq f(uw_2)$ . We assume two  $t$ -bit strings  $w_1$  and  $w_2$  exist, with  $w_1 \neq w_2$  and  $f(uw_1) = f(uw_2)$ ; we have  $uw_1 \oplus uw_2 = 0^n$  or  $uw_1 \oplus uw_2 = s$ . Since  $w_1 \neq w_2$ , we have  $uw_1 \oplus uw_2 = s$ . Then we have  $s = uw_1 \oplus uw_2 = 0^t(w_1 \oplus w_2) = s_1 s_2$ . This is contrary to  $s_1 \neq 0^{n-t}$ . So there are no repeating elements in each  $G(u)$ .

Then we can prove  $G(u) \subseteq G(v)$ . We have  $\forall z \in G(u)$ ,  $\exists w \in \{0, 1\}^t$  such that  $z = f(uw)$ . According to the definition of Simon's problem,  $f(uw \oplus s) = f(uw) = z$ . Then we have  $z = f(uw \oplus s) = f((u \oplus s_1)(w \oplus s_2)) = f(v(w \oplus s_2)) \in G(v)$ . So we have  $G(u) \subseteq G(v)$ . Similarly, we can prove that  $G(v) \subseteq G(u)$ . As a result, we have  $G(u) = G(v)$ .

(2)  $\Rightarrow$ . Since  $G(u) = G(v)$ , we have  $\forall z \in G(u)$ ,  $z \in G(v)$ . Then we have  $\exists w', w' \in \{0, 1\}^t$  such that  $z = f(uw)$  and  $z = f(vw')$ . As a result, we have  $f(uw) = f(vw')$ . According to the definition of Simon's problem, we have  $uw \oplus vw' = 0^n$  or  $uw \oplus vw' = s$ . Therefore, we have  $u \oplus v = 0^{n-t}$  or  $u \oplus v = s_1$ .  $\blacksquare$

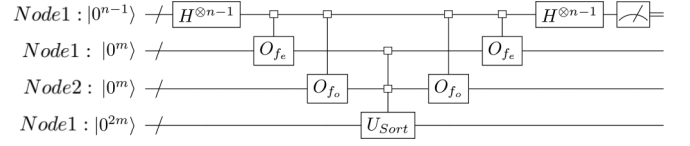


FIG. 1. The circuit for the quantum part of the distributed quantum algorithm for finding  $s_1$ : two computing nodes.

#### IV. DISTRIBUTED QUANTUM ALGORITHM FOR SIMON'S PROBLEM

We consider how to find  $s_1$ . We first give the algorithm for finding  $s_1$  with only two distributed computing nodes, i.e.,  $t = 1$ . We use an operator  $U_{\text{Sort}}$  in our algorithm. The effect of operator  $U_{\text{Sort}}$  in Fig. 1 is  $\forall a, b \in \{0, 1\}^m$  and  $c \in \{0, 1\}^{2m}$ ,

$$U_{\text{Sort}}|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus [\min(a, b) \max(a, b)]\rangle \quad (2)$$

$$= \begin{cases} |a\rangle|b\rangle|c \oplus (ab)\rangle, & a \leq b, \\ |a\rangle|b\rangle|c \oplus (ba)\rangle, & a > b. \end{cases} \quad (3)$$

