Quantum Distributed Algorithm for Triangle Finding in the CONGEST Model

Taisuke Izumi
Graduate School of Engineering, Nagoya Institute of Technology

François Le Gall
Graduate School of Mathematics, Nagoya University

Frédéric Magniez
Université de Paris, IRIF, CNRS

Abstract

This paper considers the triangle finding problem in the CONGEST model of distributed computing. Recent works by Izumi and Le Gall (PODC'17), Chang, Pettie and Zhang (SODA'19) and Chang and Saranurak (PODC'19) have successively reduced the classical round complexity of triangle finding (as well as triangle listing) from the trivial upper bound $O(n)$ to $\tilde{O}(n^{1/3})$, where n denotes the number of vertices in the graph. In this paper we present a quantum distributed algorithm that solves the triangle finding problem in $\tilde{O}(n^{1/4})$ rounds in the CONGEST model. This gives another example of quantum algorithm beating the best known classical algorithms in distributed computing. Our result also exhibits an interesting phenomenon: while in the classical setting the best known upper bounds for the triangle finding and listing problems are identical, in the quantum setting the round complexities of these two problems are now $\tilde{O}(n^{1/4})$ and $\tilde{\Theta}(n^{1/3})$, respectively. Our result thus shows that triangle finding is easier than triangle listing in the quantum CONGEST model.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum computing, distributed computing, CONGEST model

1 Introduction

Background. The problem of detecting triangles in graphs has recently become the target of intensive research by the distributed computing community [1, 5, 6, 7, 8, 9, 18, 25]. This problem comes in two main variants: the *triangle finding problem* and the *triangle listing problem*. Given as input a graph $G = (V, E)$, the triangle finding problem asks to decide[1] whether the graph contains a triangle (i.e., three vertices $u, v, w \in V$ such that $\{u, v\}, \{u, w\}, \{v, w\} \in E$), while the triangle listing problem asks to output all the triangles of G. A solution to the latter version, obviously, gives a solution to the former version. Besides its theoretical interest, another motivation for considering these problems is that for several graph problems faster distributed algorithms are known over triangle-free graphs (e.g., [16, 26]). The ability to efficiently check whether the network is triangle-free (or detect which part of the network is triangle-free) is essential when considering such algorithms.

One of the main models to study graph-theoretic problems in distributed computing is the CONGEST model. In this model the graph $G = (V, E)$ represents the topology of the network, the computation proceeds with round-based synchrony and each vertex can send one $O(\log n)$-bit message to each adjacent vertex per round, where n denotes the number of vertices. Initially, each vertex knows only the local topology of the network, i.e., the set of edges incident to itself. The triangle finding and listing problems ask, respectively, to decide

[1] Another version of the triangle finding problem asks to output one triangle of G (or report that the graph has no triangle). It is easy to see that the two versions are essentially equivalent: a triangle can be found by applying $O(\log |V|)$ times an algorithm solving the decision version.

if G contains a triangle and to list all triangles of G. The trivial strategy is for each vertex to send the list of all its neighbors to each neighbor (all the triangles can then be listed locally, i.e., without further communication). Since each list can contain up to n vertices, this requires $O(n)$ rounds in the CONGEST model.

In 2017, Izumi and Le Gall [18] gave the first nontrivial distributed algorithms for triangle detection in the CONGEST model: they constructed a $\tilde{O}(n^{2/3})$-round randomized algorithm[2] for triangle finding and a $\tilde{O}(n^{3/4})$-round randomized algorithm for triangle listing. This was soon improved by Chang, Pettie and Zhang [6], who obtained a $\tilde{O}(\sqrt{n})$-round randomized algorithm for both the triangle finding and listing problems. The key idea leading to this improvement was to decompose the graph into components with low mixing time, and then apply recent routing techniques [13, 14] that make possible to achieve efficient routing in graphs with low mixing time. The complexity of the resulting distributed algorithm was dominated by the cost required to compute the graph decomposition. Very recently, Chang and Saranurak [7] developed a much more efficient method to decompose the graph into components with low mixing time, which immediately leads to $\tilde{O}(n^{1/3})$-round randomized algorithms for the triangle finding and listing problems. Since a matching lower bound $\tilde{\Omega}(n^{1/3})$ is known for triangle listing [25], the randomized round complexity of the triangle listing problem is thus now settled, up to possible polylogarithmic factors.[3] For the triangle finding problem, on the other hand, essentially no nontrivial lower bound is known. Two exceptions are, first, the very weak lower bound obtained by Abboud, Censor-Hillel, Khoury and Lenzen [1] in the CONGEST model and, second, a lower bound obtained by Drucker, Kuhn and Oshman [9] for the much weaker CONGEST-BROADCAST model (where at each round the vertices can only broadcast a single common message to all other vertices) under a conjecture in computational complexity theory. This leads to the following intriguing question: is triangle finding easier than triangle listing?

