

Sublinear-Time Quantum Computation of the Diameter in CONGEST Networks

François Le Gall
Graduate School of Informatics
Kyoto University
legall@i.kyoto-u.ac.jp

Frédéric Magniez
IRIF, Univ Paris Diderot, CNRS
frederic.magniez@irif.fr

Abstract

The computation of the diameter is one of the most central problems in distributed computation. In the standard CONGEST model, in which two adjacent nodes can exchange $O(\log n)$ bits per round (here n denotes the number of nodes of the network), it is known that exact computation of the diameter requires $\tilde{\Omega}(n)$ rounds, even in networks with constant diameter. In this paper we investigate quantum distributed algorithms for this problem in the quantum CONGEST model, where two adjacent nodes can exchange $O(\log n)$ quantum bits per round. Our main result is a $\tilde{O}(\sqrt{nD})$ -round quantum distributed algorithm for exact diameter computation, where D denotes the diameter. This shows a separation between the computational power of quantum and classical algorithms in the CONGEST model. We also show an unconditional lower bound $\tilde{\Omega}(\sqrt{n})$ on the round complexity of any quantum algorithm computing the diameter, and furthermore show a tight lower bound $\tilde{\Omega}(\sqrt{nD})$ for any distributed quantum algorithm in which each node can use only $\text{poly}(\log n)$ quantum bits of memory.

1 Introduction

Diameter computation in CONGEST networks. The computation of the diameter is one of the most fundamental problems in distributed computing. In CONGEST networks, in which communication between nodes occurs with round-based synchrony and each channel has only $O(\log n)$ -bit bandwidth, where n denotes the number of nodes of the network, the diameter can be computed in $O(n)$ rounds [HW12, PRT12]. A matching lower bound $\tilde{\Omega}(n)$ has been first shown by Frischknecht et al. [FW12],¹ which also holds for sparse networks [ACHK16] and even for deciding whether the network has diameter 2 or diameter 3 [HW12]. The latter result immediately implies (see also [ACHK16, BK17]) that any distributed algorithm that computes a $(3/2 - \varepsilon)$ -approximation of the diameter, for any constant $\varepsilon > 0$, requires $\tilde{\Omega}(n)$ rounds.

If larger approximation ratios are allowed, however, sublinear-time² approximation algorithms can be constructed. First note that a 2-approximation of the diameter can trivially be computed in $O(D)$ rounds, where D denotes the diameter of the graph, by computing the eccentricity of any node. Much more interestingly, Lenzen and Peleg constructed a $O(\sqrt{n \log n} + D)$ -round 3/2-approximation algorithm [LP13], which was improved by Holzer et al. to $O(\sqrt{n \log n} + D)$ rounds [HPRW14].

Quantum distributed computing. While it is well known that quantum communication can offer significant advantages over classical communication in several settings such as two-party communication complexity (see, e.g., [deW02, BT08, DP08]), there are relatively few results directly relevant to distributed network computation. One first evidence of the potential of quantum distributed computing was

¹In this paper the notation $\tilde{O}(\cdot)$ suppresses $\text{poly}(\log n)$ factors, and the notation $\tilde{\Omega}(\cdot)$ suppresses $\frac{1}{\text{poly}(\log n)}$ factors.

²As usual when discussing algorithms in the CONGEST model, the *time complexity* of an algorithm refers to its round complexity. A sublinear-time algorithm means a $O(n^{1-\delta})$ -round algorithm for some constant $\delta > 0$.

| Problem | Classical | Quantum | |
|--------------------------------------|------------------------------------------|-----------------------------------|-------|
| Exact computation | $O(n)$ [HW12, PRT12] | $O(\sqrt{nD})$ | Th. 1 |
| Exact computation | $\tilde{\Omega}(n)$ [FHW12] | $\tilde{\Omega}(\sqrt{n} + D)$ | Th. 2 |
| | | $\tilde{\Omega}(\sqrt{nD/s} + D)$ | Th. 3 |
| 3/2-approximation | $\tilde{O}(\sqrt{n} + D)$ [LP13, HPRW14] | $\tilde{O}(\sqrt[3]{nD} + D)$ | Th. 4 |
| $(3/2 - \varepsilon)$ -approximation | $\tilde{\Omega}(n)$ [HW12, ACHK16, BK17] | $\tilde{\Omega}(\sqrt{n} + D)$ | Th. 2 |

Table 1: Our results on the quantum round complexity of computing/approximating the diameter, and the corresponding known classical results. In this table n denotes the number of nodes in the network, D denotes the diameter and s denotes the quantum memory used by each node.

the design of exact quantum protocols for leader election [TKM12]. Gavaille et al. [GKM09] then considered quantum distributed computing in the LOCAL model, and showed that for several fundamental problems, allowing quantum communication does not lead to any significant advantage. The power of distributed network computation in the CONGEST model has recently been investigated by Elkin et al. [EKNP14]. In this model the nodes can use quantum processing and communicate using quantum bits (qubits): each edge of the network corresponds to a quantum channel (e.g., an optical fiber if qubits are implemented using photons) of bandwidth $O(\log n)$ qubits. The main conclusions reached in that paper were that for many fundamental problems in distributed computing, such as computing minimum spanning trees or minimum cuts, quantum communication does not, again, offer significant advantages over classical communication. The main technical contribution of [EKNP14] was the introduction of techniques to prove lower bound for quantum distributed computation, which is significantly more challenging than proving lower bounds in the classical setting due to several specific properties of quantum information, such as quantum non-locality and the impossibility of “copying” quantum information. A pressing open question is to understand for which important problems in distributed computing quantum communication can help.

Our results. In this work we consider quantum distributed network computation in the CONGEST model, and especially investigate the complexity of computing the diameter. Our main contributions are for the exact computation of the diameter. We present the first quantum distributed algorithm that overcomes classical algorithms for this task:

Theorem 1. *There exists a $\tilde{O}(\sqrt{nD})$ -round quantum distributed algorithm, in which each node uses $O((\log n)^2)$ qubits of memory, that computes with probability at least $1 - 1/\text{poly}(n)$ the diameter of the network, where n denotes the number of nodes of the network and D denotes the diameter.*

Theorem 1 shows in particular that if the diameter is small (constant or at most polylogarithmic in n), then it can be computed in $\tilde{O}(\sqrt{n})$ time in the quantum setting. Moreover, whenever the diameter is $O(n^{1-\delta})$ for some constant $\delta > 0$, it can be computed in sublinear time. This significantly contrasts with the classical setting, where even distinguishing if the diameter is 2 or 3 requires $\tilde{\Omega}(n)$ time [HW12], as already mentioned. Note that our quantum algorithm uses only $O((\log n)^2)$ space, i.e., each node only needs to keep $O((\log n)^2)$ qubits of memory at any step of the computation. This low space complexity is especially appealing due to the technological challenges to construct large-scale quantum memory.

A natural question is whether the upper bounds of Theorem 1 can further be improved. We first show the following lower bound by adapting the argument from prior works that lead to the classical lower bound [HW12].

Theorem 2. *Any quantum distributed algorithm that decides, with probability at least $2/3$, whether the diameter of the network is at most 2 or at least 3 requires $\tilde{\Omega}(\sqrt{n})$ rounds.*

Note that this result holds even for quantum algorithms with an arbitrary amount of memory. Theorem 2 shows that the upper bound of Theorem 1 is tight for networks with small diameter. A similar argument can actually be used to derive a $\tilde{\Omega}(\sqrt{n})$ lower bound for any quantum algorithm that decides if the diameter is at most d or at least $d + 1$ for larger values of d , but for large diameters this does not match the upper bound of Theorem 1. We succeed in proving another lower bound, which matches the upper bound of Theorem 1 for large diameters, under the assumption that the nodes in the distributed quantum algorithm use only small space. This is one of the main technical contributions of the paper.

Theorem 3. *Any quantum distributed algorithm, in which each node uses at most s qubits of memory, computing with probability at least $2/3$ the diameter of the network requires $\tilde{\Omega}(\sqrt{nD/s})$ rounds.*

Observe that Theorems 1 and 3 together completely (up to possible polylogarithmic factor) settle the complexity of distributed quantum exact computation of the diameter with small quantum memory. While the memory restriction in the lower bound is a significant assumption, we believe that Theorem 3 is still a fairly general result since most known quantum communication protocols (e.g., protocols used to show the superiority of quantum communication in the model of communication complexity) use only a polylogarithmic amount of quantum memory. We actually conjecture that the upper bound of Theorem 1 is tight even without any restriction on the size of quantum memory used by the nodes.

We then consider approximation algorithms and show that the round complexity can be further decreased if we are only interested in computing a $3/2$ -approximation of the diameter.

Theorem 4. *There exists a $\tilde{O}(\sqrt[3]{nD} + D)$ -round quantum distributed algorithm that computes with probability at least $1 - 1/\text{poly}(n)$ a $3/2$ -approximation of the diameter of the network.*

Whenever the diameter is small, this quantum approximation algorithm again provides a significant improvement over the best known classical algorithms [LP13, HPRW14] already mentioned. The quantum algorithm of Theorem 4 actually consists of two phases, one classical and one quantum. The quantum phase still has polylogarithmic memory per node.

Our results are summarized in Table 1.

Overview of our upper bound techniques. At a high level, our approach can be described as a distributed implementation of quantum search and its generalizations (quantum amplitude amplification and quantum optimization), which are fundamental quantum techniques well studied in the centralized setting and in two-party quantum communication. In Section 2.4 we show how to implement them in the distributed setting by electing a leader in the network who will coordinate the quantum search. We develop a general framework involving three basic operations, Initialization, Setup and Evaluation, which need to be implemented on the whole network to perform quantum distributed optimization. This framework is general and can be applied to a large class of search or optimization problems on a distributed network as long as quantum communication is allowed between the nodes. The round complexity of the whole approach depends on the round complexity of each of the three basic operations, which is naturally problem-specific. We show how to implement them efficiently for the case of diameter computation.