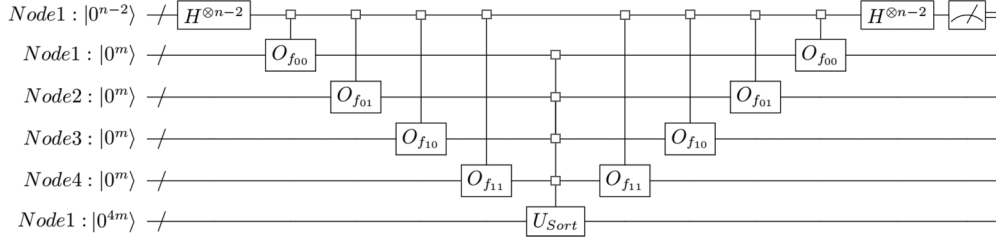
Intuitively, the effect of  $U_{\text{Sort}}$  is to sort the values in the first two registers and XOR to the third register.  $U_{\text{Sort}}$  does not change the states of the first two registers (i.e.,  $|a\rangle$  and  $|b\rangle$ ). We call these two registers the control registers of  $U_{\text{Sort}}$ . In order to show the control registers of  $U_{\text{Sort}}$  in quantum circuit diagram clearer, we use a square to mark the control registers of  $U_{\text{Sort}}$  in Fig. 1. Similarly, we use a square to mark the control registers of every oracle in Figs. 2–4.

We give a quantum circuit diagram corresponding to Algorithm 1 in Fig. 1, where we can implement controlled quantum gates spanning two computing nodes by using the quantum teleportation described in Sec. II B.

The first, second, and fourth wires in Fig. 1 are at the first computing node, and the third wire is at the second computing

**Algorithm 1.** Distributed quantum algorithm for finding  $s_1$  (two distributed computing nodes).

- 1:  $|\psi_0\rangle = |0^{n-1}\rangle|0^m\rangle|0^m\rangle|0^{2m}\rangle$ ;
- 2:  $|\psi_1\rangle = (H^{\otimes n-1} \otimes I^{\otimes 4m})|\psi_0\rangle = (H^{\otimes n-1}|0^{n-1}\rangle)|0^m\rangle|0^m\rangle|0^{2m}\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{u \in \{0, 1\}^{n-1}} |u\rangle|0^m\rangle|0^m\rangle|0^{2m}\rangle$ ;
- 3: Each computing node queries its own oracle under the control of the first quantum register:  
 $|\psi_2\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{u \in \{0, 1\}^{n-1}} |u\rangle|f_e(u)\rangle|f_o(u)\rangle|0^{2m}\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{u \in \{0, 1\}^{n-1}} |u\rangle|f(u0)\rangle|f(u1)\rangle|0^{2m}\rangle$ ;
- 4: The fourth quantum register performs its own  $U_{\text{Sort}}$  under the control of the second and third quantum registers:  
 $|\psi_3\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{u \in \{0, 1\}^{n-1}} |u\rangle|f(u0)\rangle|f(u1)\rangle\{[\min[f(u0), f(u1)] \max[f(u0), f(u1)]]\}$ ;
- 5: Each computing node queries its own oracle under the control of the first quantum register:  
 $|\psi_4\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{u \in \{0, 1\}^{n-1}} |u\rangle|0^m\rangle|0^m\rangle\{[\min[f(u0), f(u1)] \max[f(u0), f(u1)]]\}$ ;
- 6:  $|\psi_5\rangle = (H^{\otimes n-1} \otimes I^{\otimes 4m})|\psi_4\rangle$ ;
- 7: Measure the first quantum register, and get an element in  $s_1^\perp$ .

FIG. 2. The circuit for the quantum part of the distributed quantum algorithm for finding  $s_1$ : four computing nodes.

node. Using the quantum teleportation described in Sec. II B, we can implement controlled quantum gates spanning two computing nodes.

