CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2014, Article 10, pages 1–31
http://cjtcs.cs.uchicago.edu/

Simpler Exact Leader Election via Quantum Reduction*

Hirotada Kobayashi Keiji Matsumoto Seiichiro Tani

Received February 2, 2014; Revised September 26, 2014, and in final form December 9, 2014; Published December 16, 2014

Abstract: The anonymous network model was introduced by Angluin in [STOC '80, pages 82-93] to understand the fundamental properties of distributed computing by examining how much each party in a network needs to know about its own identity and the identities of other parties. In this model, all parties with the same number of communication links are identical. When studying the model, the problem of electing a unique leader is the central problem in the sense that once a unique leader is elected, he can assign a unique identity to each party. It was proved in the literature, however, that it is impossible to deterministically solve the leader election problem with any finite amount of communication. This contrasts sharply with the situation in the quantum setting: the present authors gave quantum algorithms that exactly solve the problem on anonymous *quantum* networks, where quantum communication and computation can be performed [TOCT, vol.4, no.1, article 1]. The core of their algorithm consists of somewhat tricky unitary operators, which eliminate the possibility of ties during (quantum) coin flipping and thereby reduce the number of leader candidates with certainty. This paper presents a new quantum leader election algorithm that is based on quantum reduction via exact amplitude amplification to a classically solvable problem, computing a certain symmetric function, which provides more intuitive reasoning behind the existence of exact quantum algorithms for leader election. The algorithm first achieves a round complexity that is linear in the number of parties, i.e., the largest possible diameter plus one of the underlying graph of the network.

Key words and phrases: quantum algorithm, distributed computing, leader election

*A short preliminary abstract of this work was published in the *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC '09)*, pages 334–335. Part of this work was undertaken when the authors were members of Quantum Computation and Information Project, funded by the ERATO program of Japan Science and Technology Agency.

DOI: 10.4086/cjtcs.2014.010

1 Introduction

1.1 Overview

Distributed computing algorithms mostly assume that every party has its own identity so that messages sent by different parties can be distinguished. Angluin [5] introduced a network model without this assumption (i.e., the *anonymous* network model, where all parties with the same number of communication links are identical[1]) to understand the fundamental properties of distributed computing by examining how much each party in a network needs to know about its own identity and the identities of other parties. Computing on anonymous networks has been further investigated in the literature to reveal that anonymous networks make it significantly hard or even impossible to exactly solve many distributed problems that are easy to solve on *non-anonymous* networks (i.e., the networks in which every party has its own identity), where by "exactly solve" we mean "solve without error within a bounded time". This implies that one of the most fundamental facts for distributed computing is that every party has its own identity. In this context, the leader election problem, namely the problem of electing a unique leader, is the most essential problem for computing on anonymous networks, since once a unique leader is elected, the leader can assign a unique identity to every party with only additional $O(n)$ rounds and $\tilde{O}(m)$ communication-bits, where n is the number of parties and m is the total number of communication links. In her seminal paper [5], Angluin initiated the study of the leader election problem on anonymous networks, and proved that there exist an infinite number of graphs on which the problem cannot exactly be solved even with an unbounded (but finite) amount of communication. Subsequently, Yamashita and Kameda [36, 37], Boldi et al. [8], and Boldi and Vigna [9] provided the necessary and sufficient conditions on the graphs for which the problem can exactly be solved. Because of the impossibility for an infinite number of graphs, probabilistic algorithms were also well investigated for the leader election [23, 24]. Other studies focused on easier problems than leader election such as computing Boolean functions [6, 28, 27], solitude verification [2, 22], and spanning tree construction [38]. Yet other studies consider variations of anonymous network models, motivated by specific applications such as wireless communication [18] and mobile agents [16].

This paper examines the quantum version of anonymous networks, which we call anonymous *quantum* networks, where quantum computation and quantum communication can be performed. A natural question would then be whether it is possible to exactly solve the leader election problem on anonymous quantum networks. The present authors gave the following affirmative answer to this question in Refs. [34, 35]: The leader election problem can exactly be solved in $O(n^2)$ rounds with $O(n^4)$ bits of communication for the number n of parties, if the use of quantum communication and computation is allowed. The idea in Refs. [34, 35] is to repeatedly reduce leader candidates by breaking the ties between the candidates whenever they occur during (quantum) coin flipping. The core of their algorithm is the tie breaking procedure, which uses somewhat tricky unitary operators that completely cancel the amplitudes of the states corresponding to the ties. This paper presents a new quantum leader election algorithm that is based on quantum reduction via exact amplitude amplification to a certain classically easy problem. The reduction thus provides more intuitive reasoning behind the existence of exact quantum algorithms. The core of the new algorithm is a subroutine that runs in superposition a simple classical algorithm for the classically easy problem. All the communications in the algorithm arise from this subroutine, and thus,

[1] We do not consider the input to each party as its own identity, since the model must be fixed before the input is given. The input to each party may be used (after it is given) to identify the party in lucky cases, but we are interested in the worst case.

the communication costs of the algorithm are bounded by functions of the communication costs of the subroutine.

The first part shows the following: Suppose that parties have a quantum algorithm that exactly verifies that there is a unique leader in an anonymous quantum network, when every party is given as input a bit indicating whether it is a "leader" or a "follower", and an upper bound N of the number of parties; then the parties can exactly elect a unique leader from among them by running $O(N)$ instances of the algorithm and their inversions in a *black box* manner together with a small amount of additional communication (for computing the *agreement verification problem*, a classically easy problem defined in the next paragraph) on the network, when every party is given the upper bound N. The problem of verifying that a unique leader exists is called the *solitude verification problem* (SV) and this has also been studied intensively [1, 2]. We should note that SV is known to be *properly easier* than the leader election problem on anonymous *classical* networks, although it is still non-trivial to solve: For instance, SV can exactly be solved on an anonymous classical ring if the number of parties is given, but a unique leader cannot exactly be elected on the ring [37]. Our result is hence unique to the quantum setting, that is, the reduction is essentially quantum.

The second part of this paper, which constitutes its main technical contribution, shows that SV is in turn reducible to a very basic problem: the problem of verifying that all parties have the value 0, called the *agreement verification problem* (AV)[2]. More quantitatively, SV can exactly be solved by running, in a black box manner, $O(N)$ instances of any quantum algorithm (and their inversions) that exactly solves AV (together with local quantum computation), if an upper bound N of the number of parties is given. This reduction is again essentially quantum (i.e., it is unique to the quantum setting), since no exact (or even no zero-error) classical algorithms can solve SV for a given N [5, 24].

These together imply that the leader election problem on any anonymous quantum network can exactly be solved essentially with the quantum communication required to run in a black box manner, $O(N^2)$ instances of any quantum algorithm (and their inversions) that exactly solves AV. This also means that any anonymous quantum network can be made *non-anonymous* with the above communication cost (up to the small amount of classical communication required for a leader to assign an identity to each party). Furthermore, with the promise that the number n of parties is given instead of N, the required quantum communication can be reduced to just $O(n)$ times the amount of quantum bits (a.k.a., qubits) communicated for exactly solving AV. We should emphasize that all the quantum communications are used for solving AV. This is somewhat surprising, since AV can exactly be solved even on anonymous *classical* networks but a unique leader cannot exactly be elected on the networks (even if every party has an unbounded computing ability). The point is that AV can be solved for any *superposition* of classical inputs on anonymous quantum networks: simply run in superposition any classical algorithm for solving AV on each of the superposed inputs. This may appear to be only a tiny advantage. Nevertheless, it indeed removes all the hardness that arises when solving the leader election problem on anonymous classical networks. This is because, on anonymous quantum networks, solving AV for any superposed input makes it possible to implement the reflection operators used in the exact amplitude amplification in a distributed

[2] It is easy to see that, to exactly solve AV, every party has only to repeat at most Δ times exchanging the set of values which it currently knows are in the anonymous network, where Δ is any upper bound of the diameter (of the underlying graph) of the network, and for instance, Δ can be set to N, an upper bound of the number of parties. We will present this in a slightly more formal form in Section 5.1.

manner. Consequently, it is *not* essential for every party to have its own identity on quantum networks, and such a difference between anonymous quantum and classical networks comes from the ability to *solve* AV *in a superposition* (and quantum local computation). This provides a simple interpretation of the fact that the leader election problem can exactly be solved on any anonymous quantum network for a given upper bound of the number of parties [34, 35], while no classical algorithm can exactly solve the problem for a certain broad class of network topologies [5, 37]. On the other hand, the algorithms given in Refs. [34, 35] do not involve the notion of quantum reduction, and hence a simple interpretation such as that described above does not emerge directly from their algorithms. In the following, we will give more precise descriptions of our main results in order, and also show that their applications improve the previously known complexities of some distributed computing problems. We mean the number of rounds and (quantum) bits communicated by an algorithm, by the round complexity and the bit complexity, respectively, of the algorithm.

1.2 Main Results

We assume that the underlying graphs of networks are *undirected* and that no faults exist on the networks. Let n be the number of parties and $H_k\colon \{0,1\}^n \to \{\mathsf{true},\mathsf{false}\}$ be the symmetric Boolean function over n distributed bits, which is true if and only if the Hamming weight of the n bits (i.e., the sum of the n bits) is k. Note that, in a non-faulty network, SV and AV described in the previous subsection are equivalent to computing H_1 and H_0, respectively. Hence we may write H_1 and H_0 instead of SV and AV for a consistent description. Let $\mathcal{H}_k$ be any quantum algorithm that exactly computes H_k without intermediate measurements on an anonymous quantum network whose underlying graph is an unknown undirected graph, if an upper bound N of the number n of parties is given to each party. Further, let $Q^{\mathrm{rnd}}(\mathcal{H}_k)$ and $Q^{\mathrm{bit}}(\mathcal{H}_k)$ be the numbers of rounds taken and qubits communicated, respectively, by $\mathcal{H}_k$ in the the worst-case over all possible superpositions of inputs (i.e., all quantum states in $\mathbb{C}^{2^n}$). Note that $Q^{\mathrm{rnd}}(\mathcal{H}_k)$ and $Q^{\mathrm{bit}}(\mathcal{H}_k)$ may vary according to the information (e.g., the number of parties) given to each party before starting computation, but the information will be clear in the context. First, the following is proved.

Theorem 1.1. *If an upper bound N of the number n of parties is given to each party, then the leader election problem can exactly be solved by using $\mathcal{H}_0$ and $\mathcal{H}_1$ (and their inversions) in a black box manner together with local quantum operations, in $O(Q^{\mathrm{rnd}}(\mathcal{H}_0)+Q^{\mathrm{rnd}}(\mathcal{H}_1))$ rounds and with bit complexity $O(N(Q^{\mathrm{bit}}(\mathcal{H}_0)+Q^{\mathrm{bit}}(\mathcal{H}_1)))$, on any anonymous quantum network whose underlying graph is an unknown undirected graph. Furthermore, with the promise that the number n of parties is given instead of N, the bit complexity can be reduced to $O(Q^{\mathrm{bit}}(\mathcal{H}_0)+Q^{\mathrm{bit}}(\mathcal{H}_1))$.*

This does not have a classical counterpart, since symmetric Boolean functions (including H_k) can exactly be computed for any anonymous classical network if n is given [36, 37, 28], but a unique leader can exactly be elected on an anonymous classical network if and only if its underlying graph satisfies a certain symmetry condition [37].

Next, it is proved that computing H_1 is quantumly reducible to computing H_0. This is the main technical contribution of this paper.

Theorem 1.2. *If an upper bound N of the number n of parties is given to each party, then the function H_1 can exactly be computed without intermediate measurements by using $\mathcal{H}_0$ (and its inversion) in a black box manner, in $O(Q^{\mathrm{rnd}}(\mathcal{H}_0))$ rounds and with bit complexity $O(N \cdot Q^{\mathrm{bit}}(\mathcal{H}_0))$, for any possible superposition of n-bit inputs on any anonymous quantum network whose underlying graph is an unknown undirected graph.*

This reduction cannot be dequantized either, for it is impossible to exactly compute H_1 on certain anonymous classical networks for given N [5, 24] while H_0 can be computed. Theorems 1.1 and 1.2 imply that the complexities of the leader election problem are characterized by those of computing H_0. Namely, we have the following corollary.

Corollary 1.3. *If an upper bound N of the number n of parties is given to each party, then the leader election problem can exactly be solved by using $\mathcal{H}_0$ (and its inversion) in a black box manner together with local quantum operations, in $O(Q^{\mathrm{rnd}}(\mathcal{H}_0))$ rounds and with bit complexity $O(N^2 \cdot Q^{\mathrm{bit}}(\mathcal{H}_0))$, on any anonymous quantum network whose underlying graph is an unknown undirected graph. Furthermore, with the promise that the number n of parties is given instead of N, the bit complexity can be reduced to $O(n \cdot Q^{\mathrm{bit}}(\mathcal{H}_0))$.*

Since a unique leader can assign an identity to each party with the communication cost of only $O(n)$ rounds and $O(m)$ messages of $O(\log n)$ bits each, where m is the number of all communication links, $O(Q^{\mathrm{rnd}}(\mathcal{H}_0))$ and $O(n \cdot Q^{\mathrm{bit}}(\mathcal{H}_0))$ would normally be dominant terms of the complexities of assigning identities on an anonymous quantum network.

Our quantum reductions are based on ideas that differ greatly from those used in Refs. [34, 35]. The known tool we use is the amplitude amplification technique [11], which is a very basic technique that has been used in many quantum computing studies. We use it in a non-trivial and distributed way that satisfies the requirements on anonymous networks.

The proofs of Theorems 1.1 and 1.2 can be carried over to asynchronous networks in a straightforward manner: If we assume that the communication links have the FIFO property, every party emulates a synchronous network simply by delaying the local operation that would be performed in each round in the synchronous network until a message arrives at every port, since, in our algorithms, every party receives and sends a message via *every* port in each round (even when the communication links do not have the FIFO property, our algorithms still work with the bit complexity multiplied by an additional log factor, by appending to each message a label indicating in which round it would be sent in the synchronous setting).