For exact diameter computation (Theorem 1), the optimization problem we consider corresponds to finding a set S of $\Theta(n/D)$ nodes that contains a vertex of maximum eccentricity in the network. The most delicate part is the implementation of the Evaluation operation, which given any set S distributed (as a quantum superposition) among all the nodes of the network needs to compute the maximum eccentricity among all nodes in S . Our implementation is based on a refinement of the deterministic classical distributed algorithm for shortest paths in [PRT12]. For our $3/2$ -approximation of the diameter (Theorem 4), the strategy is slightly different: our approach is inspired by the classical algorithm for $3/2$ -approximation by Holzer et al. [HPRW14].

Overview of our lower bound techniques. Essentially all the known lower bounds on the classical complexity of computing or approximating the diameter in the CONGEST model are obtained by an argument based on the two-party communication complexity of the disjointness function. More precisely,

the idea is to first reduce the two-party computation of DISJ_k , the disjointness function on k -bit inputs, for some value of k , to the computation or the estimation of the diameter of a carefully constructed network, and then use the fact that the two-party classical communication complexity of DISJ_k is $\Omega(k)$ bits [KS92, Raz92]. A first approach to obtain quantum lower bounds would be to use instead the (tight) quantum lower bound $\Omega(\sqrt{k})$ showed by Razborov [Raz03] for the quantum two-party communication complexity of DISJ_k . It turns out, however, that the dependence in \sqrt{k} is too weak to lead to non-trivial lower bounds on the quantum round complexity of diameter computation. Fortunately, we observe that we can instead use a recent result by Braverman et al. [BGK⁺15], which was obtained using quantum information complexity, showing that the r -message quantum two-party communication complexity of DISJ_k is $\tilde{\Omega}(k/r + r)$. This better dependence in k enables us to prove Theorem 2.

Proving the lower bound of Theorem 3, i.e., making the diameter appear in the lower bound, is much more challenging. Our approach is as follows. We start from the network introduced in [ACHK16] to prove a $\tilde{\Omega}(n)$ -round classical lower bound for deciding whether a sparse bipartite network has diameter 4 or diameter 5. We convert each edge between the left part and the right part of the network in this construction into a path of d dummy nodes (the total number of introduced node is $\Theta(n)$), which immediately gives a reduction from the computation of DISJ_k with $k \approx n$ into deciding whether the diameter is $d + 4$ or $d + 5$. Since d rounds of communication are needed to transfer 1 bit (or 1 quantum bit) from the left part to the right part of the network, we would expect the following statement to be true: any r -round algorithm computing the diameter can be converted into a $O(r/d)$ -message two-party protocol for DISJ_k with $\tilde{O}(r)$ qubits of communication. This would give, via the lower bound from [BGK⁺15] mentioned above, the claimed lower bound. The hard part is to prove this statement. While it is easy to see that the statement is true in the classical setting, in the quantum setting several difficulties arise (as in [EKNP14]) due to entanglement between the nodes of the network. Our main technical contribution (Theorem 11 in Section 6) shows that the statement is indeed true in the quantum setting if we assume that each node has small enough quantum memory.

2 Preliminaries

We assume that the reader is familiar with the basic notions of quantum computation and refer to, e.g., [NC11] for a good introduction. A quantum state will be written using the ket notation such as $|\psi\rangle$. A quantum register means a set of quantum bits (qubits). When the system has several identified registers, we indicate them as a subscript. For instance $|i\rangle_{\text{R}}|j\rangle_{\text{Q}}$, means that register R is in state $|i\rangle$, and register Q in state $|j\rangle$. Of course those could be entangled and in superposition. Then we write $|\psi\rangle = \sum_{i,j} \alpha_{ij} |i\rangle_{\text{R}}|j\rangle_{\text{Q}}$.

In this paper we will use the term *CNOT copy* to refer to the unitary operation on $2m$ qubits (for some integer m) that maps the state $|u\rangle|v\rangle$ to the state $|u\rangle|u \oplus v\rangle$ for any binary strings $u, v \in \{0, 1\}^m$, where $u \oplus v$ is the bitwise XOR operation on u and v . This quantum operation can be implemented by applying m Controlled NOT gates. In particular it maps the state $|u\rangle|0\rangle$, for any binary string u , to the state $|u\rangle|u\rangle$, and thus corresponds to a classical copy.

For any integer M , we denote by $[M]$ the set $\{1, 2, \dots, M\}$. For an undirected graph $G = (V, E)$, we write $d(u, v)$ the distance between $u, v \in V$, i.e., the minimum length of a path between u and v in G . The *eccentricity* of $u \in V$, denoted $\text{ecc}(u)$, is the maximum distance of u to any other node $v \in V$. Then the *diameter* D is simply the maximum eccentricity among all nodes $u \in V$. Finally, the Breadth First Search tree from u on G is denoted by $\text{BFS}(u)$ and the Depth First Search tree is denoted $\text{DFS}(u)$.

2.1 Quantum CONGEST networks

In this paper we consider the CONGEST communication model. The graph $G = (V, E)$ represents the topology of the network, executions proceed with round-based synchrony and each node can transfer one message of bw qubits to each adjacent node per round. Initially the nodes of the network do not

share any entanglement. In this paper all the networks are undirected and unweighted. Unless explicitly mentioned, the bandwidth bw will always be $\text{bw} = O(\log n)$, where $n = |V|$. The only exception is the results in Section 6.1, and in particular Theorem 11, where the bandwidth will be kept as a parameter. All links and nodes (corresponding to the edges and vertices of G , respectively) are reliable and suffer no faults. Each node has a distinct identifier. Initially, each node knows nothing about the topology of the network except the set of edges incident to itself and the value n .

We write $|\psi\rangle_v$ to denote a state in the memory space of node v . When such a state is entangled, we use the tensor product notation. For instance $\frac{1}{\sqrt{M}} \sum_{i \in [M]} \bigotimes_{v \in V} |i\rangle_v$ denotes a uniform superposition over integers in $[M]$ distributed over all nodes of V (i.e., each node is synchronized with the same value, in superposition).

When discussing the space complexity of quantum distributed algorithms, the memory refers to the number of qubits each node uses in its workspace. We assume for simplicity that each node performs unitary operations and measures its register only at the very end of the algorithm.³

2.2 Two-party communication complexity

Let X , Y and Z be three finite sets. Consider two players, usually called Alice and Bob, and assume that Alice receives as input an element $x \in X$, while Bob receives an element $y \in Y$. In the model of communication complexity, first introduced in the classical two-party setting by Yao [Yao79], the players want to compute a function $f: X \times Y \rightarrow Z$ by running a protocol such that, at the end of the protocol, both Alice and Bob obtain $f(x, y)$, and they want to minimize the communication. In the quantum communication model, introduced by Yao [Yao93], the players are allowed to communicate with qubits. More precisely, the quantum communication complexity of a quantum protocol is the maximum (over all inputs) number of qubits that the protocol sends. The quantum communication complexity of f is the minimum communication complexity of any quantum protocol that computes f with probability at least $2/3$.

For any integer $n \geq 1$, the disjointness function $\text{DISJ}_k: \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$ is the function such that $\text{DISJ}_k(x, y) = 0$ if and only if there exists an index $i \in \{1, \dots, k\}$ such that $x_i = y_i = 1$. It is well known that its (randomized) classical communication complexity is $\Theta(k)$ bits [KS92, Raz92]. Its quantum communication complexity is $\Theta(\sqrt{k})$ qubits [BCW98, HdW02, Raz03, AA05]. Recently Braverman et al. [BGK⁺15] proved the following lower bound for quantum protocol with limited interaction, i.e., when only a bounded number of messages can be exchanged between Alice and Bob, which significantly improved the previous bound from [JRS03]

Theorem 5 ([BGK⁺15]). *The r -message quantum communication complexity of DISJ_k is $\tilde{\Omega}(k/r + r)$.*

This result shows in particular that any $O(\sqrt{k})$ -qubit quantum protocol for DISJ_k requires $\tilde{\Omega}(\sqrt{k})$ messages, i.e., $\tilde{\Omega}(\sqrt{k})$ rounds of interaction between Alice and Bob.

2.3 Quantum generic search

We now review the general framework of quantum generic search in the centralized model (see for instance [MNR11] for a more thorough treatment). In this framework, we are given a set of items X and we are looking for a marked item $x \in M$, for some unknown subset $M \subseteq X$. We have two main black-box unitary operators, and their inverses, available to design a quantum search procedure. The first

³More general models, such as models allowing intermediate measurements and models that allow nodes to use both a quantum memory and a classical memory, can naturally be considered as well. We prefer not introducing such models in this paper since the precise definition (and in particular the relation between the classical part and the quantum part), which can for instance be proposed similarly to [LeG09], would be fairly technical. We just mention that under natural “hybrid” models allowing both classical and quantum parts, the lower bound of Theorem 3 would hold for any quantum algorithm that uses s qubits of quantum memory, independently to the quantity of classical memory used, since the difficulty only comes from the quantum memory. A very simple example is the straightforward generalization of our model that allows a first classical phase with unbounded classical memory and then a quantum phase using only s qubits of memory per node.

one, Setup, is the quantum analogue of a random sampling. The second one, Checking, is a quantum checking procedure.

It will be convenient to assume the existence of a third procedure, Initialization, that performs some global initialization: the procedure is applied only once at the beginning of the algorithm and its output is later used in order to implement Setup and Checking.