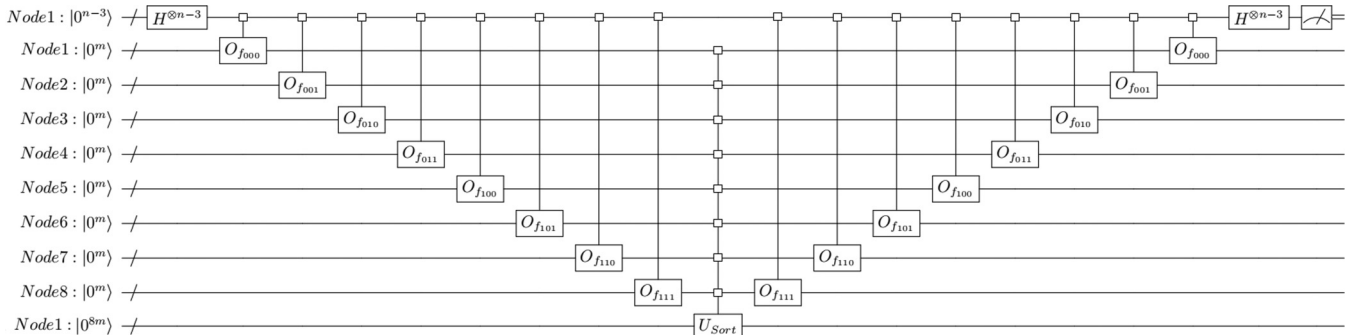
Our algorithm can be further extended to the case of  $2^t$  computing nodes. The second and third registers in Algorithm 1 will be replaced by  $2^t$  registers. The  $2^t$  registers in the middle will correspond to  $2^t$  distributed computing nodes. The extended algorithm is shown in Algorithm 2.

Similarly, each computing node uses its oracle twice. For  $t = 2$  ( $t = 3$ ), the circuit diagram of the extended algorithm is shown in Fig. 2 (Fig. 3). More generally, when  $2^t$  computing nodes are used, the quantum circuit diagram of the extended algorithm is as in Fig. 4.

The effect of  $U_{\text{Sort}}$  in Algorithm 2 is to sort the values in the  $2^t$  control registers by lexicographical order and XOR to the target register. It is not difficult to implement for sorting multiple elements. By virtue of using the sorting network in [17,18],  $2^t$  elements can be sorted in  $O(t)$  depth of comparators. Here the comparator is a basic circuit module which is easy to realize.

Note that the oracle query of each quantum computing node in Algorithms 1 and 2 can actually be completed in parallel. With the help of auxiliary  $(2^t - 1)(n - t)$  qubits, we can change the state of the control register after the first Hadamard transformation  $\frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle$  to  $\frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle |u\rangle \cdots |u\rangle$ . That is, we changed the control register from one group to the same  $2^t$  groups. In fact, after the first Hadamard transformation, we can teleport each group of  $n - t$  control bits to every quantum computing node and use this to control the oracle of the computing node.

After finding  $s_1$ , we use Algorithm 3 to find  $s_2$ . Algorithm 3 needs to query each oracle at most twice. Finally, we can obtain  $s = s_1 s_2$ .

FIG. 3. The circuit for the quantum part of the distributed quantum algorithm for finding  $s_1$ : eight computing nodes.

## V. CORRECTNESS ANALYSIS OF ALGORITHMS

In this section, we prove the correctness of our algorithm. First, we write out the state after the first step of the algorithm in Fig. 4:

$$|\phi_1\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle \left( \bigotimes_{w \in \{0,1\}^t} |0^m\rangle \right) |0^{2^t m}\rangle \quad (4)$$

$$= \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle \underbrace{|0^m\rangle \cdots |0^m\rangle}_{2^t} |0^{2^t m}\rangle. \quad (5)$$

The second step of the algorithm queries the oracle  $O_{f_{0^t}}$ , resulting in the following state:

$$|\phi_2\rangle = \left( O_{f_{0^t}} \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle |0^m\rangle \right) \underbrace{|0^m\rangle \cdots |0^m\rangle}_{2^t-1} |0^{2^t m}\rangle \quad (6)$$

$$= \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle |f_{0^t}(u)\rangle \underbrace{|0^m\rangle \cdots |0^m\rangle}_{2^t-1} |0^{2^t m}\rangle \quad (7)$$