1.3 Technical Outlines

Theorem 1.1: We first assume that every party knows the exact number n of parties. Suppose that every party is initially a leader candidate. Every party then flips a coin that lands on heads with probability $1/n$ and tails with $1-1/n$. If exactly one party sees heads, that party becomes a unique leader. The probability of this successful case is lower-bounded by some constant. We quantize this randomized algorithm and amplify the success probability to one by applying the exact quantum amplitude amplification [13, 11]. To implement the reflection operator against the success states, the parties invoke $\mathcal{H}_1$; to implement the reflection operator against the all-zero state (i.e., a component of the diffusion operator), the parties invoke $\mathcal{H}_0$. This quantumly reduces the leader election problem to computing H_0 and H_1 in the sense

that all communication is performed to compute H_0 and H_1 for superposed inputs. To generalize this to the case where only an upper bound N is given, the parties run the above algorithm in parallel with $t \in \{2,\ldots,N\}$ instead of n, and then agree by using $\mathcal{H}_1$ again on some t for which a unique leader is elected. Each run of the algorithm for $t \in \{2,\ldots,N\}$ invokes $\mathcal{H}_0$ and $\mathcal{H}_1$ constant times. The round complexity of the entire algorithm is therefore of the same order as the maximum of those of $\mathcal{H}_0$ and $\mathcal{H}_1$, and its total bit complexity is at most $O(N)$ times the sum of their bit complexities.

Theorem 1.2: Suppose that every party i is given a Boolean variable x_i. Let $\vec{x} = (x_1,\ldots,x_n)$ and $|\vec{x}| = \sum_{i=1}^{n} x_i$. The goal is to exactly compute $H_1(\vec{x})$. First consider the following algorithm that probabilistically computes $H_1(\vec{x})$ for a given N. Every party i with $x_i = 1$ sets a variable r_i to 0 or 1 each with probability $1/2$, and sends r_i to all parties (by taking $N-1$ rounds); every party i with $x_i = 0$ sets variable r_i to "$*$" (meaning "don't-care") and sends r_i to all parties. If there are no parties whose input bit is 1 (i.e., $|\vec{x}| = 0$), then every party receives only "$*$". If there is a single party whose input bit is 1 (i.e., $|\vec{x}| = 1$), then no party receives both 0 and 1 (more strictly, either every party receives 0 and $*$ or every party receives 1 and $*$). If there are at least two parties whose input bit is 1 (i.e., $|\vec{x}| \geq 2$), then every party receives both 0 and 1 (and $*$) with at least some constant probability. Let us call the case where every party receives both 0 and 1, *success*. We quantize this randomized algorithm. If the parties were able to amplify the amplitude of the success state to one, then they could decide whether $|\vec{x}| \geq 2$ or not with probability one by simply making measurement. This straightforward idea, however, fails since the exact amplitude amplification requires the exact value of the probability of the success state (before amplification) but it depends on unknown $|\vec{x}|$ in our case. To settle this problem, the parties run the exact amplitude amplification on the assumption that $|\vec{x}| = t$ for each $t \in \{2,\ldots,N\}$ in parallel. Consequently, they obtain a set $\{r_1,\ldots,r_n\}$ in which there are both 0 and 1 (and $*$) at least for $t = |\vec{x}|$ if $|\vec{x}| \geq 2$, whereas the probability of getting such a set of r_i's is zero for every t if $|\vec{x}| \leq 1$. To implement the reflection operator against the all-zero state, the parties invoke $\mathcal{H}_0$ as in the case of Theorem 1.1. To implement the reflection operator against the success state, the parties need to solve a seemingly new problem: the problem of deciding whether both 0 and 1 are included in $\{r_1,\ldots,r_n\}$. This problem is in fact reducible to computing H_0. Eventually, all the communication is devoted to computing H_0. The complexity is obtained by the same argument as in the case of Theorem 1.1.

1.4 Applications

Theorem 1.2 provides a way of developing a quantum algorithm that exactly computes H_1, i.e., exactly solves the solitude verification problem (SV), by plugging in an algorithm that exactly computes H_0, i.e., exactly solves the agreement verification problem (AV). For solving AV, there is an almost trivial deterministic algorithm that runs in $O(\Delta)$ rounds with bit complexity $O(m\Delta)$ for the number m of edges and an upper bound Δ of the diameter of the underlying graph[2] (see, e.g., Corollary 2 in Ref. [28]). Since this classical algorithm can be converted into a quantum algorithm with the same complexity up to a constant factor in a standard way, Theorem 1.2 yields the following algorithm for solving SV.

Corollary 1.4. *The solitude verification problem can exactly be solved in $O(N)$ rounds with bit complexity $O(mN^2)$ on any anonymous quantum network whose underlying graph is an unknown undirected graph,*

if an upper bound N of the number of parties is given to every party, where m is the number of edges of the underlying graph.

This is the first quantum algorithm that exactly solves SV (i.e., exactly computes H_1) with a round complexity linear in N on any anonymous quantum network. Moreover, its bit complexity has the same order as those of the bit-efficient quantum algorithms obtained from the existing leader election algorithms found in Refs. [34, 35] (in the classical setting, recall that it is impossible in general to exactly solve SV [5, 24] for a given upper bound N). With the promise that the exact number n of parties is given as the input N, there exists an $O(n)$-round classical algorithm, but the bit complexity of ours is smaller than those of the existing algorithms with linear/super-linear round complexity [28, 33, 34, 35].

Corollary 1.3 yields the following quantum leader election algorithm.

Corollary 1.5. *The leader election problem can exactly be solved in $O(N)$ rounds with bit complexity $O(mN^3)$ on any anonymous quantum network whose underlying graph is an unknown undirected graph, if an upper bound N of the number of parties is given to every party, where m is the number of edges of the underlying graph. Furthermore, with the promise that the number n of parties is given instead of N, the problem can exactly be solved in $O(n)$ rounds with bit complexity $O(mn^2)$.*

This is the first *linear-round* quantum algorithm that exactly solves the leader election problem with the bit complexity bounded by a polynomial in N (or n) for any anonymous quantum network. Our algorithm also beats the existing round-efficient algorithm [34] in bit complexity: the existing round-efficient algorithm has the round complexity $O(n \log n)$ ($O(N \log N)$) and the bit complexity $\tilde{O}(n^6)$ ($\tilde{O}(N^7)$) for given n (N). On the other hand, our algorithm is incomparable with the existing bit-efficient algorithm [35] (which has the round complexity $O(N^2)$ and the bit complexity $O(mN^2)$) when N is given, but it is not worse, up to a constant factor, even in terms of bit complexity when n is given.

Recall that once a unique leader is elected, it is possible to solve various problems by gathering all the distributed inputs in the network to the leader. For instance, Corollary 1.5 implies that the first linear-round quantum algorithm that, for a given n, exactly computes any computable Boolean function over n distributed bits with the bit complexity $O(mn^2)$, which is smaller than those of existing algorithms [37, 28, 35], in the worst case over all (computable) Boolean functions and undirected network topologies (a precise statement is given as Corollary 5.3 in Sec. 5.2). This can easily be generalized to the problem of quantum state transformation, in which the goal is for all parties to share a quantum state that is computable from a given initial state shared by all parties, as mentioned at the end of Sec. 5.2.

Theorems 1.1 and 1.2 are bridges between leader election and computing symmetric functions. This makes it possible to say more about the complexities in the case where more information is given to each party as input. Assume that each party knows the underlying graph, from which each party can also know the number n of parties, but the party does not know which node in the graph the party is identified with. Note that even in this setting, there are still an infinite number of graphs, such as rings, for which a unique leader cannot exactly be elected in the classical setting. For particular classes of graphs, it is known that H_1 can be computed as efficiently as H_0. In this case, a direct application of Theorem 1.1 gives a better bound: For a ring network, both H_0 and H_1 can be computed in $O(n)$ rounds with bit complexity $O(n^2)$, which implies that a unique leader can exactly be elected with these complexities. More generally, Kranakis et al. [28] developed a classical algorithm that exactly computes any symmetric function on any anonymous network with the underlying graph known, by derandomizing a random-walk

technique; the algorithm has a small bit complexity, when the stochastic matrix P of the random walk on the underlying graph augmented with self-loops has a small second eigenvalue (in the absolute sense). By using the straightforward quantization of the algorithms, for computing H_0 and H_1, shown as Theorem 7 in Ref. [28] (presented in Appendix A for completeness), Theorem 1.1 yields an efficient algorithm in terms of bit complexity for graphs with a large eigenvalue gap, such as toruses and hypercubes.

Corollary 1.6. *Let G be an undirected graph with n nodes and let G' be the graph G with self-loops added to each node. Let λ be the second largest eigenvalue (in absolute value) of the stochastic matrix P associated with G'. There is an algorithm that exactly elects a unique leader in $O\left(\frac{\log n}{\log(1/\lambda)}\right)$ rounds with bit complexity $O\left(\frac{m}{\log(1/\lambda)}(\log n)^2\right)$ on an anonymous quantum network with the underlying graph G, where m is the number of edges of G, if every party knows G.*

For instance, suppose that G is a $\log n$-dimensional hypercube with n vertices, where n is a power of two. Since the eigenvalues of the stochastic matrix for G are $1-\frac{2i}{\log n}$ for $i \in \{0,1,\ldots,\log n\}$, the second largest eigenvalue for G' is $\frac{\log n}{1+\log n}\left(1-\frac{2}{\log n}\right)+\frac{1}{1+\log n}=\frac{\log n-1}{\log n+1}$ (see Corollary 10 in Ref. [28]). We thus have $\log(1/\lambda) \geq \frac{2}{\log n+1}$. Since $m=\frac{1}{2}n\log n$, Corollary 1.6 implies that a unique leader can exactly be elected in $O((\log n)^2)$ rounds with bit complexity $O(n(\log n)^4)$. We should note that the round complexity in this case is super-linear in the diameter $\log n$ of the hypercube. If we apply Corollary 1.3 with the underlying graph G as an additional input, we obtain a quantum algorithm for leader election that runs in $O(\Delta)=O(\log n)$ rounds but it requires a larger bit complexity: $O(mn\Delta)=O(n^2(\log n)^2)$, where Δ is the diameter of the underlying graph and we use the simple algorithm for computing H_0 that runs in $O(\Delta)$ rounds and with bit complexity $O(m\Delta)$.

1.5 Related Work

Pal et al. [32] and D'Hondt and Panangaden [17] have dealt with the leader election and GHZ-state sharing problems in a different setting where pre-shared entanglement is assumed but only classical communication is allowed. Gavoille et al. [20] discussed the relation between several network models, including the anonymous quantum networks we deal with, which differ in available quantum resources. Buhrman et al. [12] studied distributed quantum computing over non-anonymous networks consisting of a non-constant number of parties. For fault-tolerant distributed quantum computing, Ben-Or and Hassidim [7] showed an unbounded quantum-classical separation in terms of the expected number of rounds for the Byzantine agreement problem. In a cryptographic setting where there may exist cheating parties, there have been a lot of studies related to multi-party communication, such as anonymous quantum communication [14, 10], secure multi-party quantum computation [15], multi-party quantum coin-flipping [4, 3], and leader election with cheating parties [19].

1.6 Organization

Section 2 describes the network model and some notations, and defines the problems considered in this paper. Sections 3 and 4 prove Theorems 1.1 and 1.2, respectively. Section 5 then presents exact quantum algorithms for solitude verification, and leader election, and computing Boolean functions as the corollaries of the theorems. Section 6 concludes the paper.

2 Preliminaries

We denote by $[p]$ the set $\{1,\ldots,p\}$ for any positive integer p.

2.1 Network Model

A classical *network* consists of multiple parties and *bidirectional* classical communication links between them. In a quantum network, every party can perform quantum computation and communication, and each pair of adjacent parties has a *bidirectional* quantum communication link between them (we do not assume any prior shared entanglement).

When the parties and links are regarded as nodes and edges, respectively, the topology of the network is expressed by a connected undirected graph. We denote by $\mathcal{G}_n$ the set of all n-node connected undirected graphs where n is at least two. In what follows, we may identify each party/link with its corresponding node/edge in the underlying graph for the network. Every party has *ports* corresponding one-to-one to communication links incident to the party. Every port of party l has a unique label $i \in [d_l]$, where d_l is the number of parties adjacent to l. More formally, the underlying graph $G = (V,E)$ includes a *port-numbering* [37], which is a set $\sigma = \{\sigma_v : v \in V\}$ of functions such that, for each node v of degree d_v, σ_v is a bijection from the set of edges incident to v to $[d_v]$. It is stressed that each function σ_v may be defined independently of any other $\sigma_{v'}$ with $v' \neq v$. In our model, each party knows the number of its ports and the party can appropriately choose one of its ports whenever it transmits or receives a message.

Every party p initially has local information $\mathcal{I}_p$ (i.e., the information that only party p knows such as its local state and the number of adjacent parties), and global information $\mathcal{I}_G$ [i.e., the information shared by all parties (if it exists), such as the number of parties in the system].[3] Without loss of generality, we assume that every party p runs the same algorithm for local and global information, $\mathcal{I}_p$ and $\mathcal{I}_G$, given as its arguments. If all parties have the same local information except for the number of their ports (equivalently, the number of communication links they have), the network and the parties thereof are said to be *anonymous*. For instance, if the underlying graph of an anonymous network is regular, this is essentially equivalent to a situation in which every party has the same identifier (since we can regard $\mathcal{I}_p$ as party p's identifier). This paper deals only with anonymous networks, but may refer to a party with its index (e.g., party i) solely for the purpose of providing a simple description.

A network is either *synchronous* or *asynchronous*. In the synchronous case, message passing is performed synchronously. The unit interval of synchronization is called a *round*. Following the approach in Ref. [29], one round consists of the following two sequential steps, where we assume that two (probabilistic) procedures that generate messages and change local states are defined in the common algorithm invoked by each party: (1) each party changes its local state according to a procedure that takes its current local state and the incoming messages as input, and then removes the messages from its ports; (2) each party then prepares messages and decides the ports through which the messages should be sent by using the other procedure that takes its current local state as input, and finally sends the messages out via the ports. Notice that, in the quantum setting, the two procedures are physically realizable operators (i.e., trace-preserving completely-positive super-operators acting on positive semidefinite operators in a

[3] There may be some information shared by some but not all parties. We may regard such information as local information, since we are interested in the worst case.

complex Euclidean space, which will be defined later). In an asynchronous network, messages can be sent out at any time and they reach their destinations within a finite but unknown time. The number of rounds required by an algorithm is defined by convention as the length of the longest chains of messages sent during the execution of the algorithm (intuitively, the number of rounds is the number of steps that cannot be performed simultaneously). In this paper, we describe our algorithms in synchronous networks for simplicity. Since every party sends a message via every communication link in each round in our algorithm, our algorithm can be emulated in an asynchronous network without sacrificing the communication cost just by delaying the local operation that would be performed in each round in the synchronous setting until a message arrives at every port. As complexity measures, we consider round complexity (i.e., the required number of rounds) and bit complexity (i.e., the number of bits and qubits communicated over all communication links). In this paper, we do not assume any faulty party or any faulty communication link.