Another related, but much stronger, model is the CONGEST-CLIQUE model. In this model at each round messages can even be sent between non-adjacent vertices, which makes bandwidth management significantly easier. In the CONGEST-CLIQUE model, Dolev, Lenzen and Peled [8] first showed that the triangle listing problem (and thus the triangle finding problem as well) can be solved deterministically in $\tilde{O}(n^{1/3})$ rounds for general graphs, which is tight since the lower bound $\tilde{\Omega}(n^{1/3})$ by Pandurangan, Robinson and Scquizzato [25] mentioned above holds in this model as well. Note that this lower bound also means that triangle listing in the CONGEST-CLIQUE model is not easier than triangle listing in the CONGEST model, at least as far as randomized algorithms are concerned. For the triangle finding problem, on the other hand, the better upper bound $O(n^{0.1572})$ has been obtained by Censor-Hillel et al. [5] by implementing fast matrix multiplication algorithms in the distributed setting. The CONGEST-CLIQUE model is thus a setting in which triangle finding is easier than triangle listing.

Table 1 summarizes the best known bounds on the round complexity of triangle finding and listing discussed so far.

Quantum distributed computing. Quantum versions of the main models studied in distributed computing can be easily defined by allowing quantum information, i.e., quantum bits (qubits), to be sent through the edges of the network instead of classical information, i.e.,

[2] In this paper the notations $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ remove poly(log(n)) factors.

[3] An interesting open problem, however, is to determine the deterministic round complexity of these problems. To our knowledge, no deterministic algorithm with sublinear round complexity is known in the CONGEST model.

Model	Setting	Problem	Complexity	Paper
CONGEST-CLIQUE	Classical	Listing	$\tilde{O}(n^{1/3})$	Dolev et al. [8]
CONGEST-CLIQUE	Classical	Finding	$O(n^{0.1572})$	Censor-Hillel et al. [5]
CONGEST	Classical	Listing	$\tilde{O}(n^{1/3})$	Chang and Saranurak [7]
CONGEST	Quantum	Finding	$\tilde{O}(n^{1/4})$	This paper
CONGEST-BROADCAST	Classical	Finding	$\Omega\big(\frac{n}{e^{\sqrt{\log n}}\log n}\big)$	Drucker et al. [9]
CONGEST-CLIQUE	Quantum	Listing	$\Omega\big(\frac{n^{1/3}}{\log n}\big)$	Pandurangan et al. [25]

Table 1 Prior results on the round complexity of distributed triangle finding and listing, and our new result. Here n denotes the number of vertices of the graph. Note that any upper bound for the listing problem (in particular, the upper bound from [7]) holds for the finding problem as well. Similarly, note that any lower bound for the quantum CONGEST-CLIQUE model (in particular, the lower bound from [25]) holds for the weaker classical and quantum CONGEST models as well.

bits. In particular, in the quantum version of the CONGEST model, which we will simply call the "quantum CONGEST model" below, each vertex can send one message of $O(\log n)$ qubits to each adjacent vertex per round. While a seminal work by Elkin et al. [10] showed that for many important graph-theoretical problems the quantum CONGEST model is not more powerful than the classical CONGEST model, Le Gall and Magniez [20] recently showed that one problem can be solved more efficiently in the quantum setting: computing the diameter of the network. More precisely, they constructed a $\tilde{O}(\sqrt{nD})$-round quantum algorithm for the exact computation of the diameter of the network (here D denotes the diameter), while it is known that any classical algorithm in the CONGEST model requires $\tilde{\Omega}(n)$ rounds, even for graphs with constant diameter [11]. In the CONGEST-CLIQUE model as well, a quantum algorithm faster than the best known classical algorithms has been obtained recently for the All-Pair Shortest Path problem [17]. In the LOCAL model, which is another fundamental model in distributed computing, separations between the computational powers of the classical and quantum versions have also been reported [12, 21].

When discussing the classical randomized complexity of triangle listing in the CONGEST and CONGEST-CLIQUE models, we mentioned the $\tilde{\Omega}(n^{1/3})$-round lower bound by Pandurangan, Robinson and Scquizzato [25]. This lower bound is based on an information-theoretic argument that actually holds even in the quantum setting. In view of the recent matching upper bound in the classical setting [7], we can conclude that for triangle listing the quantum CONGEST model is not more powerful than the classical CONGEST model. An intriguing question is whether quantum communication can help solving faster the triangle finding problem in the CONGEST model. In particular, can we break the $\tilde{O}(n^{1/3})$ barrier for triangle finding in the quantum setting?