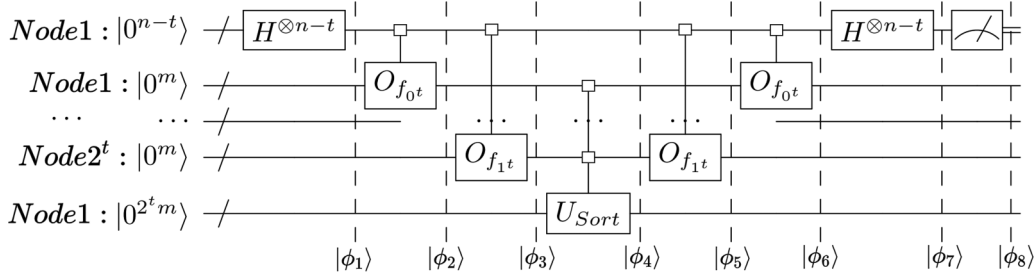
$$= \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle |f(u0^t)\rangle \underbrace{|0^m\rangle \cdots |0^m\rangle}_{2^t-1} |0^{2^t m}\rangle. \quad (8)$$

The algorithm then queries each of the other oracles based on the circuit diagram in Fig. 4 to get the following states:

$$|\phi_3\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle \underbrace{|f(u0^t)\rangle \cdots |f(u1^t)\rangle}_{2^t} |0^{2^t m}\rangle. \quad (9)$$

After sorting by using  $U_{\text{Sort}}$ , we have the following state:

$$|\phi_4\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle \underbrace{|f(u0^t)\rangle \cdots |f(u1^t)\rangle}_{2^t} |S(u)\rangle. \quad (10)$$


 FIG. 4. The circuit for the quantum part of the distributed quantum algorithm for finding  $s_1$ :  $2^t$  computing nodes.

After that, we query each oracle again and restore the status of the  $2^t$   $m$ -bit registers to  $|0^m\rangle$ . Then we obtain the following state:

$$|\phi_6\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle \underbrace{|0^m\rangle \cdots |0^m\rangle}_{2^t} |S(u)\rangle. \quad (11)$$

By tracing out the states of the  $2^t$   $m$ -bit registers in the middle, we get the state

$$|\phi'_6\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle |S(u)\rangle. \quad (12)$$

From Theorem 1, we know that the structure of the original Simon's problem exists in function  $S$ .

After Hadamard transformation on the first register, we get the following state:

$$\begin{aligned} |\phi'_7\rangle &= (H^{\otimes n-t} \otimes I^{\otimes 2^t m}) |\phi'_6\rangle \\ &= \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u\rangle) |S(u)\rangle \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \left( \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u\rangle) |S(u)\rangle \right. \end{aligned}$$

**Algorithm 2.** Distributed quantum algorithm for finding  $s_1$  ( $2^t$  distributed computing nodes).

- 
- 1:  $|\phi_0\rangle = |0^{n-t}\rangle (\otimes_{w \in \{0,1\}^t} |0^m\rangle) |0^{2^t m}\rangle$ ;
  - 2:  $|\phi_1\rangle = (H^{\otimes n-t} \otimes I^{\otimes 2^t m}) |\phi_0\rangle = (H^{\otimes n-t} |0^{n-t}\rangle) (\otimes_{w \in \{0,1\}^t} |0^m\rangle) |0^{2^t m}\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle (\otimes_{w \in \{0,1\}^t} |0^m\rangle) |0^{2^t m}\rangle$ ;
  - 3: Each computing node queries its own oracle under the control of the first quantum register:  $|\phi_3\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle (\otimes_{w \in \{0,1\}^t} |f_w(u)\rangle) |0^{2^t m}\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle (\otimes_{w \in \{0,1\}^t} |f(uw)\rangle) |0^{2^t m}\rangle$ ;
  - 4: The fourth quantum register performs its own  $U_{Sort}$  under the control of the middle  $2^t$  quantum registers:  $|\phi_4\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle (\otimes_{w \in \{0,1\}^t} |f(uw)\rangle) |S(u)\rangle$ ;
  - 5: Each computing node queries its own oracle under the control of the first quantum register:  $|\phi_6\rangle = \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} |u\rangle (\otimes_{w \in \{0,1\}^t} |0^m\rangle) |S(u)\rangle$ ;
  - 6:  $|\phi_7\rangle = (H^{\otimes n-t} \otimes I^{\otimes 2^t m}) |\phi_6\rangle$ ;
  - 7: Measure the first quantum register and get an element in  $s_1^\perp$ .
- 