2.2 Distributed Computing Problems

Once parties collaborate to assign a unique identifier, i.e., assign a distinct number from a fixed domain, to each party in an anonymous network, the network will obviously become non-anonymous. This task is classically reducible to electing a unique leader (with small additional communication cost): a unique leader first constructs a spanning tree in $O(n)$ rounds with bit complexity $O(m)$ for the number m of all communication links, and then assigns a distinct number to each party by sending out a message with a counter of $\lceil \log n \rceil$ bits that traverses the spanning tree in $O(n)$ rounds with bit complexity $O(n \log n)$. The *leader election problem* is formally defined as follows: Suppose that there is an n-party network whose underlying graph is in $\mathcal{G}_n$, and that each party $i \in [n]$ in the network has a variable y_i initialized to 1. The goal is to create a situation in which $y_k = 1$ for a certain $k \in [n]$ and $y_i = 0$ for every i in the rest $[n] \setminus \{k\}$. We denote the leader election problem with n parties by LE_n.

Another distributed computing problem we will discuss is that of *verifying* that a unique leader exists in a network of n parties. This problem is called the *solitude verification problem* (SV_n) and is formally defined as follows: Suppose that there is an n-party network whose underlying graph is in $\mathcal{G}_n$, and that each party $i \in [n]$ in the network has a variable y_i initialized to true. Given $x_i \in \{0,1\}$ as input, the goal of each party i is to set $y_i = \mathsf{true}$ if there is exactly one $j \in [n]$ such that $x_j = 1$, and set $y_i = \mathsf{false}$ otherwise. Yet another problem is that of verifying that every party has a common value in a network of n parties. This problem is called the *agreement verification problem* (AV_n) and is formally defined as follows: For a fixed domain D, suppose that there is an n-party network whose underlying graph is in $\mathcal{G}_n$, and that each party $i \in [n]$ in the network has a variable y_i initialized to true. Given $x_i \in D$ as input, the goal of each party i is to set $y_i = \mathsf{true}$ if there exists $d \in D$ such that $x_j = d$ for every j, and set $y_i = \mathsf{false}$ otherwise. We assume that $D = \{0,1\}$ in this paper. We should note that SV_n is equivalent to deciding whether or not the sum over all x_j is exactly 1, and AV_n is equivalent to deciding whether or not the sum over all x_j is exactly 0 when $D = \{0,1\}$ (ignoring the constant factor gap of the complexities).

We will consider the three problems LE_n, SV_n and AV_n on anonymous quantum networks. In general, the hardness of computing problems depends on port-numbering as well as the underlying graphs of networks. We thus consider the worst case over all port-numberings and the underlying graphs for each n (i.e., the number of parties) or each N (i.e., an upper bound of n), unless stated otherwise.

Here, we briefly review the known facts as regards exactly computing these problems on anonymous

networks: No classical algorithm can exactly solve LE_n on anonymous classical networks with the underlying graphs in a certain broad family of graphs [5, 37, 9]. More precisely, as shown by Yamashita and Kameda [37], the necessary and sufficient condition of anonymous classical networks for exactly solving LE_n is completely described by the number of non-isomorphic subtrees of a certain height depending on n in the rooted labeled tree known as the universal cover [31] or the view [37]. Since each party can construct the view from the information gathered by repeating message exchanges, the parties can decide whether LE_n is solvable or not, and moreover, they can elect a unique leader whenever LE_n is solvable, if n is given to each party. This is impossible, however, when only an upper bound N of n is given. More generally, for anonymous classical networks, a problem can be solved solely by using the information obtained from the view if and only if there exists an algorithm that exactly solves the problem; for the view contains as much information as the parties can gather by exchanging classical messages (we refer readers to [28, 31, 33, 21] for studies related to efficient algorithms that construct the view). Computing symmetric Boolean functions over n distributed bits would be the most typical problem that can exactly be solved on classical anonymous networks if n is known, since the number of ones, i.e., the Hamming weight, of the n bits can be computed from the view for the given n [28, 37, 33]. In this sense, the problem SV_n, which is equivalent to computing H_1, on anonymous network is easier than LE_n. On the other hand, when only an upper bound N is given, no classical algorithm can solve SV_n (more generally, any symmetric Boolean function other than H_0 or H_n) even with zero error (i.e., without error but with no time bound) [5, 24]. The problem AV_n, which is equivalent to H_0, can be solved exactly on any anonymous network in $O(N)$ rounds with bit complexity $O(Nm)$ by a simple classical algorithm if only an upper bound N of n is given (see Section 5.1). Finally, we mention that there are several sophisticated classical algorithms when the underlying graph is known to be in special classes, such as rings [6], hypercubes [27, 28], and distance regular graphs [28].

2.3 Quantum Computing/Communication

A *qubit* is a unit of quantum information. Mathematically, the quantum state of k qubits is represented as a positive semidefinite operator acting on the Hilbert space $\mathbb{C}^{2^k}$ with trace 1. The operator is called a *density operator*. In particular, if the operator has rank 1, the corresponding state is said to be *pure*. In this case, the state is essentially a vector in the space $\mathbb{C}^{2^k}$. In this paper, it is sufficient to assume that quantum states are pure. Physically realizable *super-operators* are completely positive linear operators that preserve the trace of quantum states (i.e., density operators) on which they act. We also say that they are *admissible* super-operators or *quantum channels*. It is known that the action of any admissible super-operator on k qubits is equivalent to performing the super-operator induced by a unitary operator acting on the k qubits plus ancilla qubits, and then discarding a part of all the qubits. In other words, it is sufficient to assume unitary operators and orthogonal projectors with ancillary qubits. For further details, see standard textbooks (e.g., [30, 26, 25]). In a quantum network, local quantum computation in each round is performed by fixed super-operators as in the definition of the network model.

Quantum communication is performed as follows. When sending quantum messages at the end of each round, each party sends out one of its quantum registers through one of its ports, where which register is to be sent through which port is designated by certain output registers of the local quantum computation. When receiving quantum messages at the beginning of each round, each party stores the message arriving through each of its ports in one of its quantum registers. We should note that in each

round, each party can record which register was received through which port, and which register was sent through which port. This makes it possible for each message to be sent back. Any quantum algorithm can thus be inverted unless it makes intermediate measurements.

The following well-known theorem is called (a special case of) *exact (quantum) amplitude amplification*, and this theorem will be used repeatedly.

Theorem 2.1 ([11, 13])**.** *Let $\chi\colon \{0,1\}^n \to \{\mathsf{true},\mathsf{false}\}$ be any Boolean function. Let A be a unitary operator acting on $(n+m)$ qubits for some non-negative integer m and suppose that $|\Psi\rangle = A|0\rangle^{\otimes(n+m)} = \sum_z \alpha_z|z\rangle|\psi_z\rangle$ for the orthonormal basis $\{|z\rangle\colon z\in\{0,1\}^n\}$ and the set of certain m-qubit pure states $\{|\psi_z\rangle\colon z\in\{0,1\}^n\}$. Define the unitary operator acting on $(n+m)$ qubits as*

$$Q(A,\chi,\phi,\theta) = -AF_0(\phi)A^{-1}F_\chi(\theta),$$

where $F_\chi(\theta)$ is a unitary operator that multiplies $|z\rangle|y\rangle$ by a factor of $e^{\mathrm{i}\theta}$ for every $y\in\{0,1\}^m$ if $\chi(z)=\mathsf{true}$ and acts as the identity operator otherwise, and $F_0(\phi)$ is a unitary operator that multiplies $|0\rangle^{\otimes(n+m)}$ by a factor of $e^{\mathrm{i}\phi}$ and acts as the identity operator otherwise. Let $a=\sum_{z\colon \chi(z)=\mathsf{true}}|\alpha_z|^2$ be the probability of obtaining any one of z with $\chi(z)=\mathsf{true}$ as an outcome by measuring the first n qubits of $|\Psi\rangle$. If the value a is exactly known and at least $1/4$, then we have

$$Q(A,\chi,\phi_a,\theta_a)|\Psi\rangle = \frac{1}{\sqrt{a}}\sum_{z\colon \chi(z)=\mathsf{true}}\alpha_z|z\rangle|\psi_z\rangle$$

for some values ϕ_a and θ_a $(0\le\phi_a,\theta_a\le 2\pi)$ determined by a.

We call a the *initial success probability* of A.

2.4 Notations

A Boolean function $f\colon \{0,1\}^n\to\{\mathsf{true},\mathsf{false}\}$ depending on n variables, $x_1,\dots,x_n$ with $x_i\in\{0,1\}$, is said to be *symmetric* if f is determined by the Hamming weight $|\vec{x}|$ of $\vec{x}=(x_1,\dots,x_n)$, where $|\vec{x}|=\sum_{i=1}^n x_i$. In particular, the symmetric function $H_k\colon \{0,1\}^n\to\{\mathsf{true},\mathsf{false}\}$ is defined as $H_k(\vec{x})=\mathsf{true}$ if and only if $|\vec{x}|$ is k. We say that an algorithm *exactly computes a Boolean function* $f\colon \{0,1\}^n\to\{\mathsf{true},\mathsf{false}\}$ over an n-party network, if every party $i\in[n]$ in the network initially has a variable y_i (initialized to "true") and input $x_i\in\{0,1\}$, and the algorithm sets y_i to $f(\vec{x})$ with certainty after computation, which takes a certain bounded time. If a quantum algorithm exactly computes f without intermediate measurements[4] on an anonymous quantum network, we say that the algorithm is an *f-algorithm*. Suppose that every party i has two one-qubit registers R_i and S_i, a register I_i whose content is (part of) the pair of local and global

[4]In general, intermediate measurements performed in quantum algorithms can be deferred to the end. This does not change complexity in certain models such as the query complexity model. With distributed computing, however, this may increase the bit complexity. For instance, consider a measurement with two possible outcomes 1 or 2 and suppose that an algorithm sends a message out via port $p\in\{1,2\}$ if the outcome is p. In this case, the algorithm sends out a single message at this time, while, if the measurement is deferred, the algorithm must be modified so that it sends one message via each of the two ports. We thus consider algorithms with and without intermediate measurements to be different.

information, $(\mathcal{I}_i, \mathcal{I}_G)$, and an ancillary register A_i. In general, an f-algorithm is a unitary operator that performs the following mapping:

$$\left[\bigotimes_{i=1}^{n}(|x_i\rangle_{\mathsf{R}_i}|(\mathcal{I}_i,\mathcal{I}_G)\rangle_{\mathsf{I}_i}|\mathsf{true}\rangle_{\mathsf{S}_i}|\vec{0}\rangle_{\mathsf{A}_i})\right] \mapsto \left[\bigotimes_{i=1}^{n}(|x_i\rangle_{\mathsf{R}_i}|(\mathcal{I}_i,\mathcal{I}_G)\rangle_{\mathsf{I}_i}|f(\vec{x})\rangle_{\mathsf{S}_i})\right] \otimes |g_{\vec{x}}\rangle_{\mathsf{A}},$$

for any $\vec{x} = (x_1,\ldots,x_n) \in \{0,1\}^n$, where A is the ordered set of the registers, $(\mathsf{A}_1,\ldots,\mathsf{A}_n)$, and $|g_{\vec{x}}\rangle$ is "garbage" left in A after computing $f(\vec{x})$. Since all communication links are bidirectional, messages can be sent back by recording the ports that were used to send or receive each message. Thus any f-algorithm is reversible (as it has no intermediate measurements). Hence we can remove the "garbage" with the standard garbage-erasing technique (i.e., by copying the value of $f(\vec{x})$ to a fresh register and then inverting the f-algorithm). We may hence assume without loss of generality (at the cost of doubling round and bit complexities) that any f-algorithm transforms

$$\sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} \bigotimes_{i=1}^{n}(|x_i\rangle_{\mathsf{R}_i}|(\mathcal{I}_i,\mathcal{I}_G)\rangle_{\mathsf{I}_i}|\mathsf{true}\rangle_{\mathsf{S}_i}) \mapsto \sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} \bigotimes_{i=1}^{n}(|x_i\rangle_{\mathsf{R}_i}|(\mathcal{I}_i,\mathcal{I}_G)\rangle_{\mathsf{I}_i}|f(\vec{x})\rangle_{\mathsf{S}_i}),$$

for any $\alpha_{\vec{x}} \in \mathbb{C}$ with $\sum_{\vec{x}\in\{0,1\}^n} |\alpha_{\vec{x}}|^2 = 1$, where $\vec{x} = (x_1,\ldots,x_n)$. In our algorithms, every party will use f-algorithms as subroutines for several functions f *in a black box manner*, meaning that every party i simply gives the registers R_i, S_i and I_i as their input and, after performing the f-algorithms, gets the registers back for its later use (we assume that the f-algorithms dynamically allocate the registers A_i of appropriate size); in other words, no party uses any information about how the f-algorithms work. For a more general function $f\colon X^n \to Y$ depending on distributed n variables $(x_1,\ldots,x_n)$ with $x_i \in X$, we say similarly that a quantum algorithm is an f-algorithm, if the algorithm exactly computes f without intermediate measurements on an anonymous quantum network. Note that $\mathcal{I}_i$ always includes d_i, the number of ports of the party i. We will hence omit d_i from the descriptions of algorithms for simplicity.

For an f-algorithm $\mathcal{F}$ on an anonymous quantum network with the underlying graph $G \in \mathcal{G}_n$, we denote by $Q^{\mathrm{bit}}_G(\mathcal{F})$ and $Q^{\mathrm{rnd}}_G(\mathcal{F})$ the worst-case bit and round complexities, respectively, of $\mathcal{F}$ over all possible superpositions of classical inputs. For simplicity, we may write $Q^{\mathrm{bit}}(\mathcal{F})$ and $Q^{\mathrm{rnd}}(\mathcal{F})$ if G is clear from the context. Although $Q^{\mathrm{bit}}_G(\mathcal{F})$ and $Q^{\mathrm{rnd}}_G(\mathcal{F})$ may vary according to the local and global information given to every party before invoking the algorithm, we do not explicitly mention it in the notation as it will be clear from the context.