We identify a specific register $|\cdot\rangle_I$ that we call *internal*, which is the core of the algorithm. Its size is polylogarithmic in all parameters. It is used to encode the element we are looking for, to perform basic operations such as arithmetic operations (e.g., counting), and also to control the application of other unitary matrices. The quantum algorithm uses two additional registers to represent additional information created by the procedures Initialization and Setup.

The framework thus assumes that the following three quantum procedures are given (as black-boxes).

Initialization: Creates an initial state $|0\rangle_I|\text{init}\rangle$ with some possible precomputed information $|\text{init}\rangle$ and a distinguished element $0 \in X$, usually a long enough bit string of 0s.

Setup: Produces a superposition from the initial state:

$$|0\rangle_I|\text{init}\rangle \mapsto \sum_{x \in X} \alpha_x |x\rangle_I |\text{data}(x)\rangle |\text{init}\rangle,$$

where the α_x 's are arbitrary amplitudes and $\text{data}(x)$ represents some information depending on x .

Checking: Performs the transformation

$$|x, 0\rangle_I |\text{data}(x)\rangle |\text{init}\rangle \mapsto |x, b_x\rangle_I |\text{data}(x)\rangle |\text{init}\rangle,$$

where $b_x = 1$ if $x \in M$ and $b_x = 0$ otherwise.

Note that the procedures Initialization and Checking are often described as deterministic or randomized (i.e., classical) procedures. They can then be quantized using standard techniques: one first transforms it to a reversible map using standard techniques [Ben89], with potentially additional garbage whose size is of the same order as the initial memory space.

Define $P_M = \sum_{x \in M} |\alpha_x|^2$, the probability to observe a marked element when measuring Register I after one application of Setup on the initial state. Classically one could iterate $\Theta(1/P_M)$ time this process in order to get at least one marked element with high probability. Amplitude amplification explains how to get a marked element using simply $\Theta(1/\sqrt{P_M})$ iterations of Setup and Checking.

Theorem 6 (Amplitude amplification [BHT98]). *Let $\varepsilon > 0$ and assume that either $P_M = 0$ or $P_M \geq \varepsilon$ holds. Then, for any $\delta > 0$, there is a quantum algorithm that can decide if $M = \emptyset$ with success probability at least $1 - \delta$ using one unitary operator Initialization, $O(\log(1/\delta)/\sqrt{\varepsilon})$ unitary operators Setup and Checking, their inverse, and $O(\log(1/\delta)/\sqrt{\varepsilon})$ other basic operations (independent of M) on the internal register.*

More precisely, an observation of the internal register when the algorithm declares that $M \neq \emptyset$, outputs a random $x \in M$ with probability $|\alpha_x|^2/P_M$.

2.4 Distributed quantum optimization

Let us first explain how to derive a generic quantum procedure for optimization problems in the centralized model. Indeed, a well known application of amplitude amplification is optimization, such as minimum finding [DHHM06]. We are now given a procedure Evaluation instead of Checking, which evaluates the (unknown) function we want to maximize.

Evaluation: Performs the transformation

$$|x, 0\rangle_I |\text{data}(x)\rangle |\text{init}\rangle \mapsto |x, f(x)\rangle_I |\text{data}(x)\rangle |\text{init}\rangle,$$

where $f: X \rightarrow \mathbb{Z}$ is the function we want to maximize.

Define the probability P_{opt} to observe an element where f takes a maximum value when measuring Register I after one application of Setup on the initial state as

$$P_{\text{opt}} = \sum_{x: f(x) \text{ is maximal}} |\alpha_x|^2.$$

A standard application of amplitude amplification shows how to maximize f using $\Theta(1/\sqrt{P_{\text{opt}}})$ iterations of Setup and Evaluation. We state this result as follows.

Corollary 1 (Quantum optimization). *Let $\varepsilon > 0$ be such that $P_{\text{opt}} \geq \varepsilon$. Then, for any $\delta > 0$, there is a quantum algorithm that can find, with probability at least $1 - \delta$, some element x such that $f(x)$ is maximum, using one unitary operator Initialization, $O(\log(1/\delta)/\sqrt{\varepsilon})$ unitary operators Setup and Evaluation, and their inverses, and $O(\log(1/\delta)/\sqrt{\varepsilon})$ other basic operations (independent of f) on the internal register.*

Proof. The proof uses standard applications of Theorem 6 such as in [DHHM06]. We give the main lines of the reduction:

1. Start with some fixed $a \in X$;
2. Use amplitude amplification of Theorem 6 with $\varepsilon' = 1/2$ and $\delta' = \delta$ to find $b > a$;
3. If an element is found, then set $a = b$ and go to (2);
4. Else if $\varepsilon' > \varepsilon$, then set $\varepsilon' = \varepsilon'/2$ and go to (2);
5. Output a and stop.

The expected number of iterations, that is of jumps to Step (2), can be shown to be $O(\log(1/\varepsilon))$, with an expected number of applications of unitary maps and of other basic operations in $O(\log(1/\delta)/\sqrt{\varepsilon})$ using similar arguments to [DHHM06] for minimum finding. In order to get the claimed worst case bounds, one just need to abort the computation when too much resources have been used and to output the current value of a . \square

We now explain how to implement quantum optimization in the distributed model. In the distributed setting, the internal register I and therefore the control of the algorithm itself are simply centralized by a leader node. The two other registers $|data(x)\rangle$ and $|init\rangle$, however, can be distributed among all the nodes of the network, and implementing Procedures Setup and Evaluation thus generally requires communication through the network. As stated in Corollary 1, besides calls to Setup and Evaluation and their inverses, all the other operations performed in quantum optimization are done exclusively on the internal register I , and then can be implemented locally (i.e., without communication) by the chosen leader node. The round complexity of distributed quantum optimization thus depends on the round complexity of Procedures Setup and Evaluation. The precise statement follows.

Theorem 7 (Distributed quantum optimization). *Let $G = (V, E)$ be a distributed network with a pre-defined node leader $\in V$. Assume that Initialization can be implemented within T_0 rounds and using s memory per node in the quantum CONGEST model, and that unitary operators Setup and Evaluation and their inverses can be implemented within T rounds and using s memory per node. Assume that $s = \Omega(\log |X|)$.*

Let $\varepsilon > 0$ be such that $P_{\text{opt}} \geq \varepsilon$. Then, for any $\delta > 0$, the node leader can find, with probability at least $1 - \delta$, some element x such that $f(x)$ is maximum, in $T_0 + O(\log(1/\delta)/\sqrt{\varepsilon}) \times T$ rounds. This quantum algorithm uses $O(s \times \log(1/\varepsilon))$ memory per node.

Proof. The round complexity follows immediately from Corollary 1. Let us now analyze the amount of memory needed for the node leader. Since measurements in the procedure described in Corollary 1 are only performed at the end of the computation, all outcomes of amplitude amplification in Line (2) are recorded in the procedure, leading to an additional term $O(\log |X| \times \log(1/\varepsilon))$ in the memory size. The total amount of memory needed for the implementation is thus $O(s + \log |X| \times \log(1/\varepsilon)) = O(s \times \log(1/\varepsilon))$ qubits for leader and $O(s)$ qubits for the other nodes. \square

In $\text{ecc}(\text{leader}) = O(D)$ rounds:

1. First, only leader is activated, it declares itself as its own parent, and sends a message through all its edges in order to activate its neighbors for the next round.
2. When a message reaches node v from node u :
 - If v was already activated, then v ignores the message.
 - Else v becomes activated, it sets its parent to u and sends, through all its edges, its distance to leader in order to activate its neighbors for the next round.

Figure 1: Construction of $\text{BFS}(\text{leader})$ in Proposition 1.

3 First Algorithm : Exact Computation of the Diameter

Let $G = (V, E)$ denote the network we consider. We write $n = |V|$ and use D to denote its diameter. We assume that the network G has elected a node $\text{leader} \in V$, and computed its eccentricity $\text{ecc}(\text{leader})$. This can be done using standard methods in $O(D)$ classical rounds and $O(\log n)$ memory space per node. In addition, our algorithm will use a Breadth First Search tree rooted at leader (denoted $\text{BFS}(\text{leader})$), which can be computed efficiently as described in the following proposition.

Proposition 1. *There is a classical procedure that allocates to each node $v \in V$ its parent on $\text{BFS}(\text{leader})$ and its distance to leader in $O(D)$ rounds and $O(\log n)$ memory space per node.*

Proof. The procedure is the classical procedure described in Figure 1 that performs a Breadth First Search tree from leader. This procedure has complexity $O(D)$ and can be implemented with $O(\log n)$ bits of memory per node. \square

Our quantum algorithm will compute the diameter, i.e., the maximum eccentricity among all nodes, using quantum optimization. In Section 3.1, we first give a simpler quantum algorithm that computes the diameter in $O(\sqrt{nD})$ rounds. In Section 3.2 we then give our $O(\sqrt{nD})$ -round quantum algorithm.

3.1 A simpler quantum algorithm

We apply the framework of Section 2.4 with $X = V$ and the function $f: V \rightarrow \mathbb{Z}$ defined as

$$f(u) = \text{ecc}(u) \tag{1}$$

for all $u \in V$. Obviously, maximizing f gives the diameter of the network.

Procedure Initialization is the classical procedure of Proposition 1. This means that $|\text{init}\rangle$ represents the classical information computed by this classical procedure: this is a quantum register shared by all the nodes of the network, in which the part owned by each node contains its parent on $\text{BFS}(\text{leader})$ and its distance to leader.

For any $u_0 \in V$, the quantum state $|\text{data}(u_0)\rangle$ used in the quantum algorithm is defined as $|\text{data}(u_0)\rangle = \bigotimes_{v \in V} |u_0\rangle_v$. We set $\alpha_{u_0} = 1/\sqrt{n}$ for all $u_0 \in V$, and thus have $P_{\text{opt}} \geq 1/n$. We first observe that Procedure Setup can be implemented efficiently.