$$\begin{aligned} &+ \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u\rangle) |S(u)\rangle) \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \left( \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u\rangle) |S(u)\rangle \right. \\ &\quad \left. + \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u \oplus s_1\rangle) |S(u \oplus s_1)\rangle) \right) \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \left( \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u\rangle) |S(u)\rangle \right. \\ &\quad \left. + \sum_{u \in \{0,1\}^{n-t}} (H^{\otimes n-t} |u \oplus s_1\rangle) |S(u)\rangle) \right) \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \sum_{u \in \{0,1\}^{n-t}} [H^{\otimes n-t} (|u\rangle + |u \oplus s_1\rangle)] |S(u)\rangle \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \sum_{u \in \{0,1\}^{n-t}} \left( \frac{1}{\sqrt{2^{n-t}}} \sum_{z \in \{0,1\}^{n-t}} \right. \\ &\quad \left. \times [(-1)^{uz} + (-1)^{(u \oplus s_1)z}] |z\rangle \right) |S(u)\rangle \\ &= \frac{1}{\sqrt{2^{n-t+2}}} \sum_{u \in \{0,1\}^{n-t}} \left( \frac{1}{\sqrt{2^{n-t}}} \sum_{z \in \{0,1\}^{n-t}} \right. \\ &\quad \left. \times (-1)^{uz} [1 + (-1)^{s_1 z}] |z\rangle \right) |S(u)\rangle. \end{aligned}$$

Note that if  $s_1 z = 1$ , we have  $1 + (-1)^{s_1 z} = 0$ , and the basis state  $|z\rangle$  vanishes in the above state. If  $s_1 z = 0$ , we have  $1 + (-1)^{s_1 z} = 2$ , so we have

$$\begin{aligned} |\phi'_7\rangle &= \frac{1}{\sqrt{2^{n-t+2}}} \sum_{u \in \{0,1\}^{n-t}} \left( \frac{1}{\sqrt{2^{n-t}}} \sum_{z \in \{0,1\}^{n-t}} \right. \\ &\quad \left. \times (-1)^{uz} [1 + (-1)^{s_1 z}] |z\rangle \right) |S(u)\rangle \\ &= \frac{1}{\sqrt{2^{n-t}}} \sum_{u \in \{0,1\}^{n-t}} \left( \frac{1}{\sqrt{2^{n-t}}} \sum_{z \in s_1^\perp} (-1)^{uz} |z\rangle \right) |S(u)\rangle \end{aligned}$$

**Algorithm 3.** Distributed quantum algorithm for finding  $s_2$  ( $2^t$  distributed computing nodes).

- 
- 
- 1: Query each oracle  $O_{f_w}$  once in parallel to get  $f(0^{n-t}w)$  ( $w \in \{0, 1\}^t$ );
  - 2: Query oracle  $O_{f_v}$  once to get  $f(s_1 0^t)$ ;
  - 3: Find a  $v \in \{0, 1\}^t$  such that  $f(0^{n-t}v) = f(s_1 0^t)$ ;
  - 4: Obtain  $s_2 = v$ .
- 
- 

$$\begin{aligned}
&= \frac{1}{2^{n-t}} \sum_{u \in \{0,1\}^{n-t}} \sum_{z \in s_1^\perp} (-1)^{uz} |z\rangle |S(u)\rangle \\
&= \frac{1}{2^{n-t}} \sum_{z \in s_1^\perp} |z\rangle \sum_{u \in \{0,1\}^{n-t}} (-1)^{uz} |S(u)\rangle.
\end{aligned}$$

After measurement on the first register, we can get a string that is in  $s_1^\perp$ . After  $O(n-t)$  repetitions of the above algorithm, we can obtain  $O(n-t)$  elements in  $s_1^\perp$ . Then, using the classical Gaussian elimination method, we can obtain  $s_1$ .