Finally, we emphasize that we will add subscripts to registers' names only for the purpose of clarifying the owner of each register in describing algorithms. For instance, we will use R_i and S_i as registers' names to mean that they are owned by the party i. We should note, however, that in fact the party does not have any index in anonymous networks, in other words, the party cannot see the subscript i of R_i and S_i. The party can thus distinguish only between local registers (including the registers that it has already received via communication). Moreover, the operations that the party can perform must be independent of the index i. The reader will be able to verify that this condition is satisfied in our algorithms.

3 Proof of Theorem 1.1

Recall that every party knows only an upper bound N of the number n of parties. For the ease of understanding, however, we first assume that every party knows the number n of parties. We then

generalize the arguments at the end of the section.

3.1 Basic Idea

Suppose that every party is initially eligible to be the leader and is given the number n of parties as input. Every party then flips a coin that lands on heads with probability $1/n$ and tails with $1-1/n$. If exactly one party sees heads, that party becomes a unique leader. The probability of this successful case is given by

$$s(n) = \binom{n}{1} \cdot \frac{1}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} = \left(1-\frac{1}{n}\right)^{n-1} > \frac{1}{e} > \frac{1}{4}.$$

We will amplify the probability of this case to one by applying the exact quantum amplitude amplification stated in Theorem 2.1. To do this, we use an H_1-algorithm in a black-box manner to check in $F_\chi(\theta_{s(n)})$ whether or not a run of the above randomized algorithm has succeeded and use an H_0-algorithm in a black-box manner to realize the diffusion operator $F_0(\phi_{s(n)})$. In other words, we quantumly reduce the leader election problem to computing H_0 and H_1. In our algorithm, all communication is performed for computing H_0, H_1 and their inversions. The only non-trivial part is how to implement $F_\chi(\theta_{s(n)})$ and $F_0(\phi_{s(n)})$ for $s(n) = (1-1/n)^{n-1}$ in a distributed way on an anonymous network, i.e., in such a distributed way that every party with the same number of communication links runs the same algorithm.

3.2 The Algorithm

Before describing the algorithm, we define *solving* and *unsolving* strings. Suppose that each party i has a bit $x_i \in \{0,1\}$, i.e., the n parties share an n-bit string $\vec{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$. A string $\vec{x}$ is said to be *solving* if $\vec{x}$ has Hamming weight one, i.e., $|\vec{x}| = 1$. Otherwise, $\vec{x}$ is said to be *unsolving*. We also say that an n-qubit pure state $|\psi\rangle = \sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} |\vec{x}\rangle$ shared by the n parties is *solving (unsolving)* if $\alpha_{\vec{x}} \neq 0$ only for $\vec{x}$ that is solving (unsolving) (there are pure states that are neither solving nor unsolving, but we do not need to define such states to describe the algorithms).

Fix any H_0-algorithm and H_1-algorithm. We use these in a black-box manner as follows.

Base algorithm: Let B be a two-by-two unitary matrix defined by

$$B = \frac{1}{\sqrt{n}} \begin{pmatrix} \sqrt{n-1} & 1 \\ 1 & -\sqrt{n-1} \end{pmatrix}.$$

At the beginning of the algorithm, each party i prepares three single-qubit quantum registers R_i, S_i, and S'_i, where the qubit in R_i is initialized to $|0\rangle$, the qubits in S_i and S'_i are initialized to $|\text{“true”}\rangle$ (the qubits in S_i and S'_i will be used as ancillary qubits when performing phase-shift operations on the qubit in R_i). Each party i first applies B to the qubit in R_i to create the quantum state $|\psi\rangle_{\mathsf{R}_i} = B|0\rangle_{\mathsf{R}_i} = \sqrt{1-\frac{1}{n}}\,|0\rangle_{\mathsf{R}_i} + \sqrt{\frac{1}{n}}\,|1\rangle_{\mathsf{R}_i}$. Let $A = B^{\otimes n}$. All n parties then share the n-qubit quantum state

$$|\Psi\rangle_{\mathsf{R}} = \bigotimes_{i=1}^{n} |\psi\rangle_{\mathsf{R}_i} = A \bigotimes_{i=1}^{n} |0\rangle_{\mathsf{R}_i} = \bigotimes_{i=1}^{n} \left(\sqrt{1-\frac{1}{n}}\,|0\rangle_{\mathsf{R}_i} + \sqrt{\frac{1}{n}}\,|1\rangle_{\mathsf{R}_i} \right)$$

Input: classical variable $\mathbf{status}_i :=$ "eligible", and integer t.
Output: classical variable $\mathbf{status}_i \in \{$"eligible","ineligible"$\}$.
Notation: $\mathsf{R} = (\mathsf{R}_1,\ldots,\mathsf{R}_n)$, $\mathsf{S} = (\mathsf{S}_1,\ldots,\mathsf{S}_n)$, $\mathsf{S}' = (\mathsf{S}'_1,\ldots,\mathsf{S}'_n)$, and $\mathsf{I} = (\mathsf{I}_1,\ldots,\mathsf{I}_n)$.

1. Prepare single-qubit registers R_i, S_i, S'_i and a $\lceil \log t \rceil$-qubit register I_i,
   and initialize R_i, S_i, S'_i, and I_i to the states $|0\rangle$, $|$"true"$\rangle$, $|$"true"$\rangle$, and $|t\rangle$, respectively.
2. $[A]$: Apply $B = \frac{1}{\sqrt{t}}\left(\begin{smallmatrix} \sqrt{t-1} & 1 \\ 1 & -\sqrt{t-1} \end{smallmatrix}\right)$ to R_i to generate $|\psi\rangle_{\mathsf{R}_i} = \sqrt{\frac{t-1}{t}}|0\rangle_{\mathsf{R}_i} + \sqrt{\frac{1}{t}}|1\rangle_{\mathsf{R}_i}$.
3. Perform $AF_0(\phi_{s(t)})A^\dagger F_\chi(\theta_{s(t)})$ consisting of the following steps, where $s(t) = (1-1/t)^{t-1}$ and $\mathbb{I} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$:
   - 3.1 $[F_\chi(\theta_{s(t)})]$: Perform the following steps:
     - 3.1.1 Run an H_1-algorithm with $(\mathsf{R}_i, \mathsf{S}_i, \mathsf{I}_i)$
       (for the purpose of computing H_1 over the contents of R, and storing the result in S_i for each $i \in [n]$).
     - 3.1.2 Multiply the phase by $\exp(\mathrm{i}\frac{1}{t}\theta_{s(t)})$ if the content of S_i is "true".
     - 3.1.3 Invert every computation and communication of Step 3.1.1 to disentangle S.
   - 3.2 $[A^\dagger]$: Invert the computation of Step 2.
   - 3.3 $[F_0(\phi_{s(t)})]$: Perform the following steps:
     - 3.3.1 Run an H_0-algorithm with $(\mathsf{R}_i, \mathsf{S}'_i, \mathsf{I}_i)$
       (for the purpose of computing H_0 over the contents of R, and storing the result in S'_i for each $i \in [n]$).
     - 3.3.2 Multiply the phase by $\exp(\mathrm{i}\frac{1}{t}\phi_{s(t)})$ if the content of S'_i is "true".
     - 3.3.3 Invert every computation and communication of Step 3.3.1 to disentangle S'.
   - 3.4 $[A]$: Perform the same operation as that performed in Step 2.
4. Measure R_i with respect to the basis $\{|0\rangle, |1\rangle\}$, and set $\mathbf{status}_i$ to "ineligible" if the outcome is 0.
5. Output $\mathbf{status}_i$.

Figure 1: Algorithm QLE(t) (the operations performed by each party $i \in [n]$).

in $\mathsf{R} = (\mathsf{R}_1,\ldots,\mathsf{R}_n)$. Let $S_n = \{\vec{x} \in \{0,1\}^n : |\vec{x}| = 1\}$ be the set of solving strings of length n, and let $|\Psi_{\mathsf{solving}}\rangle = \frac{1}{\sqrt{n}}\sum_{\vec{x}\in S_n}|\vec{x}\rangle$ be the uniform superposition of solving strings of length n. Notice that $|\Psi\rangle_{\mathsf{R}}$ is a superposition of the solving state $|\Psi_{\mathsf{solving}}\rangle$ and some unsolving state $|\Psi_{\mathsf{unsolving}}\rangle$:

$$|\Psi\rangle_{\mathsf{R}} = \alpha_{\mathsf{solving}}|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}} + \alpha_{\mathsf{unsolving}}|\Psi_{\mathsf{unsolving}}\rangle_{\mathsf{R}}.$$

The amplitude $\alpha_{\mathsf{solving}}$ of $|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}}$ is given by $\alpha_{\mathsf{solving}} = \sqrt{s(n)} > 1/2$.

Exact amplitude amplification: Now the task for the n parties is to amplify the amplitude of $|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}}$ to one by using Theorem 2.1, which involves one application of $-AF_0(\phi_a)A^{-1}F_\chi(\theta_a)$ for $A = B^{\otimes n}$ since the initial success probability is $\alpha^2_{\mathsf{solving}} = s(n) > 1/4$ (recall $s(n) = (1-1/n)^{n-1}$). Define $\chi(\vec{x}) = 1$ if $\vec{x}$ is solving and $\chi(\vec{x}) = 0$ otherwise.

To realize $F_\chi(\theta_{s(n)})$ in a distributed manner, each party multiplies the amplitude of any basis state $|\vec{x}\rangle_{\mathsf{R}}$ with $\chi(\vec{x}) = 1$ by a factor of $e^{\mathrm{i}\frac{1}{n}\theta_{s(n)}}$ in the following way. This multiplies the amplitude of the basis state by a factor of $e^{\mathrm{i}\theta_{s(n)}}$ as a whole.

First every party needs to compute $\chi(\vec{x})$. For this, every party i runs the H_1-algorithm with R_i and S_i, which sets the content of S_i to "true" if the number of 1's among the contents of $\mathsf{R} = (\mathsf{R}_1, \ldots, \mathsf{R}_n)$ is exactly one and sets it to "false" otherwise (recall that the H_1-algorithm computes H_1 for each basis state in a superposition). This operation transforms the state as follows:

$$|\Psi\rangle_{\mathsf{R}}(|\text{“true”}\rangle^{\otimes n})_{\mathsf{S}} \mapsto \alpha_{\mathsf{solving}}|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}}(|\text{“true”}\rangle^{\otimes n})_{\mathsf{S}} + \alpha_{\mathsf{unsolving}}|\Psi_{\mathsf{unsolving}}\rangle_{\mathsf{R}}(|\text{“false”}\rangle^{\otimes n})_{\mathsf{S}},$$

where $\mathsf{S} = (\mathsf{S}_1, \ldots, \mathsf{S}_n)$. Every party i then multiplies the amplitude of each basis state by a factor of $e^{\mathrm{i}\frac{1}{n}\theta_{s(n)}}$, if the content of S_i is "true" (here, no party i measures S_i; every party i simply performs a phase shift controlled by the qubit in S_i). Namely, the state of the qubits in R and S is transformed into

$$(e^{\mathrm{i}\frac{1}{n}\theta_{s(n)}})^n \alpha_{\mathsf{solving}}|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}}(|\text{“true”}\rangle^{\otimes n})_{\mathsf{S}} + \alpha_{\mathsf{unsolving}}|\Psi_{\mathsf{unsolving}}\rangle_{\mathsf{R}}(|\text{“false”}\rangle^{\otimes n})_{\mathsf{S}}.$$

Finally, every party inverts every computation and communication performed by the H_1-algorithm to disentangle S. The resulting state is $(e^{\mathrm{i}\theta_{s(n)}}\alpha_{\mathsf{solving}}|\Psi_{\mathsf{solving}}\rangle_{\mathsf{R}} + \alpha_{\mathsf{unsolving}}|\Psi_{\mathsf{unsolving}}\rangle_{\mathsf{R}})(|\text{“true”}\rangle^{\otimes n})_{\mathsf{S}}$.

The implementation of $F_0(\phi_{s(n)})$ is similar to that of $F_\chi(\theta_{s(n)})$, except that $F_0(\phi_{s(n)})$ multiplies the all-zero basis state $|0\cdots 0\rangle$ by $e^{\mathrm{i}\phi_{s(n)}}$. Every party i first runs the H_0-algorithm with R_i and S'_i, which sets the content of S'_i to "true" if the content of R_i is the all-zero string, and sets it to "false" otherwise. Every party i then multiplies the amplitude of the all-zero state by a factor of $e^{\mathrm{i}\frac{1}{n}\phi_{s(n)}}$, if the content of S'_i is "true". Finally, every party inverts every computation and communication performed by the H_0-algorithm to disentangle $\mathsf{S}' = (\mathsf{S}'_1, \ldots, \mathsf{S}'_n)$.

A pseudo-code is given in Figure 1, where the inputs to Algorithm QLE(t) are $\mathbf{status}_i :=$ "eligible" and $t := n$. After the execution of the algorithm, exactly one party i has the value "eligible" in $\mathbf{status}_i$. Since all communication is performed to compute H_0 and H_1 and their inversions, the algorithm runs in $O(Q_G^{\mathrm{rnd}}(\mathcal{H}_0) + Q_G^{\mathrm{rnd}}(\mathcal{H}_1))$ rounds with bit complexity $O(Q_G^{\mathrm{bit}}(\mathcal{H}_0) + Q_G^{\mathrm{bit}}(\mathcal{H}_1))$ for any graph $G \in \mathcal{G}_n$, where $\mathcal{H}_0$ and $\mathcal{H}_1$ are the H_0- and H_1-algorithms, respectively, that we fixed. This completes the proof of Theorem 1.1 when the number n of parties is given.