Proposition 2. *The procedure Setup can be implemented in $O(D)$ rounds and using $O(\log n)$ memory space per node.*

Proof. The node leader first prepares the quantum state

$$\frac{1}{\sqrt{n}} \sum_{u_0 \in V} |u_0\rangle_{\text{leader}}.$$

Then this state is simply broadcast using CNOT copies to all nodes of the network along $\text{BFS}(\text{leader})$, which takes d rounds. The resulting state is

$$\frac{1}{\sqrt{n}} \sum_{u_0 \in V} |u_0\rangle_{\text{leader}} \bigotimes_{v \in V} |u_0\rangle_v = \frac{1}{\sqrt{n}} \sum_{u_0 \in V} |u_0\rangle_{\text{leader}} |\text{data}(u_0)\rangle,$$

as required. \square

We now observe that Procedure Evaluation can also be implemented efficiently.

Proposition 3. *The procedure Evaluation for the function f defined in (1) can be implemented in $O(D)$ rounds and using $O(\log n)$ memory space.*

Proof. Consider the following problem: all the nodes of the network receive as input u_0 , for some fixed $u_0 \in V$, and the goal is for node leader to output $\text{ecc}(u_0)$. There is a straightforward $O(D)$ -round classical algorithm solving this problem using $O(\log n)$ bits of memory per node: node u_0 starts building a Breadth First Search tree rooted at u_0 , the nodes use this tree to compute their distance to u_0 , and finally $\text{ecc}(u_0)$ is transmitted to the node leader. Note that this algorithm is precisely a procedure to compute $f(u_0)$ when the nodes initially share the classical information contained in $|\text{data}(u_0)\rangle$ (i.e., the nodes all know u_0). As mentioned in Section 2.3, this procedure can then be quantized using standard techniques in order to obtain a quantum algorithm that performs the transformation

$$|u_0, 0\rangle_I |\text{data}(u_0)\rangle |\text{init}\rangle \mapsto |u_0, f(u_0)\rangle_I |\text{data}(u_0)\rangle |\text{init}\rangle,$$

with the same round and space complexities. \square

We are now ready to analyze the overall complexity of our algorithm. We apply Theorem 7 to maximize the function f defined in (1). Propositions 1, 2 and 3, along with the lower bound $P_{\text{opt}} \geq 1/n$, show that this quantum algorithm uses $O(\sqrt{nD})$ rounds and $O((\log n)^2)$ memory per node for any constant error probability δ .

3.2 The final quantum algorithm

We now show how to reduce the complexity to $O(\sqrt{nD})$. The key idea is to modify the function to be maximized, in order to increase P_{opt} when d is large. Before defining the new function, let us introduce some definitions.

We take some integer d such that $d \leq D \leq 2d$. Since $\text{ecc}(v) \leq D \leq 2 \times \text{ecc}(v)$ for any node $v \in V$, this will concretely be done by choosing $d = \text{ecc}(\text{leader})$. Similarly to [PRT12], we introduce the following numbering on the vertices of G via a Depth-First-Search traversal of $\text{BFS}(\text{leader})$.

Definition 1 (DFS-numbering). *The DFS(leader)-number of $v \in V$, denoted by $\tau(v)$, is the length of the path to reach v from leader on a DFS-traversal of $\text{BFS}(\text{leader})$. In particular $\tau(\text{leader}) = 0$.*

We now use the DFS-numbering to define sets of vertices.

Definition 2. *For any $u \in V$, define $S(u) = \{v \in V : \tau(v) \in [\tau(u), \tau(u) + 2d \bmod 2n]\}$. Let \mathcal{S} be the collection of subsets $S(u)$ for all $u \in V$.*

The following lemma, which will be crucial for our analysis, shows that \mathcal{S} covers well the set V .

Lemma 1. *Fix any $v \in V$, and take some $u_0 \in V$ uniformly at random. Then $\Pr[v \in S(u_0)] \geq \frac{d}{2n}$.*

Proof. Consider a sequential execution π of DFS(leader) on BFS(leader). We now consider π as a circle by attaching its extremities. Observe that each top-down move to some node v at step t causes the numbering $\tau(v) = t$. In addition, since BFS(leader) has depth d , any segment of π with m_d top-down moves and m_u bottom-up ones must satisfy $|m_d - m_u| \leq d$.

Thus for any vertex $v \in V$, consider the part of π of length $2d$ ending in a top-down move to v . It must contain at least $\lceil d/2 \rceil$ top-down moves to some vertices $u_1, \dots, u_{\lceil d/2 \rceil}$. Therefore $v \in S(u_i)$, for $i = 1, \dots, \lceil d/2 \rceil$, and we conclude that $\Pr[v \in S(u_0)] \geq \frac{d}{2n}$. \square

The function we will now optimize is the function $f: V \rightarrow \mathbb{Z}$ defined as

$$f(u) = \max_{v \in S(u)} (\text{ecc}(v)) \quad (2)$$

for any $u \in V$. Obviously, maximizing f gives the diameter of the network. We optimize this function via the distributed quantum optimization procedure of Section 2.4. We set X , α_x , $|\text{init}\rangle$ and $|\text{data}(x)\rangle$ exactly as in Section 3.1. Lemma 1 immediately implies that $P_{\text{opt}} \geq d/2n$.

The procedures Initialization and Setup are the same as in Section 3.1. The procedure Evaluation, however, is different since the function to evaluate is not the same. We describe the new procedure, which is based on a refinement of the deterministic distributed algorithm for diameter in [PRT12], in the next proposition. The proof of this proposition is postponed to Section 3.3.

Proposition 4. *The procedure Evaluation for the function f defined in (2) can be implemented in $O(D)$ rounds and using $O(\log n)$ memory space.*

We are now ready to analyze the overall complexity of our algorithm and prove Theorem 1.

Proof of Theorem 1. We apply Theorem 7 to maximize the function f defined in (2). Propositions 1, 2 and 4, along with the lower bound $P_{\text{opt}} \geq d/2n$, show that this quantum algorithm uses $O(\sqrt{nD})$ rounds and $O((\log n)^2)$ memory per node for any constant error probability δ . \square

3.3 The evaluation procedure

We prove Proposition 4 by describing the Procedure Evaluation for the function f defined in (2). Remember that this procedure should, for any $u_0 \in V$ given as input to all the nodes of the network, enable the node leader to output $f(u_0) = \max_{v \in S(u_0)} (\text{ecc}(v))$. Remember also that we are assuming the each node $v \in V$ knows its parent on BFS(leader) and its distance to leader (this information was computed in Procedure Initialization).

The procedure Evaluation is described in Figure 2. (The procedure is described as a classical procedure. As mentioned in Section 2.3 and in the proof of Proposition 3, it can then be quantized using standard techniques.)

The main idea is simple: each node $v \in S(u_0)$ starts a process in order to globally compute its eccentricity. This will be done in parallel, and the main difficulty is to avoid congestions. We achieve this goal by refining techniques used in the deterministic distributed algorithm from [PRT12]. As in [PRT12], we coordinate the starts of all the processes so that messages initiated by nodes $v \in S(u_0)$ arrive in the same order as their DFS(leader)-numbers. Therefore, there is no congestion, and messages to disregard are easily identified, even with a local memory of logarithmic size for each node.

A difference with [PRT12] is that we do not use the DFS(leader)-number $\tau(v)$, which is too costly to compute, but instead the quantity $\tau'(v) = \tau(v) - \tau(u_0)$, which is much easier to compute, to coordinate the starts of the process. More precisely, node $v \in S(u_0)$ starts its process at round $2\tau'(v)$, see Step 2(2) of the algorithm.

Another difference is that each node v in Step 2 of the algorithm only keeps in memory the maximum of their previously computed distances. Nodes receive at each round messages of the form (τ', δ) that they either disregard or broadcast after incrementing δ . This message has been initiated at round τ' by

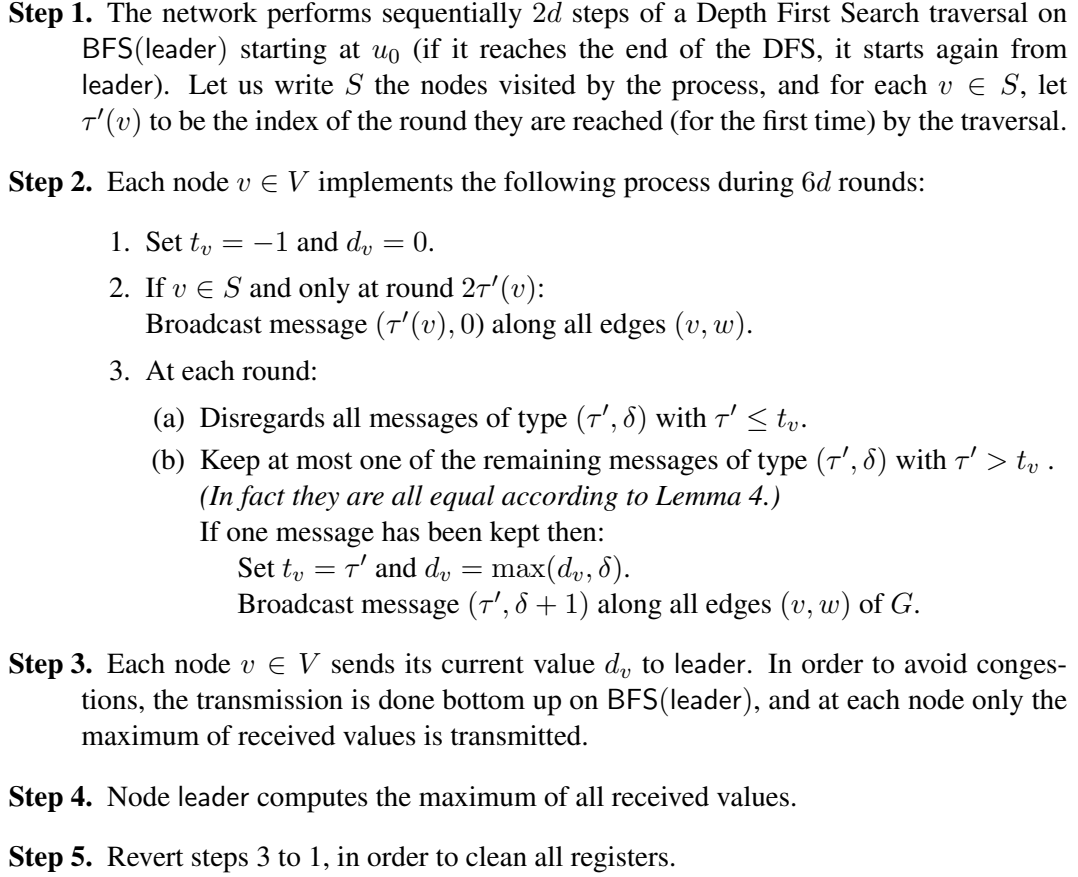


Figure 2: The procedure Evaluation.