If we have already found  $s_1$ , we can use Algorithm 3 to find  $s_2$ . Since  $f(s_1 0^t) = f((s_1 0^t) \oplus s)$ , we have  $f(s_1 0^t) = f(0^{n-t} s_2)$ . So we can find a  $v$  such that  $f(s_1 0^t) = f(0^{n-t} v)$ . Then we can obtain  $s_2 = v$ . Finally, we can obtain  $s = s_1 s_2$ .

## VI. COMPARISONS WITH OTHER ALGORITHMS

First, we compare our results with other distributed algorithms. Based on the previous analysis, our distributed quantum algorithm needs  $O(n-t)$  queries for each oracle  $O_{f_u}$  to solve Simon's problem. However, in order to find  $s_1$ , distributed classical algorithms need to query oracles  $O(\sqrt{2^{n-t}})$  times. Our algorithm has the advantage of exponential acceleration compared with the classical distributed algorithm.

For  $t=1$ , the algorithm in paper [12] requires  $O(n^2)$  queries to solve Simon's problem. However, our algorithm needs only  $O(n)$  queries and has the advantage of square acceleration. In addition, the method in [12] cannot deal with the case of  $t > 1$ . Therefore, our distributed quantum algorithm has higher scalability.

Simon's problem is based on the quantum query model. In the quantum query model, we are mainly concerned with reducing the number of oracle queries. In the distributed scenario, we want to minimize the depth of oracle queries. The original Simon's algorithm [10] needs  $O(n)$  queries to solve Simon's problem, and our algorithm needs  $O(n-t)$  queries for each oracle. Our algorithm can reduce the depth of oracle queries compared to the original algorithm.

Although the number of qubits used may be increased in our algorithm, the query depth of our algorithm is decreased. This is advantageous when the implementation of the oracle is the main overhead. On the other hand, after the original function is decomposed into subfunctions, the depth of the implemented oracle of the subfunctions may be smaller than the original function because the subfunctions are simpler than the original function. So the circuit depth may be further reduced. In addition, in Simon's problem for the distributed scenario we described in Sec. III, the original Simon's algo-

rithm will no longer be applicable, but our algorithm is still applicable.

## VII. CONCLUSION

In this paper, we have designed a distributed quantum algorithm to solve Simon's problem. With multiple quantum computing nodes processing in parallel, each node needs to query its own oracle fewer times. This reduces the depth of the query complexity for each node. This helps to reduce circuit noise and makes it easier to implement with exponential acceleration advantages in the current NISQ era.

Our distributed quantum algorithm has the advantage of exponential acceleration compared with the classical distributed algorithm. Compared with previous distributed quantum algorithms, our algorithm has the advantage of square acceleration and has higher scalability.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for important suggestions that helped us improve the quality of the paper. This work is supported in part by the National Natural Science Foundation of China (Grants No. 61876195 and No. 61572532) and the Natural Science Foundation of Guangdong Province of China (Grant No. 2017B030311011).

## APPENDIX: AN EXAMPLE OF $G(u)$ AND $S(u)$

We now give a specific function  $f$  in Simon's problem and discuss  $G(u)$  and  $S(u)$  for  $t=2$  (see Tables I–III).

Consider  $s = 1001$  ( $s_1 = 10$  and  $s_2 = 01$ ) and the function  $f : \{0, 1\}^4 \rightarrow \{0, 1\}^6$  as follows: We can see that  $\forall x, y \in \{0, 1\}^4$ ,  $f(x) = f(y)$  if and only if  $x \oplus y = 0000$  or  $x \oplus y = 1001$  (i.e.,  $x \oplus y = s$ ).