3.3 Generalization for a Given N

Now we consider the case where only an upper bound N of n is given to every party. Note that here we assume an H_0-algorithm and an H_1-algorithm that always output the correct values of H_0 and H_1, respectively, for any given upper bound N of n (such an H_0-algorithm is well-known and such an H_1-algorithm in the quantum setting will be given in Section 4). Let us denote by Algorithm QLE(t) the above quantum algorithm, which uses t instead of the number n as the input. The idea is that every party runs the following two steps in parallel for every $t = \{2, \ldots, N\}$ (since there are at least two parties by definition, it suffices to start at $t = 2$):

1. Run Algorithm QLE(t).
2. Run the H_1-algorithm with the upper bound N to verify that there is a unique leader.

Note that Algorithm QLE(t) always halts with output $\mathbf{status}_i$ for any integer $t \geq 2$, but it may fail to elect a unique leader for $t \neq n$ since the success probability of the coin flipping algorithm is no longer $s(t)$ (the H_0- and H_1-algorithms may also fail to output the correct answer for $t < n$).

For $t = n$, a unique leader is elected by Algorithm QLE(t) with certainty and thus the H_1-algorithm in Step 2 always outputs true. For each $t > n$, if the H_1-algorithm in Step 2 outputs true, this guarantees that a unique leader is elected (since we assumed that the H_1-algorithm always outputs the correct answer for any upper bound of n). With these observations, we can see that a unique leader can be elected with certainty by performing the following operation as the last step: Every party decides that the party elected by the instance for $t_{\max}$ is a leader, where $t_{\max}$ is the maximum of t for which the H_1-algorithm outputs true in Step 2. Note that all parties agree on $t_{\max}$, since for the given N, the H_1-algorithm always outputs the correct answer to each party and thus every party agrees on its output.

In summary, for a given upper bound N of n, the entire algorithm works in $O(Q_G^{\mathrm{rnd}}(\mathcal{H}_0) + Q_G^{\mathrm{rnd}}(\mathcal{H}_1))$ rounds with bit complexity $O(N(Q_G^{\mathrm{bit}}(\mathcal{H}_0) + Q_G^{\mathrm{bit}}(\mathcal{H}_1)))$ for any graph $G \in \mathcal{G}_n$, where $\mathcal{H}_0$ and $\mathcal{H}_1$ are the H_0- and H_1-algorithms, respectively, that we fixed.

4 Proof of Theorem 1.2

The proof consists of the following two steps:

- Reduce computing H_1 to computing H_0 and the consistency function C_S, where C_S is a Boolean function that is true if and only if a subset (specified by S) of parties has the same classical value (its formal definition will be given later).
- Reduce computing C_S to computing H_0.

Actually, the second step is almost trivial. We start with the first step. Recall that there are n parties in the network, and they initially know only an upper bound N of n.

4.1 Basic Idea

Suppose that every party i is given a Boolean variable x_i. We can probabilistically compute $H_1(\vec{x})$ with the following classical algorithm for a given N, where $\vec{x} = (x_1, \ldots, x_n)$: Every party i with $x_i = 1$ sets a variable r_i to 0 or 1 each with probability $1/2$ and sends r_i to all parties (by taking $N-1$ rounds); every party i with $x_i = 0$ sets variable r_i to "$*$" (meaning "don't-care") and sends r_i to all parties. It is not difficult to see that the following three hold: (i) if $|\vec{x}| = 0$, every party receives only "$*$", (ii) if $|\vec{x}| = 1$, either no party receives "1" or no party receives "0", and (iii) if $|\vec{x}| \geq 2$, every party receives both "0" and "1" with probability $1 - 2/2^{|\vec{x}|}$. Therefore, every party can conclude that $H_1(\vec{x}) = \mathsf{true}$ ($H_1(\vec{x}) = \mathsf{false}$) with probability one if $|\vec{x}| = 1$ ($|\vec{x}| = 0$) and that $H_1(\vec{x}) = \mathsf{false}$ with probability $1 - 2/2^{|\vec{x}|} \geq 1/2$ if $|\vec{x}| \geq 2$. Roughly speaking, our quantum algorithm for computing H_1 is obtained by first quantizing this probabilistic algorithm and then amplifying the amplitudes to boost the success probability to one. More concretely, we boost the success probability p that both 0 and 1 are included in $r_1, \ldots, r_n$. Seemingly, this could be done in a straightforward manner by using the exact amplitude amplification. Let us assume that p_{init} and p_{final} are the values of p before and after, respectively, the exact amplitude amplification. Obviously, if $p_{\mathrm{init}} = 0$, then $p_{\mathrm{final}} = 0$. It hence holds that $p_{\mathrm{final}} = 0$ for $|\vec{x}| \leq 1$, whereas when $|\vec{x}| \geq 2$, p could be boosted to $p_{\mathrm{final}} = 1$ if the exact value of p_{init} were known to every party. The probability $p_{\mathrm{init}} = 1 - 2/2^{|\vec{x}|}$ is, however, determined by $|\vec{x}|$, and computing $|\vec{x}|$ is not easier than just deciding whether $|\vec{x}| = 1$ or not. Therefore, instead of actual $|\vec{x}|$, we run the amplitude amplification for each $t := 2, \ldots, N$,

a *guess* at $|\vec{x}|$, in parallel. We can then observe that exactly one of the $(N-1)$ runs boosts p to one if and only if $|\vec{x}| \geq 2$. Consequently, we obtain $r_1, \ldots, r_n$ in which there are both 0 and 1 at least for $t = |\vec{x}|$ if $|\vec{x}| \geq 2$, whereas the probability of obtaining such r_i's is zero for every t if $|\vec{x}| \leq 1$.

4.2 Terminology

Suppose that each party i has a bit x_i (i.e., the n parties share an n-bit string $\vec{x} = (x_1, x_2, \ldots, x_n)$). For an index set $S \subseteq [n]$, a string $\vec{x}$ is said to be *consistent* over S if x_i is equal to x_j for all i, j in S. Otherwise $\vec{x}$ is said to be *inconsistent* over S. If S is the empty set, any $\vec{x}$ is said to be consistent over S. We also say that an n-qubit pure state $|\psi\rangle = \sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} |\vec{x}\rangle = \sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} |x_1\rangle \otimes \cdots \otimes |x_n\rangle$ shared by the n parties is *consistent (inconsistent)* over S if $\alpha_{\vec{x}} \neq 0$ only for $\vec{x}$'s that are consistent (inconsistent) over S (there are pure states that are neither consistent nor inconsistent over S, but we do not need to define such states). Note that the index set S is used solely to provide a simple definition of "consistent/inconsistent" (recall that no party has an index in the anonymous setting). The set S is actually defined as follows: every party has a variable $z \in \{\text{“marked”}, \text{“unmarked”}\}$, and S is defined as the set of all the parties with $z = \text{“marked”}$.

We further define the *consistency function* $C_S \colon \{0,1\}^n \to \{\text{“consistent”}, \text{“inconsistent”}\}$, which decides if a given string $\vec{x} \in \{0,1\}^n$ distributed over n parties is consistent over S. Namely, $C_S(\vec{x})$ returns "consistent" if $\vec{x}$ is consistent over S and "inconsistent" otherwise. In particular, AV_n is equivalent to computing $C_{[n]}$.

4.3 The H_1-Algorithm

As in Section 3, we fix an H_0-algorithm and a C_S-algorithm, which we use in a black-box manner. At the beginning of the algorithm, every party i prepares two single-qubit registers X_i and Y_i. Let $\mathsf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ and $\mathsf{Y} = (\mathsf{Y}_1, \ldots, \mathsf{Y}_n)$. We will describe an H_1-algorithm that exactly computes the function H_1 over the contents of X and sets the content of each Y_i to the function value. We assume that each register Y_i is initialized to $|\text{“true”}\rangle$ for an orthonormal basis $\{|\text{“true”}\rangle, |\text{“false”}\rangle\}$ of $\mathbb{C}^2$. We basically follow the idea in Section 4.1 to reduce computing H_1 to computing the binary-valued functions H_0 and C_S. Since the idea is essentially to compute a three-valued function (i.e., it differentiates between three cases: $|\vec{x}| = 0$, $|\vec{x}| = 1$, and $|\vec{x}| \geq 2$), we cast the idea into two yes-no tests. Namely, the algorithm first tests if $|\vec{x}|$ is 0 or not. If $|\vec{x}| = 0$, then it concludes that $H_1(\vec{x}) = \text{“false”}$. The algorithm also performs another test to decide if $|\vec{x}| \leq 1$ or $|\vec{x}| \geq 2$, which, together with the first test, determines $H_1(\vec{x})$. The second test is needed only when the first test concludes $|\vec{x}| \neq 0$. Just for a simple description of the algorithm, however, we will always run both tests and compute the value of H_1 from their results.

4.3.1 First Test

To test if $|\vec{x}| = 0$, each party i prepares a single-qubit register $\mathsf{O}_i^{(0)}$, the content of which is initialized to $|\text{“true”}\rangle$. Each party then performs the H_0-algorithm to exactly compute the value of H_0 over the contents of X, and stores the computed value in each $\mathsf{O}_i^{(0)}$. From the definition of the H_0-algorithm, this transforms

the state in X and $\mathsf{O}^{(0)} = (\mathsf{O}_1^{(0)}, \ldots, \mathsf{O}_n^{(0)})$ as follows:

$$\bigotimes_{i=1}^{n} \left(|x_i\rangle_{\mathsf{X}_i} |\text{“true”}\rangle_{\mathsf{Y}_i} |\text{“true”}\rangle_{\mathsf{O}_i^{(0)}} \right) \mapsto \bigotimes_{i=1}^{n} \left(|x_i\rangle_{\mathsf{X}_i} |\text{“true”}\rangle_{\mathsf{Y}_i} |H_0(\vec{x})\rangle_{\mathsf{O}_i^{(0)}} \right).$$

By rearranging registers, we have $|\vec{x}\rangle_{\mathsf{X}} (|\text{“true”}\rangle^{\otimes n})_{\mathsf{Y}} (|H_0(\vec{x})\rangle^{\otimes n})_{\mathsf{O}^{(0)}}$, where $|\vec{x}\rangle = |x_1, \ldots, x_n\rangle$. If the content of $\mathsf{O}_i^{(0)}$ is “true”, then the content of Y_i will be set to “false” later (because this means $|\vec{x}| = 0$).

4.3.2 Second Test

Next each party tests whether $|\vec{x}| \leq 1$ or $|\vec{x}| \geq 2$ with certainty. Recall the probabilistic algorithm described in Section 4.1, in which every party i sets a classical variable r_i to 0 or 1 each with probability $1/2$ if $x_i = 1$ and sets the variable r_i to “$*$” if $x_i = 0$, and then sends copies of r_i to all parties. Our goal is to amplify the probability p that both 0 and 1 are included in $r_1, \ldots, r_n$ by somehow using the exact amplitude amplification. The difficulty is that no party knows the value of p_{init} $(= 1 - 2/2^{|\vec{x}|})$. The test thus uses a *guess* t at $|\vec{x}|$ and tries to amplify p assuming that $p_{\text{init}} = 1 - 2/2^t$. If t is indeed equal to $|\vec{x}|$, then the procedure outputs the correct answer with probability one by Theorem 2.1. By running the test for every $t = 2, \ldots, N$ *in parallel*, we can decide if $|\vec{x}| \leq 1$ or $|\vec{x}| \geq 2$ without error, since the probability of obtaining $r_1, \ldots, r_n$ in which both 0 and 1 are included is zero for every t if $|\vec{x}| \leq 1$.

We now describe the test procedure for each t. Assume that each party i has an additional single-qubit register $\mathsf{Z}_i^{(t)}$ initialized to $|\text{“unmarked ”}\rangle$. Let $\mathsf{Z}^{(t)} = (\mathsf{Z}_1^{(t)}, \ldots, \mathsf{Z}_n^{(t)})$. The state is then

$$\sum_{\vec{x} \in \{0,1\}^n} \alpha_{\vec{x}} \bigotimes_{i=1}^{n} \left(|x_i\rangle_{\mathsf{X}_i} |\text{“unmarked ”}\rangle_{\mathsf{Z}_i^{(t)}} \right),$$

where registers Y and $\mathsf{O}^{(0)}$ are omitted to avoid complication. If the content of X_i is 1, the party i flips the content of $\mathsf{Z}_i^{(t)}$ to “marked”, where $\{|\text{“marked ”}\rangle, |\text{“unmarked ”}\rangle\}$ is an orthonormal basis in $\mathbb{C}^2$.[5]

The state is thus a linear combination of $\bigotimes_{i=1}^{n} \left(|x_i\rangle_{\mathsf{X}_i} |z(x_i)\rangle_{\mathsf{Z}_i^{(t)}} \right)$ over all $\vec{x}$, where $z\colon \{0,1\} \to \{\text{“marked ”}, \text{“unmarked ”}\}$ is a function such that $z(1) = \text{“marked ”}$ and $z(0) = \text{“unmarked ”}$. This tensored state for each $\vec{x}$ is, after rearranging the registers, equivalent to

$$\bigotimes_{i \in S} \left(|1\rangle_{\mathsf{X}_i} |\text{“marked ”}\rangle_{\mathsf{Z}_i^{(t)}} \right) \otimes \bigotimes_{i \in [n] \setminus S} \left(|0\rangle_{\mathsf{X}_i} |\text{“unmarked ”}\rangle_{\mathsf{Z}_i^{(t)}} \right),$$

where S is the set of $i \in [n]$ such that $\mathsf{Z}_i^{(t)}$ is in the state $|\text{“marked ”}\rangle$. Note that the content of $\mathsf{Z}_i^{(t)}$ does not depend on t. We prepared $\mathsf{Z}_i^{(t)}$ for each t just because every party i needs to access $\mathsf{Z}_i^{(t)}$ for $t = 2, \ldots, N$ in parallel.

The base algorithm A (to be amplified) is described as follows. Suppose that every party i has a single-qubit register $\mathsf{R}_i^{(t)}$ initialized to $|0\rangle$, and let $\mathsf{R}^{(t)} = (\mathsf{R}_1^{(t)}, \ldots, \mathsf{R}_n^{(t)})$. If the content of $\mathsf{Z}_i^{(t)}$ is“marked”, the party i applies the Hadamard operator $H = \frac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix} \right)$ to the qubit in $\mathsf{R}_i^{(t)}$ to create $(|0\rangle + |1\rangle)/\sqrt{2}$

[5] We use $\{|\text{“marked ”}\rangle, |\text{“unmarked ”}\rangle\}$, instead of $\{|0\rangle, |1\rangle\}$, as a basis of the linear space associated with $\mathsf{Z}_i^{(t)}$ simply for ease of understanding. It is not harmful to assume that $\{|\text{“marked ”}\rangle, |\text{“unmarked ”}\rangle\}$ is $\{|0\rangle, |1\rangle\}$.