the node $u \in S$ such that $\tau' = \tau'(u)$ in Step 2(2). The number δ is the number of rounds this message took to reach v for the first time, i.e., $\delta = d(u, v)$. The value t_v is used to decide which messages can be disregarded. The value d_v represents the maximum distance $d(u, v)$ over all the nodes u processed so far.

The procedure Evaluation can be implemented in $O(D)$ rounds and using $O(\log n)$ space per node since at any time each node only needs to keep one message (τ', δ) , in addition to t_v and d_v , in memory. The correctness follows from the following three lemmas, which together show that the procedure computes the maximum of $\text{ecc}(v)$ over all $v \in S(u_0)$.

Lemma 2. *At Step 1, the computed set S satisfies $S = S(u_0)$. Furthermore, for every nodes $v, w \in S$, if $\tau'(v) < \tau'(w)$ then $d(v, w) \leq \tau'(w) - \tau'(v)$.*

Proof. First observe that at Step 1, $\tau'(v) = \tau(v) - \tau(u_0)$, for all $v \in S(u_0)$. Therefore, the computed set S satisfies $S = S(u_0)$. Then the rest of the statement is a direct consequence of [PRT12, Property 1] which was stated for the DFS(leader)-numbering τ , and therefore remains valid for τ since $\tau'(v) = \tau(v) - \tau(u_0)$, for all $v \in S(u_0)$. \square

Lemma 3. *Let r_1 be the first round that a message of type $(\tau'_1, *)$ reaches a given node v , and similarly for r_2 and τ'_2 . Then $r_2 - r_1 \geq \tau'_2 - \tau'_1$. In particular, if $\tau'_1 < \tau'_2$ then $r_1 < r_2$.*

Proof. Let z_1, z_2 be such that $\tau'_i = \tau'(z_i)$, for $i = 1, 2$. This is the first time that a message from z_i reaches v , therefore $r_i = 2\tau'(z_i) + d(z_i, v)$. In addition, by the triangle inequality, we also have that $d(z_1, v) - d(z_2, v) \leq d(z_1, z_2)$. Thus

$$d(z_1, z_2) \geq d(z_1, v) - d(z_2, v) = r_1 - r_2 + 2(\tau'(z_2) - \tau'(z_1)).$$

Preparation in $\tilde{O}(n/s + D)$ rounds. This part is the same as Steps 1 to 5 of Algorithm 1 in [HPRW14]:

1. Each vertex joins a set S with probability $(\log n)/s$.
If more than $(n(\log n)^2/s)$ vertices join S , then abort.
2. Each vertex $v \in V$ computes the closest node in S to v , which is denoted $p(v)$.
Compute a node $w \in V$ maximizing the distance $d(w, p(w))$.
3. Compute a BFS tree from w .
The s closest nodes to w in the tree join a set R .

Quantum optimization $\tilde{O}(\sqrt{sD} + D)$ rounds. Compute the maximum eccentricity of the vertices in R .

Output. Return the maximum computed eccentricity.

Figure 3: Our quantum algorithm computing a $3/2$ -approximation of the diameter.

Since $\tau'(z_1) < \tau'(z_2)$, then by Lemma 2 we also have that

$$d(z_1, z_2) \leq \tau'(z_2) - \tau'(z_1).$$

Therefore we get

$$r_2 - r_1 \geq \tau'(z_2) - \tau'(z_1) > 0,$$

which proves the lemma. \square

Lemma 4. *At Step 3b, all remaining messages are identical.*

Proof. Assume by contradiction that at least two messages (τ'_1, δ_1) and (τ'_2, δ_2) remain such that either $\tau'_1 \neq \tau'_2$ or $\delta_1 \neq \delta_2$. By construction of our algorithm, they are the first messages of types $(\tau'_1, *)$ and $(\tau'_2, *)$ reaching v .

If $\tau'_1 \neq \tau'_2$, by Lemma 3, those messages cannot arrive at the same round, which contradicts the fact they both arrive at the current round.

If $\tau'_1 = \tau'_2$, but $\delta_1 \neq \delta_2$, we get another contradiction. Indeed, those messages come from the same vertex u such that $\tau'(u) = \tau'_1$, and reach v at the same round for the first time. Thus they must satisfy $\delta_1 = \delta_2$. \square

4 Second Algorithm: $3/2$ -Approximation of the Diameter

The framework for distributed quantum optimization we developed in this paper, and in particular its application to find vertices of maximum eccentricities presented in Section 3, can be used as well to design fast quantum approximation algorithms for the diameter. In this section we show how to apply our framework to speed up the $3/2$ classical approximation algorithm designed in [HPRW14], which computes a value \bar{D} such that $\bar{D} \leq D \leq 3\bar{D}/2$, and prove Theorem 4.

Our quantum algorithm for $3/2$ -approximation is described in Figure 3. In the description of the algorithm, $s \in [n]$ is an integer parameter that will be set later.

This quantum algorithm is exactly the same as Algorithm 1 in [HPRW14], except for the second part: in [HPRW14] the maximum eccentricity was computed classically in $O(s + D)$ rounds by computing a BFS tree for each vertex in R . We thus omit the details about the implementation of the first part and the proof of correctness of the whole algorithm (which can be found in [HPRW14]). Let us nevertheless

point out that this first part requires a polynomial amount of (classical) memory. The quantum procedure for the second part we describe below, however, will only require a polylogarithmic amount of memory.

The quantum procedure for the second part is essentially the same as the quantum algorithm presented in Section 3, which computed the maximum eccentricity of all the nodes in V (i.e., the diameter of the network). The difference is that this time we do the optimization only on the vertices in R : the function to optimize the function of Equation (2) restricted to the domain $R \subseteq V$. By replacing node leader by node w , and replacing “mod $2n$ ” by “mod $2s$ ” in Definition 2, we obtain $P_{\text{opt}} \geq d/2s$ instead $P_{\text{opt}} \geq d/2n$ since R is defined as the set of the s closest nodes to w . By additionally modifying the procedure Setup so that it distributes the state $\frac{1}{\sqrt{s}} \sum_{u_0 \in R} \bigotimes_{v \in V} |u_0\rangle_v$ instead of $\frac{1}{\sqrt{n}} \sum_{u_0 \in V} \bigotimes_{v \in V} |u_0\rangle_v$, Theorem 7 implies that the maximum eccentricity of the vertices in R can be computed with high probability in $\tilde{O}(\sqrt{sD} + D)$ rounds and using $O((\log n)^2)$ qubits of memory per node.

The overall round complexity of our algorithm is $\tilde{O}(n/s + \sqrt{ns} + D)$. Choosing $s = \Theta(n^{2/3}D^{-1/3})$ gives complexity $\tilde{O}(\sqrt[3]{nD} + D)$, as claimed in Theorem 4.

5 Lower Bound: Tight Bound for Small Diameter

In this section we show that the connection established in prior works (in particular in [FHW12, HW12, ACHK16, BK17]) between the round complexity of computing the diameter and the two-party communication complexity of the disjointness function still holds in the quantum setting. We then use this connection to prove Theorem 2.

In the following we will use the notation $G = (U, V, E)$ to denote a bipartite graph whose partition has parts U and V : it will always be implicitly assumed that the sets U and V are disjoint, and that each edge in E has an endpoint in U and its other endpoint in V . We will use $\Delta(G)$ to denote the largest distance between one vertex in U and one vertex of V . We will use $\mathcal{E}(U)$ to denote the set of all pairs of elements in U and $\mathcal{E}(V)$ to denote the set of all pairs of elements in V .

All the recent lower bounds on the classical round complexity of computing or approximating the diameter rely on the notion of reduction introduced in the next definition.

Definition 3. Let b, k, d_1 and d_2 be four functions from \mathbb{N} to \mathbb{N} . A (b, k, d_1, d_2) -reduction from disjointness to diameter computation is a family of triples $\{(G_n, g_n, h_n)\}_{n \geq 1}$ such that, for each $n \geq 1$, $G_n = (U_n, V_n, E_n)$ is a bipartite graph with $|U_n \cup V_n| = n$ and $|E_n| = b$, g_n is a function from $\{0, 1\}^k$ to $\mathcal{E}(U_n)$ and h_n is a function from $\{0, 1\}^k$ to $\mathcal{E}(V_n)$ satisfying the following two conditions for any $x, y \in \{0, 1\}^k$:

(i) if $\text{DISJ}_k(x, y) = 1$ then $\Delta(G_n(x, y)) \leq d_1$,

(ii) if $\text{DISJ}_k(x, y) = 0$ then $\Delta(G_n(x, y)) \geq d_2$,

where $G_n(x, y)$ denotes the bipartite graph obtained from G_n by adding an edge $\{w, w'\}$ for each $\{w, w'\} \in g_n(x) \cup h_n(y)$.