Under the above conditions, we have

$$G(00) = \{f(0000), f(0001), f(0010), f(0011)\} \quad (\text{A1})$$

$$= \{100101, 101100, 000100, 110101\}, \quad (\text{A2})$$

$$G(01) = \{f(0100), f(0101), f(0110), f(0111)\} \quad (\text{A3})$$

$$= \{101010, 011001, 001101, 111100\}, \quad (\text{A4})$$

$$G(10) = \{f(1000), f(1001), f(1010), f(1011)\} \quad (\text{A5})$$

$$= \{101100, 100101, 110101, 000100\}, \quad (\text{A6})$$

$$G(11) = \{f(1100), f(1101), f(1110), f(1111)\} \quad (\text{A7})$$

$$= \{011001, 101010, 111100, 001101\}. \quad (\text{A8})$$

TABLE I. An example of function  $f$ .

$x$	$f(x)$	$x$	$f(x)$	$x$	$f(x)$	$x$	$f(x)$
0000	100101	1000	101100	0100	101010	1100	011001
0001	101100	1001	100101	0101	011001	1101	101010
0010	000100	1010	110101	0110	001101	1110	111100
0011	110101	1011	000100	0111	111100	1111	001101

TABLE III. An example of function  $S$ .

$u$	$S(u)$
00	000100100101101100110101
01	001101011001101010111100
10	000100100101101100110101
11	001101011001101010111100

So the function  $G$  is as follows: We can see that  $\forall u, v \in \{0, 1\}^2$ ,  $G(u) = G(v)$  if and only if  $u \oplus v = 00$  or  $u \oplus v = 10$  (i.e.,  $u \oplus v = s_1$ ).

For each  $G(u)$  ( $u \in \{0, 1\}^2$ ), we can obtain  $S(u)$  by concatenating all strings in  $G(u)$  according to lexicographical order.

The function  $S$  is as follows: We can see that  $\forall u, v \in \{0, 1\}^2$ ,  $S(u) = S(v)$  if and only if  $u \oplus v = 00$  or  $u \oplus v = 10$  (i.e.,  $u \oplus v = s_1$ ).

TABLE II. An example of function  $G$ .

$u$	$G(u)$
00	{100101,101100,000100,110101}
01	{101010,011001,001101,111100}
10	{101100,100101,110101,000100}
11	{011001,101010,111100,001101}

[1] P. W. Shor, *SIAM J. Comput.* **26**, 1484 (1997).  
 [2] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (ACM Press, Philadelphia, 1996), pp. 212–219.  
 [3] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, *Science* **309**, 1704 (2005).  
 [4] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, *Chem. Rev.* **119**, 10856 (2019).  
 [5] J. Preskill, *Quantum* **2**, 79 (2018).  
 [6] H. Buhrman and H. Röhrig, in *Mathematical Foundations of Computer Science 2003*, edited by G. Goos, J. Hartmanis, J. van Leeuwen, B. Rován, and P. Vojtáš, Lecture Notes in Computer Science Vol. 2747 (Springer, Berlin, 2003), pp. 1–20.  
 [7] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, *Proc. R. Soc. A* **469**, 20120686 (2013).  
 [8] K. Li, D. W. Qiu, L. Li, S. Zheng, and Z. Rong, *Inf. Process. Lett.* **120**, 23 (2017).  
 [9] D. W. Qiu, L. Luo, [arXiv:2204.10487](https://arxiv.org/abs/2204.10487).  
 [10] D. R. Simon, *SIAM J. Comput.* **26**, 1474 (1997).  
 [11] G. Cai and D. W. Qiu, *J. Comput. Syst. Sci.* **97**, 83 (2018).  
 [12] J. Avron, O. Casper, and I. Rozen, *Phys. Rev. A* **104**, 052404 (2021).  
 [13] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, Oxford, 2007).  
 [14] C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. K. Wootters, *Phys. Rev. Lett.* **70**, 1895 (1993).  
 [15] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication* (Association for Computing Machinery, New York, 2018), p. 1.  
 [16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, 2010).  
 [17] M. S. Paterson, *Algorithmica* **5**, 75 (1990).  
 [18] M. Ajtai, J. Komlos, and E. Szemerédi, in *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 1983), pp. 1–9.