(one can see that the register $\mathsf{R}_i^{(t)}$ of each party i is the quantum equivalent of r_i used in the probabilistic algorithm[6]). The state is now represented as, for any $\vec{x}$,

$$\bigotimes_{i\in S}\left(|1\rangle_{\mathsf{X}_i}|\text{“marked ”}\rangle_{\mathsf{Z}_i^{(t)}}\frac{|0\rangle_{\mathsf{R}_i^{(t)}}+|1\rangle_{\mathsf{R}_i^{(t)}}}{\sqrt{2}}\right)\otimes\bigotimes_{i\in[n]\setminus S}\left(|0\rangle_{\mathsf{X}_i}|\text{“unmarked ”}\rangle_{\mathsf{Z}_i^{(t)}}|0\rangle_{\mathsf{R}_i^{(t)}}\right).$$

By rearranging registers, we have

$$|\vec{x}\rangle_{\mathsf{X}}|\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}}\bigotimes_{i\in S}\left(\frac{|0\rangle_{\mathsf{R}_i^{(t)}}+|1\rangle_{\mathsf{R}_i^{(t)}}}{\sqrt{2}}\right)\bigotimes_{i\in[n]\setminus S}|0\rangle_{\mathsf{R}_i^{(t)}}=|\vec{x}\rangle_{\mathsf{X}}|\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}}|\psi(\vec{x})\rangle_{\mathsf{R}^{(t)}},\tag{4.1}$$

where $|\vec{z}(\vec{x})\rangle=\bigotimes_{i=1}^{n}|z(x_i)\rangle$, (i.e., the n-tensor product of $|\text{“marked ”}\rangle$ or $|\text{“unmarked ”}\rangle$ corresponding to $\vec{x}$) and

$$|\psi(\vec{x})\rangle=\left(\frac{1}{\sqrt{2^{|S|}}}\sum_{\vec{y}\in\{0,1\}^{|S|}}|\vec{y}\rangle\right)|0\rangle^{\otimes(n-|S|)}.$$

This is the end of the base algorithm A.

We then boost the amplitudes of the basis states superposed in $|\psi_t(\vec{x})\rangle$ that are inconsistent over S (i.e., the amplitudes of the basis states such that there are both $|0\rangle$ and $|1\rangle$ in R_t's of the parties in S). For this, we identify the register of the index part $|z\rangle$ in Theorem 2.1 as $\mathsf{R}^{(t)}$ (the register of the remaining part $|\psi_z\rangle$ has dimension 0 in our case). We then define the Boolean function χ in Theorem 2.1 as follows: $\chi\colon\{0,1\}^n\to\{\mathsf{true},\mathsf{false}\}$ is a Boolean function over the content (i.e., n-bit strings) of $\mathsf{R}^{(t)}$ such that χ is true if and only if the consistency function C_S is $\mathsf{inconsistent}$. We also define the success probability a as $a(t)=1-\frac{2}{2^t}$. For convenience, we express $|\psi(\vec{x})\rangle$ as

$$|\psi(\vec{x})\rangle=|\psi_{\mathsf{inconsistent}}\rangle+|\psi_{\mathsf{consistent}}\rangle,$$

where $|\psi_{\mathsf{inconsistent}}\rangle$ and $|\psi_{\mathsf{consistent}}\rangle$ are unnormalized states defined as follows:

$$|\psi_{\mathsf{inconsistent}}\rangle=\left(\frac{1}{\sqrt{2^{|S|}}}\sum_{\vec{y}\in\{0,1\}^{|S|}:|\vec{y}|\neq 0,|S|}|\vec{y}\rangle\right)|0\rangle^{\otimes(n-|S|)},$$
$$|\psi_{\mathsf{consistent}}\rangle=\frac{1}{\sqrt{2^{|S|}}}\left(|0\rangle^{\otimes|S|}+|1\rangle^{\otimes|S|}\right)|0\rangle^{\otimes(n-|S|)}.$$

To realize $F_\chi(\theta_{a(t)})$, every party i prepares a single-qubit register $\mathsf{S}_i^{(t)}$ initialized to $|\text{“consistent”}\rangle$ and then performs the following operations (the corresponding formulae will be given after describing the operations):

[6]The content of $\mathsf{R}_i^{(t)}$ of each “unmarked” party is set to $|0\rangle$ here, while the equivalent r_i in the base classical algorithm is set to “$*$” (instead of 0). Actually, the symbol “$*$” is used to distinguish between $|\vec{x}|=0$ and $|\vec{x}|=1$. However we do not need it any longer thanks to the first test.

(1) Perform a C_S-algorithm with $(\mathsf{R}_i^{(t)}, \mathsf{S}_i^{(t)}, \mathsf{Z}_i^{(t)}, N)$, which computes C_S for each basis state $|\vec{y}\rangle|0\rangle^{n-|S|}$ of $|\psi(\vec{x})\rangle_{\mathsf{R}^{(t)}}$ and sets the content of $\mathsf{S}_i^{(t)}$ to the value of C_S, where S is the set of $i \in [n]$ such that the content of $\mathsf{Z}_i^{(t)}$ is "marked";

(2) Multiply the amplitude of each basis state of $\mathsf{R}_i^{(t)}$ by a factor of $\exp\left(\mathrm{i}\frac{\theta_{a(t)}}{t}\right)$ if the content of $\mathsf{S}_i^{(t)}$ is "inconsistent" and the content of $\mathsf{Z}_i^{(t)}$ is "marked" [note that no party makes measurement here, but every party performs a phase shift controlled by the qubits in $(\mathsf{S}_i^{(t)}, \mathsf{Z}_i^{(t)})$];

(3) Invert every computation and communication of (1) to disentangle all $\mathsf{S}^{(t)} = (\mathsf{S}_1^{(t)}, \ldots, \mathsf{S}^{(n)})$.

The state evolves with the above operations as follows: For any $\vec{x}$,

$$
\begin{aligned}
&|\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\psi(\vec{x})\rangle_{\mathsf{R}^{(t)}} (|\text{“consistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}} \\
&\overset{(1)}{\mapsto} |\vec{z}_t(\vec{x})\rangle_{\mathsf{Z}^{(t)}} \left(|\psi_{\mathsf{inconsistent}}\rangle_{\mathsf{R}^{(t)}} (|\text{“inconsistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}} + |\psi_{\mathsf{consistent}}\rangle_{\mathsf{R}^{(t)}} (|\text{“consistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}}\right) \\
&\overset{(2)}{\mapsto} |\vec{z}_t(\vec{x})\rangle_{\mathsf{Z}^{(t)}} \left(\left(e^{\mathrm{i}\frac{\theta_{a(t)}}{t}}\right)^{|S|} |\psi_{\mathsf{inconsistent}}\rangle_{\mathsf{R}^{(t)}} (|\text{“inconsistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}} + |\psi_{\mathsf{consistent}}\rangle_{\mathsf{R}^{(t)}} (|\text{“consistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}}\right) \\
&\overset{(3)}{\mapsto} |\vec{z}_t(\vec{x})\rangle_{\mathsf{Z}^{(t)}} \left(\left(e^{\mathrm{i}\theta_{a(t)}\frac{|S|}{t}} |\psi_{\mathsf{inconsistent}}\rangle_{\mathsf{R}^{(t)}} + |\psi_{\mathsf{consistent}}\rangle_{\mathsf{R}^{(t)}}\right) (|\text{“consistent”}\rangle^{\otimes n})_{\mathsf{S}^{(t)}}\right).
\end{aligned}
$$

We have now finished the first operation, $F_\chi(\theta_{a(t)})$, of $-AF_0(\phi_{a(t)})A^\dagger F_\chi(\theta_{a(t)})$. Hereafter, we omit $\mathsf{S}^{(t)}$ since it has been disentangled and will not be used.

Next, $A^\dagger$ is performed. The operation $F_0(\phi_{a(t)})$ is then realized with the H_0-algorithm instead of the C_S-algorithm in a way similar to the operation $F_\chi(\theta_{a(t)})$. Finally, perform the operation A again. This is the end of the amplitude amplification. In summary, the state over $\mathsf{Z}^{(t)}$ and $\mathsf{R}^{(t)}$ in formula (4.1) is transformed as follows: For any $\vec{x}$,

$$|\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\psi(\vec{x})\rangle_{\mathsf{R}^{(t)}} \mapsto |\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\psi'_t(\vec{x})\rangle_{\mathsf{R}^{(t)}},$$

where $|\psi'_t(\vec{x})\rangle$ is given in the following claim.

Claim 4.1. *For each $\vec{x} \in \{0,1\}^n$, it holds that*

$$|\psi'_t(\vec{x})\rangle = \begin{cases} \sqrt{\frac{2^{|S|}}{2^{|S|}-2}}\, |\psi_{\mathsf{inconsistent}}\rangle & (|S| \geq 2 \text{ and } t = |S|), \\ \sum_{\vec{y}\in\{0,1\}^{|S|}} \beta_{\vec{y}}^{(t)} |\vec{y}\rangle|0\rangle^{\otimes(n-|S|)} & (|S| \geq 2 \text{ and } t \neq |S|), \\ -e^{\mathrm{i}\phi_{a(t)}\frac{|S|}{t}} |\psi_{\mathsf{consistent}}\rangle & (|S| \leq 1 \text{ and all } t), \end{cases}$$

where $t \in \{2, \ldots, N\}$ and $\beta_{\vec{y}}^{(t)}$ is a certain complex number for each t and $\vec{y} \in \{0,1\}^{|S|}$ such that $\sum_{y\in\{0,1\}^{|S|}} |\beta_{\vec{y}}^{(t)}|^2 = 1$ for each t.

Proof of Claim 4.1. If $|S| \geq 2$ and $t = |S|$, the claim follows from Theorem 2.1. If $|S| \geq 2$ and $t \neq |S|$, the claim is obvious. If $|S| \leq 1$, then $|\psi(\vec{x})\rangle = |\psi_{\mathsf{consistent}}\rangle$ and thus $F_\chi(\theta_{a(t)})$ does nothing. We hence have $|\psi'_t(\vec{x})\rangle_{\mathsf{R}^{(t)}} = -AF_0(\phi_{a(t)})A^\dagger |\psi_{\mathsf{consistent}}\rangle_{\mathsf{R}^{(t)}} = -AF_0(\phi_{a(t)})|0^n\rangle_{\mathsf{R}^{(t)}} = -e^{\mathrm{i}\phi_{a(t)}\frac{|S|}{t}} |\psi_{\mathsf{consistent}}\rangle_{\mathsf{R}^{(t)}}$. □

Note that the parties do not know what state they share after the amplitude amplification. To know it, each party i prepares a new quantum register $\mathsf{O}_i^{(t)}$, initialized to $|\text{“consistent”}\rangle$, and again performs the C_S-algorithm with $(\mathsf{R}_i^{(t)}, \mathsf{O}_i^{(t)}, \mathsf{Z}_i^{(t)}, N)$, which transforms the state as follows:

$$|\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\psi_t'(\vec{x})\rangle_{\mathsf{R}^{(t)}} (|\text{“consistent”}\rangle^{\otimes n})_{\mathsf{O}^{(t)}} \mapsto |\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\Psi_t(\vec{x})\rangle_{(\mathsf{R}^{(t)},\mathsf{O}^{(t)})},$$

where $\mathsf{O}^{(t)} = (\mathsf{O}_1^{(t)}, \ldots, \mathsf{O}_n^{(t)})$ and

$$|\Psi_t(\vec{x})\rangle = \begin{cases} \sqrt{\frac{2^{|S|}}{2^{|S|}-2}}\, |\psi_{\mathsf{inconsistent}}\rangle |\text{“inconsistent”}\rangle^{\otimes n} & (|S| \geq 2 \text{ and } t = |S|), \\ \sum_{\vec{y}\in\{0,1\}^{|S|}} \beta_{\vec{y}}^{(t)} |\vec{y}\rangle |0\rangle^{\otimes(n-|S|)} |C_S(\vec{y}\, 0^{(n-|S|)})\rangle^{\otimes n} & (|S| \geq 2 \text{ and } t \neq |S|), \\ -e^{\mathrm{i}\phi_{a(t)}\frac{|S|}{t}} |\psi_{\mathsf{consistent}}\rangle |\text{“consistent”}\rangle^{\otimes n} & (|S| \leq 1 \text{ and all } t). \end{cases}$$

The content of $\mathsf{O}_i^{(t)}$ is the result of the second test.

4.3.3 Final Evaluation

After the first and the second tests for $t = 2, \ldots, N$, the entire state becomes

$$|\vec{x}\rangle_{\mathsf{X}} \otimes (|\mathsf{true}\rangle^{\otimes n})_{\mathsf{Y}} \otimes (|H_0(\vec{x})\rangle^{\otimes n})_{\mathsf{O}^{(0)}} \otimes \left(\bigotimes_{t=2}^{N} |\vec{z}(\vec{x})\rangle_{\mathsf{Z}^{(t)}} |\Psi_t(\vec{x})\rangle_{(\mathsf{R}^{(t)},\mathsf{O}^{(t)})} \right),$$

for each $\vec{x}$.

Recall that every party i has registers Y_i, $\mathsf{O}_i^{(0)}$, $\mathsf{O}_i^{(t)}$ for $t = 2, \ldots, N$. In the final step of our algorithm for computing H_1, every party i concludes the value of $H_1(\vec{x})$ from the contents of $\mathsf{O}_i^{(0)}$ and $\mathsf{O}_i^{(t)}$ as follows:

> If either the content of $\mathsf{O}_i^{(0)}$ is “true” or the content of $\mathsf{O}_i^{(t)}$ is “inconsistent” for some $t \in \{2, \ldots, N\}$, then set the content of Y_i to “false”.