In particular we will use in this paper the following two constructions from prior works.

Theorem 8 ([HW12]). *There exists a $(\Theta(n), \Theta(n^2), 2, 3)$ -reduction from disjointness to diameter computation.*

Theorem 9 ([ACHK16]). *There exists a $(\Theta(\log n), \Theta(n), 4, 5)$ -reduction from disjointness to diameter computation.*

We include a proof of Theorem 8 as an example to illustrate the notations of Definition 3.

Proof of Theorem 8. We define the graph $G_n = (U_n, V_n, E_n)$ as follows. Here we assume, for simplicity, that $n - 2$ is a multiple of 4 (the construction can be easily modified, by adding dummy nodes, to deal

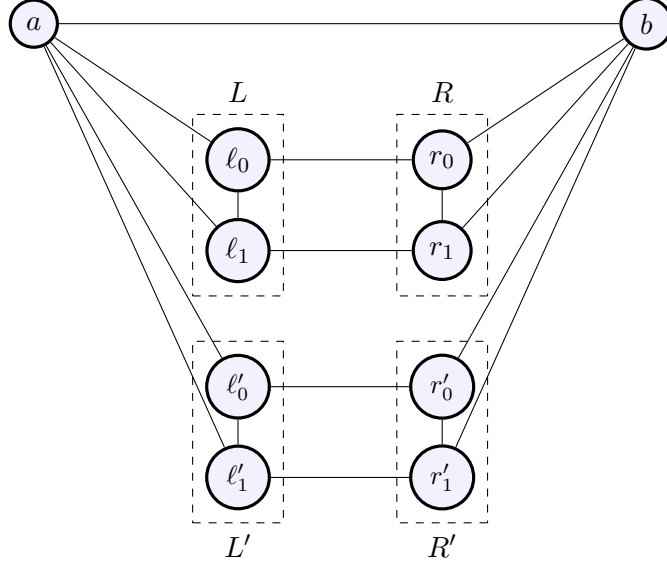


Figure 4: The graph G_n used in the proof of Theorem 8, for $n = 2$.

with the case when n is not of this form). We define $s = (n - 2)/4$. The construction is illustrated for $n = 10$ in Figure 4.

We have $U_n = L \cup L' \cup \{a\}$ and $V_n = R \cup R' \cup \{b\}$, where $L = \{\ell_0, \dots, \ell_{s-1}\}$, $L' = \{\ell'_0, \dots, \ell'_{s-1}\}$, $R = \{r_0, \dots, r_{s-1}\}$ and $R' = \{r'_0, \dots, r'_{s-1}\}$ are four sets each containing s nodes, and a, b are two additional nodes. The set E_n is defined as follows: we put edges to make each of L , L' , R and R' an s -clique. Moreover, for each $i \in \{0, \dots, s-1\}$, we add an edge from ℓ_i to r_i and an edge from ℓ'_i to r'_i . Finally, node a is connected by an edge to each node in $L \cup L'$, while node b is connected by an edge to each node in $R \cup R'$. Nodes a and b are also connected by an edge. Note that $|E_n| = 2s + 1 = \Theta(n)$.

Now we define the functions g_n and h_n . Given any binary string $x \in \{0, 1\}^{s \times s}$, we define $g_n(x)$ as the set of all pairs $\{\ell_i, \ell'_j\}$ with $i, j \in \{0, \dots, s-1\}$ such that $x_{i,j} = 0$. Similarly, given any binary string $y \in \{0, 1\}^{s \times s}$, we define $h_n(y)$ as the set of all pairs $\{r_i, r'_j\}$ with $i, j \in \{0, \dots, s-1\}$ such that $y_{i,j} = 0$. This means that the graph $G_n(x, y)$ is obtained from G_n by putting an additional edge between node ℓ_i and node ℓ'_j if and only if $x_{i,j} = 0$, and similarly by putting an additional edge between node r_i and node r'_j if and only if $y_{i,j} = 0$.

It is easy to check that conditions (i) and (ii) of Definition 3 hold with $k = s^2 = \Theta(n^2)$, $d_1 = 2$ and $d_2 = 3$, by observing that the distance in $G_n(x, y)$ between ℓ_i and r'_j (and, symmetrically, the distance between ℓ'_j and r_i as well) is 3 if $x_{i,j} = y_{i,j} = 1$, and 2 otherwise. \square

An easy argument, used in all these prior works mentioned above, shows that the existence of a (b, k, d_1, d_2) -reduction from disjointness to diameter computation implies a $\tilde{\Omega}(k/b)$ lower bound on the round complexity of any classical distributed algorithm that decides whether the diameter of the network is at most d_1 or at least d_2 . The main technical contribution of this section is the following quantum version of this argument, which is proved using the recent result from [BGK⁺15] on the message-bounded quantum communication complexity of disjointness presented in Section 2.2.

Theorem 10. *Assume that there exists a (b, k, d_1, d_2) -reduction from disjointness to diameter computation. Then any quantum distributed algorithm that decides, with high probability, whether the diameter of the network is at most d_1 or at least d_2 requires $\tilde{\Omega}(\sqrt{k/b})$ rounds.*

Proof. Assume that there exists an r -round quantum algorithm \mathcal{A} that decides if the diameter of a network is at most d_1 or at least d_2 . We will use this algorithm to construct a protocol that computes the disjointness function DISJ_k in the two-party quantum communication complexity setting. Let $x \in \{0, 1\}^k$ and $y \in \{0, 1\}^k$ be Alice's input and Bob's input, respectively.

The protocol works as follows. First note that Alice and Bob can jointly construct the graph $G_n(x, y)$: Alice can construct the subgraph induced by U_n (which depends on x), while Bob can construct the subgraph induced by V_n (which depends on y). Alice and Bob will thus simply simulate the computation of \mathcal{A} on $G_n(x, y)$. To do this, they only need to exchange messages corresponding to the communication occurring along the b edges of $G_n(x, y)$ at each round of Algorithm \mathcal{A} . This can be done by replacing one round of communication in \mathcal{A} by two messages of $O(b \log n)$ qubits (one from Alice to Bob, and the other from Bob to Alice). At the end of simulation Alice and Bob obtain the result of the computation by \mathcal{A} on $G_n(x, y)$. From our assumption on \mathcal{A} , this enables them to decide if the diameter of $G_n(x, y)$ is at most d_1 or at least d_2 , and thus to compute $\text{DISJ}_k(x, y)$. The whole simulation uses $2r$ messages and a total number of $O(rb \log n)$ qubits of communication. From Theorem 5, we conclude that $rb = \tilde{\Omega}\left(\frac{k}{r} + r\right)$, which implies $r = \tilde{\Omega}(\sqrt{k/b})$. \square

Theorem 2 directly follows by combining Theorem 8 and Theorem 10.

6 Lower Bound: Case of Large Diameter

In this section we prove Theorem 3. We first prove in Section 6.1 a general simulation result that will be crucial for our analysis, and then give the proof of Theorem 3 in Section 6.2.

6.1 Simulation argument

For any $d \geq 1$, let us introduce the following network \mathcal{G}_d . The network consists of two nodes A and B connected by a path of length $d + 1$, corresponding to d intermediate nodes P_1, \dots, P_d . The total number of nodes is thus $d + 2$, and the number of edges is $d + 1$. Each edge of \mathcal{G}_d is a quantum channel of bandwidth bw qubits. The construction is illustrated in Figure 5.

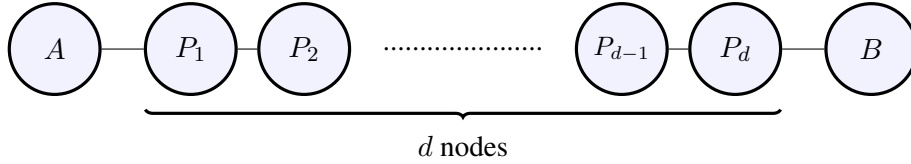


Figure 5: The graph \mathcal{G}_d .

Let X, Y be two finite sets, and let f be a function from $X \times Y$ to $\{0, 1\}$. Consider the following computation problem over the network \mathcal{G}_d : node A receives an input $x \in X$ and node B receives an input $y \in Y$. The goal is for Alice and Bob to compute the value $f(x, y)$. Note that the intermediate nodes P_1, \dots, P_d do not receive any input. The following theorem relates the round complexity of this problem to the two-party communication complexity of f .

Theorem 11. *Let d and r be any positive integers, and f be any Boolean function. If there exists an r -round quantum distributed algorithm, in which each intermediate node uses at most s qubits of memory, that computes function f with probability p over \mathcal{G}_d , then there exists a $O(r/d)$ -message two-party quantum protocol computing f with probability p using $O(r(\text{bw} + s))$ qubits of communication.*

Proof. Consider any r -round quantum algorithm \mathcal{A} computing function f over \mathcal{G}_d in which each intermediate node uses s qubits of internal memory. For notational convenience, we will use P_0 and P_{d+1} to refer to nodes A and B , respectively. For any $i \in \{0, 1, \dots, d+1\}$, let R_i denote the quantum register representing the memory of node P_i . Note that if $1 \leq i \leq d$ then R_i consists of at most s qubits, which we assume (without loss of generality) are initially in the all-zero quantum state. The registers R_0 and R_{d+1} contain the input x and y , respectively, in addition to qubits initially in the all-zero quantum state.

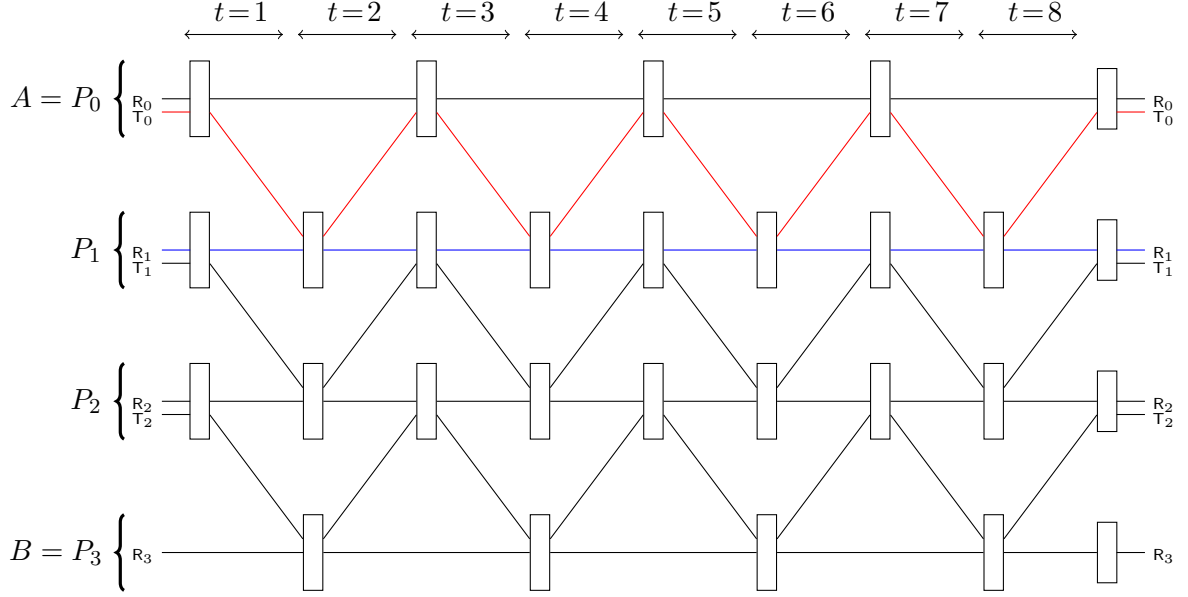


Figure 6: Representation of an 8-round algorithm for $d = 2$. Each thin line represents a quantum register. Each rectangle represents a unitary operator. The red line represents the message register T_0 (which is exchanged between P_0 and P_1 during the execution of the algorithm) and the green line represents the private register R_1 (which stays at node P_1 during the execution of the algorithm).

For simplify notations, we will make the following assumption: at each round $t \in \{1, \dots, r\}$ of the algorithm \mathcal{A} , if t is odd then messages are sent only from the left to the right (i.e, from P_i to P_{i+1} for each $i \in \{0, \dots, d\}$), and if t is even then messages are sent only from the right to the left (i.e, from P_i to P_{i-1} for each $i \in \{1, \dots, d+1\}$). This can be done without loss of generality since any algorithm can be converted into an algorithm satisfying this assumption by increasing the round complexity only by a factor 2. Without loss of generality, again, we can also assume that Algorithm \mathcal{A} proceeds as follows (see Figure 6 for an illustration for $d = 2$ and $r = 8$):

- Each node P_i , for $i \in \{0, \dots, d\}$, initially owns one additional b -qubit quantum register T_i containing the all-zero quantum state.
- At the first round, for each $i \in \{0, \dots, d\}$, node P_i performs a unitary operator on (R_i, T_i) , and then sends T_i to P_{i+1} . (Note that the contents of the register T_i can be — and will in general be — modified by the unitary operator.) Node P_{d+1} does nothing.
- At round number $t \in \{2, \dots, r\}$ with t even, for each $i \in \{1, \dots, d+1\}$ node P_i , who just received register T_{i-1} from P_{i-1} at the previous round, performs a unitary operator on (R_i, T_{i-1}) , and then sends T_{i-1} to P_{i-1} . Node P_0 does nothing.
- At round number $t \in \{3, \dots, r\}$ with t odd, for each $i \in \{0, \dots, d\}$ node P_i , who just received register T_i from P_{i+1} at the previous round, performs a unitary operator on (R_i, T_i) , and then sends T_i to P_{i+1} . Node P_{d+1} does nothing.
- After round number r (the last round of communication), each node performs a unitary operator on its registers, then measures them and decides its output depending of the outcome of the measurement.

The registers R_0, \dots, R_{d+1} will be called the private registers (since they do not move during the execution of the protocol), while the registers T_0, \dots, T_d will be called the message registers.

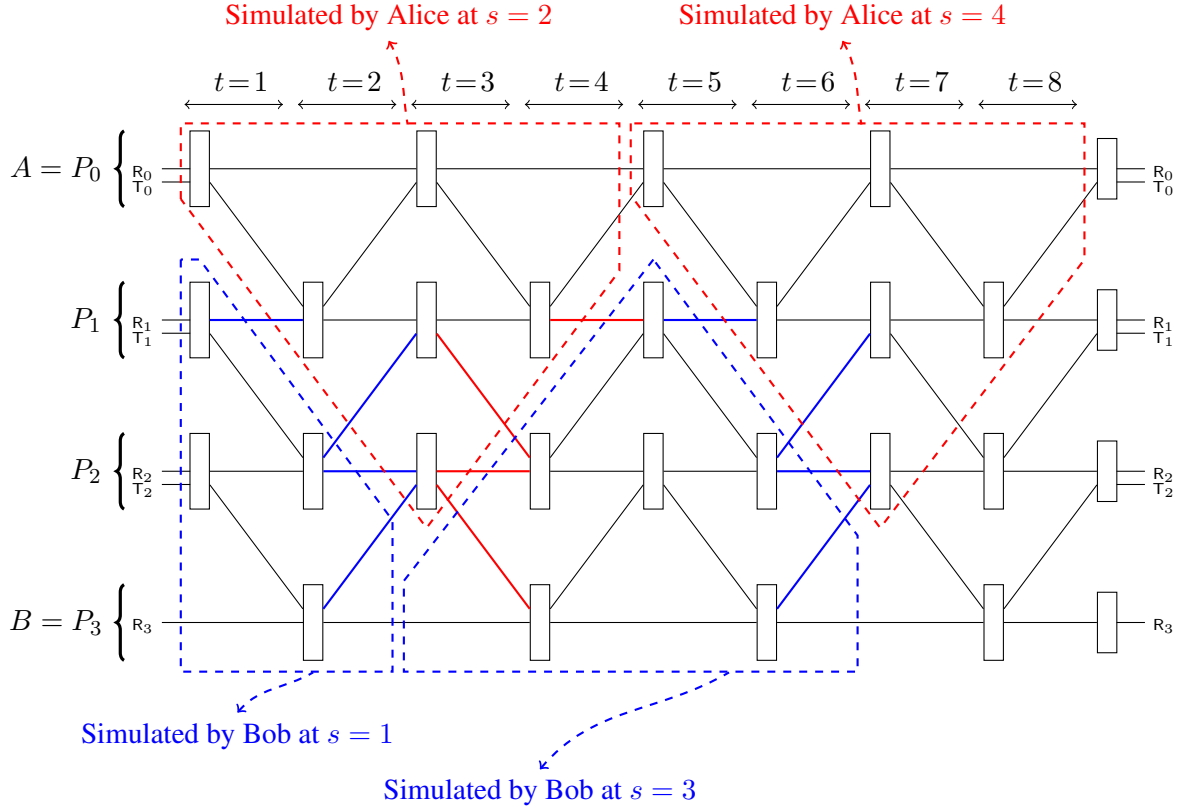


Figure 7: Simulation by \mathcal{P} of the algorithm from Figure 6. The blue and red dashed areas represent the parts that are simulated by Bob and Alice, respectively, during the protocol. The blue plain lines represent the quantum registers that are sent by Bob to Alice during the protocol. The red plain lines represent the quantum registers that are sent by Alice to Bob during the protocol. For instance, for $s = 1$, Bob will send to Alice four registers: he first simulates the computation of P_2 and P_3 until steps $t = 1$ and $t = 2$ of \mathcal{A} , respectively, and sends to Alice the corresponding message register of P_3 (since 3 is odd). He then simulates P_1 and P_2 until steps $t = 1$ and $t = 2$, respectively, and sends to Alice the message register of P_2 (since 2 is even). He also sends to Alice the private registers of P_1 and P_2 .

We now describe a $O(r/d)$ -message protocol \mathcal{P} that computes f in the standard model of communication complexity, where Alice and Bob receive as input x and y , respectively. The idea is that Alice and Bob jointly simulate the computation of \mathcal{A} . The crucial observation is that, due to the time needed for a message from A to reach B in the network \mathcal{G}_d , the simulation can be done by dividing the computation into $O(r/d)$ areas, each of width d , and doing the simulation area per area. Each area will be simulated by one of the players, who will then send to the other player only the $O(d)$ registers that are needed to simulate the next area. The details of the simulation are as follows (see also Figure 7 for an illustration).