It is not difficult to show the correctness. If the content of $\mathsf{O}_i^{(0)}$ is “true”, then the value of $H_1(\vec{x})$ is obviously “false” (because it means $|\vec{x}| = 0$). Suppose that the content of $\mathsf{O}_i^{(0)}$ is “false” (i.e., $|\vec{x}| := |S| \neq 0$). From the definition of $|\Psi_t(\vec{x})\rangle$, we can observe the following facts: (1) If $|\vec{x}| := |S| = 1$, then the contents of $\mathsf{O}_i^{(t)}$ is “consistent” for all $t = 2, \ldots, N$; (2) If $|\vec{x}| := |S| \geq 2$, then the content of $\mathsf{O}_i^{(t)}$ is “inconsistent” at least for $t = |\vec{x}|(= |S|)$. The content of Y_i is hence set to $H_1(\vec{x})$ with certainty. By inverting every computation and communication of the first and second tests, the registers $\mathsf{O}^{(0)}$, $\mathsf{Z}^{(t)}$, $\mathsf{R}^{(t)}$, and $\mathsf{O}^{(t)}$ are all made unentangled. Then we have the entire state $|\vec{x}\rangle_{\mathsf{X}} \otimes (|H_1(\vec{x})\rangle^{\otimes n})_{\mathsf{Y}}$. For a superposed input, our algorithm thus transforms

$$\sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} \bigotimes_{i=1}^{n} |x_i\rangle_{\mathsf{X}_i} |\mathsf{true}\rangle_{\mathsf{Y}_i} \mapsto \sum_{\vec{x}\in\{0,1\}^n} \alpha_{\vec{x}} \bigotimes_{i=1}^{n} |x_i\rangle_{\mathsf{X}_i} |H_1(\vec{x})\rangle_{\mathsf{Y}_i}.$$

A full description of our H_1-algorithm is given in Figure 2. We summarize the algorithm as a lemma.

Input: Single-qubit registers X_i and Y_i (Y_i is initialized to $|\text{“}\mathsf{true}\text{”}\rangle$), an integer N.

Output: Single-qubit registers X_i and Y_i.

Notation: $\mathsf{X} := (\mathsf{X}_1,\ldots,\mathsf{X}_n)$, $\mathsf{Y} := (\mathsf{Y}_1,\ldots,\mathsf{Y}_n)$, $\mathsf{R}^{(t)} := (\mathsf{R}_1^{(t)},\ldots,\mathsf{R}_n^{(t)})$,
$\mathsf{Z}^{(t)} := (\mathsf{Z}_1^{(t)},\ldots,\mathsf{Z}_n^{(t)})$, $\mathsf{S}^{(t)} := (\mathsf{S}_1^{(t)},\ldots,\mathsf{S}_n^{(t)})$, and $\mathsf{S}'^{(t)} := (\mathsf{S}_1'^{(t)},\ldots,\mathsf{S}_n'^{(t)})$.
$S := \{i \in [n] : \text{the content of } \mathsf{Z}_i^{(t)} \text{ is “}\mathsf{marked}\text{”}\}$.

1. Prepare single-qubit registers $\mathsf{R}_i^{(t)}$, $\mathsf{S}_i^{(t)}$, $\mathsf{S}_i'^{(t)}$, $\mathsf{O}_i^{(t)}$ for $t \in \{2,\ldots,N\}$, and $\mathsf{O}_i^{(0)}$, and initialize the state of $\mathsf{R}_i^{(t)}$ to $|0\rangle$ for each $t = 2,\ldots,N$.
2. [**First Test**]: Run an H_0-algorithm with $(\mathsf{X}_i, \mathsf{O}_i^{(0)}, N)$ (to compute H_0 over the contents of X and store the result in $\mathsf{O}_i^{(0)}$ for each $i \in [n]$).
3. [**Second Test**]: Perform the following steps for $t \in \{2,\ldots,N\}$ in parallel, where $a(t) = 1 - \frac{2}{2^t}$ and $\mathbb{I} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$.
   - 3.1 Set the state of $\mathsf{Z}_i^{(t)}$ to $|\text{“}\mathsf{marked}\text{”}\rangle$ if the content of X_i is 1, and $|\text{“}\mathsf{unmarked}\text{”}\rangle$ otherwise.
   - 3.2 [A]: If the content of $\mathsf{Z}_i^{(t)}$ is “marked”, apply $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ to the qubit in $\mathsf{R}_i^{(t)}$.
   - 3.3 [$F_\chi(\psi_{a(t)})$]: Initialize the state of $\mathsf{S}_i^{(t)}$ to $|\text{“}\mathsf{consistent}\text{”}\rangle$.
     - 3.3.1 Run a C_S-algorithm with $(\mathsf{R}_i^{(t)}, \mathsf{S}_i^{(t)}, \mathsf{Z}_i^{(t)}, N)$ (to compute C_S over the contents of $\mathsf{R}^{(t)}$ and store the result in $\mathsf{S}_i^{(t)}$ for each $i \in [n]$).
     - 3.3.2 Multiply the phase by $\exp(\mathrm{i}\frac{1}{t}\psi_{a(t)})$ if the content of $\mathsf{S}_i^{(t)}$ is “$\mathsf{inconsistent}$” and the content of $\mathsf{Z}_i^{(t)}$ is “marked”.
     - 3.3.3 Invert every computation and communication of Step 3.3.1 to disentangle $\mathsf{S}^{(t)}$.
   - 3.4 [$A^\dagger$]: Invert the computation in Step 3.2.
   - 3.5 [$F_0(\phi_{a(t)})$]: Initialize the state of $\mathsf{S}_i'^{(t)}$ to $|\text{“}\mathsf{true}\text{”}\rangle$.
     - 3.5.1 Run the H_0-algorithm with $(\mathsf{R}_i^{(t)}, \mathsf{S}_i'^{(t)}, N)$ (to compute H_0 over the content of $\mathsf{R}^{(t)}$ and store the result in $\mathsf{S}_i'^{(t)}$ for each $i \in [n]$).
     - 3.5.2 Multiply the phase by $\exp(\mathrm{i}\frac{1}{t}\phi_{a(t)})$ if the content of $\mathsf{S}_i'^{(t)}$ is “true” and the content of $\mathsf{Z}_i^{(t)}$ is “marked”.
     - 3.5.3 Invert every computation and communication of Step 3.5.1 to disentangle $\mathsf{S}'^{(t)}$.
   - 3.6 [A]: Perform the same operation as in Step 3.2.
   - 3.7 Initialize the state of $\mathsf{O}_i^{(t)}$ to $|\text{“}\mathsf{consistent}\text{”}\rangle$, and perform a C_S-algorithm with $(\mathsf{R}_i^{(t)}, \mathsf{O}_i^{(t)}, \mathsf{Z}_i^{(t)})$ and N (to compute C_S over the content of $\mathsf{R}^{(t)}$, and store the result in $\mathsf{O}_i^{(t)}$ for each $i \in [n]$).
4. [**Final Evaluation**]: Flip Y (i.e., transform the state $|\text{“}\mathsf{true}\text{”}\rangle$ of Y into $|\text{“}\mathsf{false}\text{”}\rangle$), if either the content of $\mathsf{O}_i^{(0)}$ is “true” or the content of $\mathsf{O}_i^{(t)}$ is “$\mathsf{inconsistent}$” for some $t \in \{2,\ldots,N\}$.
5. Invert every computation and communication of Steps 2 and 3 to disentangle all registers except X and Y.
6. Output X_i and Y_i, and then halt.

Figure 2: H_1-Algorithm (the operations performed by each party $i \in [n]$).

Lemma 4.2. *For any graph $G \in \mathcal{G}_n$, if every party knows an upper bound N of the number n of parties, there exists an H_1-algorithm that runs in $O(Q_G^{\mathrm{rnd}}(\mathcal{H}_0) + Q_G^{\mathrm{rnd}}(\mathcal{C}_S))$ rounds with bit complexity $O(N(Q_G^{\mathrm{bit}}(\mathcal{H}_0) + Q_G^{\mathrm{bit}}(\mathcal{C}_S)))$, where $\mathcal{H}_0$ and $\mathcal{C}_S$ are any H_0-algorithm and any C_S-algorithm that work for a given upper bound of n, respectively. Moreover, all the quantum communication is performed solely to invoke $\mathcal{H}_0$ and $\mathcal{C}_S$, and their inversions.*

Proof. The correctness follows from the above description of the algorithm. For the complexity, all communications are performed to compute H_0, C_S, and their inversions. Namely, the first test runs the H_0-algorithm and its inversion $O(1)$ times and the second test runs the H_0-algorithm, the C_S-algorithm and their inversions $O(1)$ times for $t = 2, \ldots, N$ in parallel. The lemma thus follows. □

4.4 Computing C_S with Any H_0-Algorithm

Finally, we show that computing C_S is reducible to computing H_0.

Lemma 4.3. *For any graph $G \in \mathcal{G}_n$, there exists a C_S-algorithm that runs in $O(Q_G^{\mathrm{rnd}}(\mathcal{H}_0))$ rounds with bit complexity $O(Q_G^{\mathrm{bit}}(\mathcal{H}_0))$, where $\mathcal{H}_0$ is any H_0-algorithm. Moreover, all the quantum communication is performed solely to invoke $\mathcal{H}_0$ and its inversion.*

Proof. Function C_S can be computed by first computing H_0 and $H_{|S|}$ in parallel over the input bits of the parties in S, and then computing their OR (and outputting "consistent" if and only if the OR is "true"). To compute H_0 over the $|S|$ bits with any H_0-algorithm over n bits, every party not in S sets its input to 0, and all parties then run the H_0-algorithm. The function $H_{|S|}$ over the $|S|$ bits can be computed in the same way as with H_0 over the $|S|$ bits except that every party in S negates its input before computation. All work space can be made unentangled by inverting all operations after copying the value of C_S to another register. □

Lemmas 4.2 and 4.3 imply that, for any graph $G \in \mathcal{G}_n$, there is an H_1-algorithm that runs in $O(Q_G^{\mathrm{rnd}}(\mathcal{H}_0))$ rounds with bit complexity $O(N \cdot Q_G^{\mathrm{bit}}(\mathcal{H}_0))$. This completes the proof of Theorem 1.2.

5 Applications

5.1 The Algorithms for SV_n and LE_n

As an application of Corollary 1.3, we present a quantum algorithm that exactly solves SV_n and LE_n, which runs with smaller round complexity than the existing algorithms while keeping the best bit complexity.

Proofs of Corollaries 1.4 and 1.5. We first give a simple H_0-algorithm to apply Theorem 1.2 and Corollary 1.3. The algorithm is a straightforward quantization of the following deterministic algorithm: Every party sends a copy of its input bit to each adjacent party and keeps a copy of the bit for itself. Every party then computes the OR of all the bits it received and the bit it kept, and sends a copy of the resulting bit to each adjacent party while also retaining a copy. By repeating this procedure Δ times for an upper bound Δ of the network diameter, every party can know the OR of all bits and thus the value of H_0. This classical algorithm can easily be converted to the quantum equivalent with the same complexity (up to a constant factor). Thus, we have proved the following claim.

Claim 5.1. *Let G be any graph in $\mathcal{G}_n$, and let m be the number of edges in G. Then, there exists an H_0-algorithm $\mathcal{H}_0$ that runs in $O(\Delta)$ rounds with bit complexity $O(\Delta m)$ on an anonymous quantum network of the underlying graph G (i.e., $Q_G^{\mathrm{rnd}}(\mathcal{H}_0) \in O(\Delta)$ and $Q_G^{\mathrm{bit}}(\mathcal{H}_0) \in O(\Delta m)$) if an upper bound Δ of the diameter of G is given to each party.*

Corollary 1.4 follows from Theorem 1.2 and Claim 5.1 with the trivial upper bound $n-1$ (or $N-1$) for Δ. We then obtain Corollary 1.5 from Corollary 1.3 and Claim 5.1. □

5.2 Computing Boolean Functions

Recall that once a unique leader is elected, a spanning tree can be constructed by starting at the leader and traversing the underlying graph and then the leader can assign a unique identifier to every party by traversing the tree (mentioned in Section 2.2). Moreover, if a unique leader exists, the underlying graph can be recognized as shown in the following Lemma 5.2. In the lemma, we say that a graph can be *recognized* if every party can compute the adjacency matrix of the graph up to isomorphism. Although it is almost obvious that the number of required rounds is $O(n)$, we provide a proof in Appendix B to justify the bit complexity.

Lemma 5.2. *Once a unique leader is elected on an anonymous quantum network with any undirected underlying graph, the graph can exactly be recognized in $O(n)$ rounds with $O(n^3)$ bit complexity.*

Hence it is possible to compute a wider class of Boolean functions than symmetric functions, i.e., all Boolean functions that may depend on the graph topology (but are independent of the way of assigning unique identifiers to parties) as well as the input bit given to each party. For instance, consider the Boolean function $f_\triangle$ over n distributed bits such that $f_\triangle$ is true if and only if there exist three parties (i,j,k) with $x_i = x_j = x_k = 1$ that form a triangle on the underlying graph. This function $f_\triangle$ can be solved on quantum anonymous networks as follows: After electing a unique leader, every party sends a matrix together with its input bit to its parent node when constructing the adjacent matrix in the algorithm in Lemma 5.2. On the other hand, $f_\triangle$ cannot be solved on classical anonymous networks: for instance no party can distinguish between a 3-cycle and a 4-cycle as the underlying graph, when one of them is the case. This example can easily be generalized: Let $G_{\vec{x}}$ be the underlying graph with each vertex i labeled by x_i. Let $\mathcal{M}$ be a collection of graphs with a certain property of at most N vertices labeled by 0 or 1. Then define $f_{\mathcal{M}}$ as follows: $f_{\mathcal{M}}$ is true if and only if $G_{\vec{x}}$ is isomorphic to one of the elements in $\mathcal{M}$. In the case of $f_\triangle$, $\mathcal{M}$ is the set of all graphs with at most N vertices that has a triangle with the three vertices labeled by 1. In the case of OR, $\mathcal{M}$ is the set of all graphs with at most N vertices including at least one vertex labeled by 1.