Note that Alice can create the initial state of register R_0 , which depends only on x , and Bob can create the initial state R_{d+1} , which depends only on y . For simplicity we will assume that r is an even multiple of d , but the description can be readily adjusted for any value of r . For each s from 1 to r/d , Alice and Bob do the following:

- If s is odd then this is Bob's turn. Bob first simulates the computation of P_i in \mathcal{A} up to step $t = (s - 1)d + i - 1$, for each $i \in \{2, \dots, d + 1\}$. For each odd $i \in \{2, \dots, d + 1\}$, Bob then sends to Alice the message register of P_i . Bob then proceeds with the simulation and simulates the computation of P_i in \mathcal{A} up to step $t = (s - 1)d + i$, this time for each $i \in \{1, \dots, d\}$. For any even $i \in \{1, \dots, d\}$, Bob sends to Alice the message register of P_i . Finally, Bob sends to Alice the private register of P_i for each $i \in \{1, \dots, d\}$. Note that all the communication from Bob to

Alice can be concatenated into one message of $O(d(\text{bw} + s))$ qubits.

- If s is even then this is Alice's turn. Alice first simulates the computation of P_i in \mathcal{A} up to step $t = sd - i$, for each $i \in \{0, \dots, d - 1\}$. For any odd $i \in \{0, \dots, d - 1\}$, Alice then sends to Bob the message register of P_i . Alice then proceeds with the simulation and simulates the computation of P_i in \mathcal{A} up to step $t = sd - i + 1$, this time for each $i \in \{1, \dots, d\}$. For any even $i \in \{1, \dots, d\}$, Alice sends to Bob the message register of P_i . Finally, Alice sends to Bob the private register of P_i for each $i \in \{1, \dots, d\}$. Note that all the communication from Alice to Bob can be concatenated into one message of $O(d(\text{bw} + s))$ qubits.

After all those steps, Alice has received all the messages needed to complete the computation done by node A in Algorithm \mathcal{A} . She completes the computation, decides her output (as done by node A in \mathcal{A}), and sends it to Bob.

Protocol \mathcal{P} uses $O(r/d)$ messages and simulates Algorithm \mathcal{A} , i.e., computes the function f . Its communication complexity is $O(r/d \times d(\text{bw} + s)) = O(r(\text{bw} + s))$ qubits. \square

6.2 Proof of Theorem 3

We are now ready to give the proof of Theorem 3.

Proof of Theorem 3. Let $\{G_n\}_{n \geq 1}$ be the family of graphs that realizes the (b, k, d_1, d_2) -reduction from disjointness to diameter computation of Theorem 9. This means that we have $b = \Theta(\log n)$, $k = \Theta(n)$, $d_1 = 4$ and $d_2 = 5$. We will use below the same notations as in Definition 3.

Let d be any positive integer such that $d \leq n$. Assume that there exists an r -round quantum algorithm \mathcal{A} that decides if the diameter of a network is at most $d + d_1$ or at least $d + d_2$. For any $k \geq 0$, we will use this algorithm to construct an r -round algorithm that computes the disjointness function DISJ_k in the model described in Section 6.1 with $d + 2$ players A, P_1, \dots, P_d, B and bandwidth $\text{bw} = \Theta(b \log n)$. Let $x \in \{0, 1\}^k$ and $y \in \{0, 1\}^k$ be the inputs of nodes A and B , respectively.

Consider the bipartite graph $G_n(x, y)$, which has b edges between U_n and V_n . We modify this graph to obtain a new graph $G'_n(x, y)$ by replacing each of these b edges by a path of length $d + 1$ corresponding to d new nodes (see Figure 8 for an illustration). Note that $G'_n(x, y)$ contains $n' = n + bd$ nodes, i.e., the quantity n does not correspond anymore to the size of the network considered. This point can nevertheless be ignored since $b = \Theta(\log n)$ in our construction, which implies $n' = \Theta(n \log n)$, and since we ignore logarithmic terms in our lower bound.

The protocol works as follows. First note that the players can jointly construct the graph $G'_n(x, y)$: A can construct the subgraph induced by U_n , B can construct the subgraph induced by V_n , and the intermediate nodes can naturally be partitioned in d vertical layers constructed by P_1, \dots, P_d (see Figure 8). Nodes A, P_1, \dots, P_d, B will thus simply simulate the computation of \mathcal{A} on $G'_n(x, y)$ in r rounds. At the end of simulation A and B obtain the result of the computation by \mathcal{A} on $G'_n(x, y)$. From our assumption on \mathcal{A} , this enables them to decide whether the diameter of the network is at most $d + d_1$ or at least $d + d_2$ in r rounds and with at most s qubits of memory per node, and thus to compute $\text{DISJ}_k(x, y)$ with the same complexity. Theorem 11 therefore implies the existence of a two-party quantum protocol for DISJ_k with $O(r/d)$ rounds and total quantum communication complexity $O(r(b \log n + s))$. From Theorem 5, we conclude that

$$r(b \log n + s) = \tilde{\Omega} \left(\frac{k}{r/d} + r/d \right),$$

which implies $r = \tilde{\Omega}(\sqrt{kd/(b + s)})$. The statement of Theorem 3 directly follows, since $b = \Theta(\log n)$ and $k = \Theta(n)$. \square

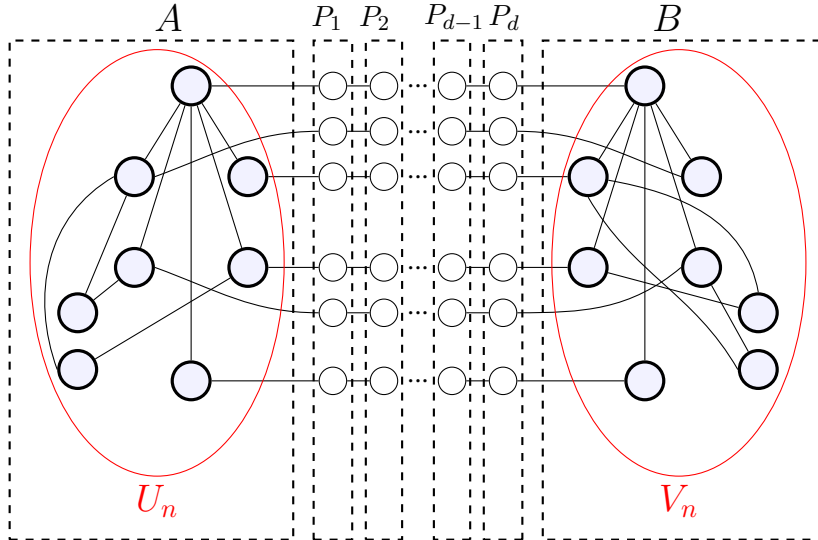


Figure 8: Example of graph $G'_n(x, y)$. Here $b = 6$: there are 6 edges between U_n and V_n in $G_n(x, y)$, and each of these edge is replaced by a path of length $d + 1$ in $G'_n(x, y)$.

Acknowledgments

The authors are grateful to Rahul Jain, Ashwin Nayak and Dave Touchette for helpful discussions, and to the anonymous referees for valuable comments and suggestions. FLG was partially supported by the JSPS KAKENHI grants No. 16H01705 and No. 16H05853. FM was partially supported by the ERA-NET Cofund in Quantum Technologies project QuantAlgo and the French ANR Blanc project RDAM.

References

- [AA05] Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005.
- [ACHK16] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC 2016)*, pages 29–42, 2016.
- [BCW98] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC 1998)*, pages 63–68, 1998.
- [Ben89] Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.
- [BGK⁺15] Mark Braverman, Ankit Garg, Young Kun-Ko, Jieming Mao, and Dave Touchette. Near-optimal bounds on bounded-round quantum communication complexity of disjointness. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 773–791, 2015.
- [BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP 1998)*, pages 820–831, 1998.

- [BK17] Karl Bringmann and Sebastian Krinninger. A note on hardness of diameter approximation. In *Proceedings of the 31st International Symposium on Distributed Computing (DISC 2017)*, pages 44:1–44:3, 2017.
- [BT08] Anne Broadbent and Alain Tapp. Can quantum mechanics help distributed computing? *SIGACT News*, 39(3):67–76, 2008.
- [DHHM06] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.
- [DP08] Vasil S. Denchev and Gopal Pandurangan. Distributed quantum computing: a new frontier in distributed systems or science fiction? *SIGACT News*, 39(3):77–95, 2008.
- [EKNP14] Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. Can quantum communication speed up distributed computation? In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing (PODC 2014)*, pages 166–175, 2014.
- [FHW12] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 1150–1162, 2012.
- [GKM09] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? In *Proceedings of the 23rd International Symposium on Distributed Computing (DISC 2009)*, pages 243–257, 2009.
- [HdW02] Peter Høyer and Ronald de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2002)*, pages 299–310, 2002.
- [HPRW14] Stephan Holzer, David Peleg, Liam Roditty, and Roger Wattenhofer. Distributed $3/2$ -approximation of the diameter. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC 2014)*, pages 562–564, 2014.
- [HW12] Stephan Holzer and Roger Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing (PODC 2012)*, pages 355–364, 2012.
- [JRS03] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A lower bound for the bounded round quantum communication complexity of set disjointness. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 220–229, 2003.
- [KS92] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.
- [LeG09] François Le Gall. Exponential separation of quantum and classical online space complexity. *Theory of Computing Systems*, 45(2):188–202, 2009.
- [LP13] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing (PODC 2013)*, pages 375–382, 2013.
- [MNRS11] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.
- [NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2011.

- [PRT12] David Peleg, Liam Roditty, and Elad Tal. Distributed algorithms for network diameter and girth. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP 2012)*, pages 660–672, 2012.
- [Raz92] Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.
- [Raz03] Alexander A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Science: Mathematics*, 67(1):159–176, 2003.
- [TKM12] Seiichiro Tani, Hirotada Kobayashi, and Keiji Matsumoto. Exact quantum algorithms for the leader election problem. *ACM Transactions on Computation Theory*, 4(1):1:1–1:24, 2012.
- [deW02] Ronald de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002.
- [Yao79] Andrew C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 209–213, 1979.
- [Yao93] Andrew C. Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1993)*, pages 352–361, 1993.