Corollary 5.3. *Let $f_{\mathcal{M}}$ be a Boolean function over n distributed input bits defined as above. Then, there exists a quantum algorithm that exactly computes $f_{\mathcal{M}}$ in $O(N)$ rounds with bit complexity $O(mN^3)$ on an anonymous quantum network with its underlying graph unknown, if an upper bound N of the number of parties is given to every party, where m is the number of edges of G. Furthermore, with the promise that the exact number n of parties is given instead of N, $f_{\mathcal{M}}$ can be solved in $O(n)$ rounds with bit complexity $O(mn^2)$.*

Proof. Lemma 5.2 can be modified so that the leader can gather all the input bits associated with unique identifiers along the spanning tree when constructing the adjacent matrix. From this and Corollary 1.5, the statement holds. □

The problem of computing classical functions can be naturally generalized to that of sharing quantum states. Let $\mathrm{L}(\mathcal{X})$ be the linear space of all linear operators acting on a complex Euclidean space $\mathcal{X}$, and let $\mathrm{D}(\mathcal{X})$ be the linear space of all density operators (all positive-semidefinite oprators with trace one) on $\mathcal{X}$. We say that an admissible super-operator (i.e., trace-preserving completely-positive super-operator) $\Phi\colon \mathrm{L}(\mathbb{C}^{2^n}) \to \mathrm{L}(\mathbb{C}^{2^m})$ acting on n qubits is *invariant under the automorphisms of* G if for any unitary operator $P_G \in \mathrm{L}(\mathbb{C}^{2^n})$ that acts as an automorphism π of G over n qubits, it holds that $\Phi(P_G X {P_G}^\dagger) = \Phi(X)$ for every $X \in \mathrm{D}(\mathbb{C}^{2^n})$. Let $\Phi_G\colon \mathrm{L}(\mathbb{C}^{2^n}) \to \mathrm{L}(\mathbb{C}^{2^{cn}})$ be an admissible super-operator invariant under the automorphisms of G that maps an n-qubit state to a cn qubit state for a constant natural number c. Then, an argument similar to the proof of Corollary 5.3 can show that there exists a quantum algorithm that exactly computes $\Phi_G(\rho)$ in $O(N)$ rounds with bit complexity $O(mN^3)$ on an anonymous quantum network with its underlying graph unknown, if the n parties share an n-qubit state $\rho \in \mathrm{D}(\mathbb{C}^{2^n})$ and each of them is given an upper bound N of the number of parties, where m is the number of edges of G.

6 Concluding Remarks

In our reductions, we need to invert some of the computation and communication for two reasons. One is to implement the adjoint $A^\dagger$ of the base algorithm A in the amplitude amplification; the other is to erase the garbage left after computing H_0 and C_S, after using their values as the control of phase-shift operators. The former case is simply local computation, while the latter involves communication. To invert communication, each communication link must be bidirectional, i.e., the underlying graph must be undirected, since the only way to return a message to the sender in an anonymous network with its underlying graph unknown is to send it back in the reverse direction along the path (more strictly, along the sequence of the port numbers uniquely determined for the path) through which the message was transferred. Therefore, our algorithm does not work on anonymous networks with *directed* underlying graphs.

The algorithm for LE_n in [35, Sec. 4] avoids this issue by using a classical subroutine that computes the parity function over distributed input bits, which, however, increases the complexity. It remains an open question as to whether we must compute the parity function to solve LE_n when the underlying graph is directed. Another open question is whether LE_n can exactly be solved in $O(N)$ rounds for a given upper bound N of n, when the underlying graph is directed.

Acknowledgements

We are grateful to several anonymous referees for their helpful comments.

References

[1] KARL R. ABRAHAMSON, ANDREW ADLER, LISA HIGHAM, AND DAVID G. KIRKPATRICK: Probabilistic solitude verification on a ring. In *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC '86)*, pp. 161–173, 1986. 3

[2] KARL R. ABRAHAMSON, ANDREW ADLER, LISA HIGHAM, AND DAVID G. KIRKPATRICK: Tight lower bounds for probabilistic solitude verification on anonymous rings. *Journal of the ACM*, 41(2):277–310, 1994. 2, 3

[3] N. AHARON AND J. SILMAN: Quantum dice rolling: a multi-outcome generalization of quantum coin flipping. *New Journal of Physics*, 12(033027), 2010. 8

[4] ANDRIS AMBAINIS, HARRY M. BUHRMAN, YEVGENIY DODIS, AND HEIN RÖHRIG: Multiparty quantum coin flipping. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pp. 250–259, 2004. 8

[5] DANA ANGLUIN: Local and global properties in networks of processors (extended abstract). In *Proceedings of the 20th Annaul ACM Symposium on Theory of Computing*, pp. 82–93, 1980. 2, 3, 4, 5, 7, 11

[6] HAGIT ATTIYA, MARC SNIR, AND MANFRED K. WARMUTH: Computing on an anonymous ring. *Journal of the ACM*, 35(4):845–875, 1988. 2, 11

[7] MICHAEL BEN-OR AND AVINATAN HASSIDIM: Fast quantum Byzantine agreement. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pp. 481–485, 2005. 8

[8] PAOLO BOLDI, SHELLA SHAMMAH, SEBASTIANO VIGNA, BRUNO CODENOTTI, PETER GEMMELL, AND JANOS SIMON: Symmetry breaking in anonymous networks: Characterizations. In *Proceedings of the Fourth Israel Symposium on Theory of Computing and Systems*, pp. 16–26. IEEE Computer Society, 1996. 2

[9] PAOLO BOLDI AND SEBASTIANO VIGNA: Fibrations of graphs. *Discrete Mathematics*, 243:21–66, 2002. 2, 11

[10] GILLES BRASSARD, ANNE BROADBENT, JOSEPH FITZSIMONS, SÉBASTIEN GAMBS, AND ALAIN TAPP: Anonymous quantum communication. In *Advances in Cryptology - ASIACRYPT 2007, Proceedings of the 13th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4833 of *Lecture Notes in Computer Science*, pp. 460–473. Springer, 2007. 8

[11] GILLES BRASSARD, PETER HØYER, MICHELE MOSCA, AND ALAIN TAPP: Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pp. 53–74. AMS, 2002. 5, 12

[12] HARRY M. BUHRMAN, WILLEM VAN DAM, PETER HØYER, AND ALAIN TAPP: Multiparty quantum communication complexity. *Physical Review A*, 60(4):2737–2741, 1999. 8

[13] DONG PYO CHI AND JINSOO KIM: Quantum database search by a single query. In *Proceedings of the First NASA International Conference Quantum Computing and Quantum Communications*, volume 1509 of *Lecture Notes in Computer Science*, pp. 148–151. Springer, 1998. 5, 12

[14] MATTHIAS CHRISTANDL AND STEPHANIE WEHNER: Quantum anonymous transmissions. In *Advances in Cryptology - ASIACRYPT 2005, Proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3788 of *Lecture Notes in Computer Science*, pp. 217–235. Springer, 2005. 8

[15] CLAUDE CRÉPEAU, DANIEL GOTTESMAN, AND ADAM D. SMITH: Secure multi-party quantum computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 643–652, 2002. 8

[16] SHANTANU DAS, PAOLA FLOCCHINI, AMIYA NAYAK, AND NICOLA SANTORO: Effective elections for anonymous mobile agents. In *Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC '07)*, volume 4288 of *Lecture Notes in Computer Science*, pp. 732–743. Springer, 2006. 2

[17] ELLIE D'HONDT AND PRAKASH PANANGADEN: The computational power of the W and GHZ states. *Quantum Information and Computation*, 6(2):173–183, 2006. 8

[18] KRZYSZTOF DIKS, EVANGELOS KRANAKIS, ADAM MALINOWSKI, AND ANDRZEJ PELC: Anonymous wireless rings. *Theoretical Computer Science*, 145(1-2):95–109, 1995. 2

[19] MAOR GANZ: Quantum leader election. Technical report, arXiv:0910.4952, 2009. 8

[20] CYRIL GAVOILLE, ADRIAN KOSOWSKI, AND MARCIN MARKIEWICZ: What can be observed locally? In *Proceedings of the 23rd International Symposium on Distributed Computing (DISC 2009)*, volume 5805 of *Lecture Notes in Computer Science*, pp. 243–257. Springer, 2009. 8

[21] JULIEN M. HENDRICKX: Views in a graph: To which depth must equality be checked? *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1907–1912, 2014. 11

[22] LISA HIGHAM, DAVID G. KIRKPATRICK, KARL R. ABRAHAMSON, AND ANDREW ADLER: Optimal algorithms for probabilistic solitude detection on anonymous rings. *Journal of Algorithms*, 23(2):291–328, 1997. 2

[23] ALON ITAI AND MICHAEL RODEH: Symmetry breaking in distributive networks. In *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science*, pp. 150–158, 1981. 2

[24] ALON ITAI AND MICHAEL RODEH: Symmetry breaking in distributed networks. *Information Computation*, 88(1):60–87, 1990. 2, 3, 5, 7, 11

[25] PHILLIP KAYE, RAYMOND LAFLAMME, AND MICHELE MOSCA: *An Introduction to Quantum Computing*. Oxford University Press, 2007. 11

[26] ALEXEI YU. KITAEV, ALEXANDER H. SHEN, AND MIKHAIL N. VYALYI: *Classical and Quantum Computation*. Volume 47 of *Graduate Studies in Mathematics*. AMS, 2002. 11

[27] EVANGELOS KRANAKIS AND DANNY KRIZANC: Distributed computing on anonymous hypercube networks. *Journal of Algorithms*, 23(1):32–50, 1997. 2, 11

[28] EVANGELOS KRANAKIS, DANNY KRIZANC, AND JACOV VAN DEN BERG: Computing Boolean functions on anonymous networks. *Information Computation*, 114(2):214–236, 1994. 2, 4, 6, 7, 8, 11, 30

[29] NANCY A. LYNCH: *Distributed Algorithms*. Morgan Kaufman Publishers, 1996. 9

[30] MICHAEL A. NIELSEN AND ISAAC L. CHUANG: *Quantum Computation and Quantum Information*. Cambridge University Press, 2000. 11

[31] N. NORRIS: Universal covers of graphs: Isomoriphism to depth n-1 implies isomoriphism to all depths. *Discrete Applied Mathematics*, 56(1):61–74, 1995. 11

[32] SUDEBKUMAR PRASANT PAL, SUDHIR KUMAR SINGH, AND SOMESH KUMAR: Multi-partite quantum entanglement versus randomization: Fair and unbiased leader election in networks. arXiv:quant-ph/0306195, 2003. 8

[33] SEIICHIRO TANI: Compression of view on anonymous networks – folded view –. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):255 – 262, 2012. 7, 11

[34] SEIICHIRO TANI, HIROTADA KOBAYASHI, AND KEIJI MATSUMOTO: Exact quantum algorithms for the leader election problem. In *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS '05)*, volume 3404 of *Lecture Notes in Computer Science*, pp. 581–592. Springer, 2005. 2, 4, 5, 7

[35] SEIICHIRO TANI, HIROTADA KOBAYASHI, AND KEIJI MATSUMOTO: Exact quantum algorithms for the leader election problem. *ACM Transactions on Computation Theory*, 4(1):Article 1, 2012. 2, 4, 5, 7, 26

[36] MASAFUMI YAMASHITA AND TSUNEHIKO KAMEDA: Computing on an anonymous network. In *Proceedings of the Seventh ACM Symposium on Principles of Distributed Computing*, pp. 117–130, 1988. 2, 4

[37] MASAFUMI YAMASHITA AND TSUNEHIKO KAMEDA: Computing on anonymous networks: Part I – characterizing the solvable cases. *IEEE Transactions on Parallel Distributed Systems*, 7(1):69–89, 1996. 2, 3, 4, 7, 9, 11

[38] MASAFUMI YAMASHITA AND TSUNEHIKO KAMEDA: Computing on anonymous networks: Part II – decision and membership problems. *IEEE Transactions on Parallel Distributed Systems*, 7(1):90–96, 1996. 2

Appendix

A Theorem 7 in Ref. [28]

For completeness, we present Theorem 7 in Ref. [28], which is used for deriving Corollary 1.6. The stochastic matrix P of a network with the underlying undirected graph G is defined as $P = DA$, where A is the adjacency matrix of G, and D is the diagonal matrix with $D_{ii} = 1/d(i)$ for the degree $d(i)$ of vertex i.

Theorem A.1 ([28, Theorem 7])**.** *Let G be an undirected connected graph on n vertices. Let $\mathcal{N}$ be an anonymous classical network with the underlying graph G and let $\mathcal{N}'$ be the network $\mathcal{N}$ with self-loops added to each vertex of G. Let λ be the second largest eigenvalue (in absolute value) of the stochastic matrix P associated with $\mathcal{N}'$. There exists a classical algorithm that computes any symmetric function on the network $\mathcal{N}$ in $O\left(\frac{\log n}{\log(1/\lambda)}\right)$ rounds and with bit complexity $O\left(\frac{m}{\log(1/\lambda)}(\log n)^2\right)$.*

B Proof of Lemma 5.2

Next we provide a proof of Lemma 5.2 in Sec. 5.2.

Proof of Lemma 5.2. Once a unique leader is elected, the following procedure can recognize the underlying graph. First, construct a spanning tree in $O(n)$ rounds with $O(m)$ bit complexity by traversing the graph for the number m of edges of the underlying graph. Second, assign a unique identifier to each party in $O(n)$ rounds with bit complexity $O(n\log n)$ by making a message with a $\lceil \log n \rceil$-bit counter traverse the spanning tree starting at the leader. Finally, gather into the leader the information about which parties are adjacent to each party by conveying adjacency matrices along the spanning tree as follows: Each party communicates with every adjacent party to determine the identifier of the adjacent party in one round, in which $O(m\log n)$ bits are communicated over all parties. Next, each leaf-node party i prepares an n-by-n adjacency matrix with all entries being zero, puts 1 in the entries (i, j) of the matrix for all adjacent parties j, and then sends the matrix to its parent-node party of the tree with $O(n^2)$ bit complexity. Every internal-node party k of the tree takes the entry-wise OR of all the received matrices, puts 1 in the entries (k, j) for all adjacent parties j, and then sends the resulting matrix to its parent-node party. Finally, the leader obtains the adjacency matrix of the underlying graph, and then broadcasts the matrix along the tree. These gathering and broadcasting steps take $O(n)$ rounds with bit complexity $O(n^3)$ since the adjacentcy matrix consists of n^2 bits and the number of edges of the tree is $n-1$. □

AUTHORS

Hirotada Kobayashi
Principles of Informatics Research Division,
National Institute of Informatics.
hirotada@nii.ac.jp

Keiji Matsumoto
Principles of Informatics Research Division,
National Institute of Informatics.
keiji@nii.ac.jp

Seiichiro Tani
NTT Communication Science Laboratories,
NTT Corporation.
tani.seiichiro@ntt.lab.co.jp
http://www.brl.ntt.co.jp/people/tani/