Our result. In this paper we break this barrier. Our main result is the following theorem.

▶ **Theorem 1.** *In the quantum CONGEST model, the triangle finding problem can be solved with probability at least $1 - 1/\mathrm{poly}(n)$ in $\tilde{O}(n^{1/4})$ rounds, where n denotes the number of vertices in the network.*

In comparison, as already explained, in the classical CONGEST model the best known upper bound on the randomized round complexity of triangle finding is $\tilde{O}(n^{1/3})$. Our result thus gives another example of quantum algorithm beating the best known classical algorithms in distributed computing. It also exhibits an interesting phenomenon: while in the classical setting the best known upper bounds for the triangle finding and listing problems are both $\tilde{O}(n^{1/3})$, in the quantum setting the round complexity of the former problem becomes $\tilde{O}(n^{1/4})$ while the round complexity of the latter problem remains $\tilde{\Theta}(n^{1/3})$. Theorem 1 thus shows that triangle finding is easier than triangle listing in the quantum CONGEST model.

Overview of our approach. Our approach starts similarly to the classical algorithms developed by Chang, Pettie and Zhang [6] and Chang and Saranurak [7]. As in [6], by using an expander decomposition of the network, the triangle finding problem over the whole network is reduced to the task of detecting whether the subnetwork induced by a set of edges $E^{\text{in}} \cup E^{\text{out}}$ contains a triangle. We denote the latter problem FindTriangleInSubnetwork. Here E^{in} and E^{out} are two subsets of edges that satisfy specific conditions. In particular, the subnetwork induced by the edges in E^{in} is guaranteed to have low mixing time (but in general nothing can be said about the mixing time of the subnetwork induced by $E^{\text{in}} \cup E^{\text{out}}$). We use the algorithm from [7] to compute the expander decomposition efficiently, which implies that an efficient algorithm for FindTriangleInSubnetwork gives an efficient algorithm for the triangle finding problem over the whole network. More precisely, a $\tilde{O}(n^{1/4})$-round algorithm for FindTriangleInSubnetwork leads to a $\tilde{O}(n^{1/4})$-round algorithm for the triangle finding problem. The details of this reduction are described in Section 3.

Our main approach to solve the problem FindTriangleInSubnetwork is to apply the framework for quantum distributed search developed in [20]. We briefly sketch the main ideas. Let us write $\mathcal{V} \subseteq V$ the set of vertices of the graph induced by the edges in $E^{\text{in}} \cup E^{\text{out}}$. We will partition $\mathcal{V}$ into $t = \tilde{\Theta}(\sqrt{n})$ subsets $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_t$ each containing $\tilde{O}(\sqrt{n})$ vertices. Let us write $\Lambda = [t] \times [t] \times [t]$. We will try to find a triple $(i, j, k) \in \Lambda$ for which there exist $u \in \mathcal{V}_i$, $v \in \mathcal{V}_j$ and $w \in \mathcal{V}_k$ such that $\{u, v, w\}$ is a triangle. To do this, we partition the set Λ into t subsets $\Lambda_1, \cdots, \Lambda_t$ each containing $t^2 = \tilde{\Theta}(n)$ triples and consider the following search problem: find an index $\ell \in [t]$ such that the set Λ_ℓ contains a triple (i, j, k) for which there exist $u \in \mathcal{V}_i$, $v \in \mathcal{V}_j$ and $w \in \mathcal{V}_k$ such that $\{u, v, w\}$ is a triangle. Quantum distributed search enables us to solve this problem in $\tilde{O}(\sqrt{t}\delta)$ rounds in the quantum CONGEST model if a checking procedure (i.e., a procedure that on input ℓ checks if the set Λ_ℓ contains a triple (i, j, k) for which there exist $u \in \mathcal{V}_i$, $v \in \mathcal{V}_j$ and $w \in \mathcal{V}_k$ such that $\{u, v, w\}$ is a triangle) can be implemented in δ rounds.

The checking procedure distributes the $\tilde{\Theta}(n)$ triples in Λ_ℓ among the vertices of the network proportionally to the degree of the vertices. In particular, vertices with very low degree do not receive any triple. This technique is essentially the same as for the main procedure in the classical algorithm by Chang, Pettie and Zhang [6]. Next, each vertex owning a triple (i, j, k) checks whether there exist $u \in \mathcal{V}_i$, $v \in \mathcal{V}_j$ and $w \in \mathcal{V}_k$ such that $\{u, v, w\}$ is a triangle, which requires gathering information of all the edges with extremities in these sets. Since the subnetwork induced by the edges in E^{in} has low mixing time, we would like to use the same classical routing techniques [13, 14] as used in the main procedure of the classical algorithms [6, 7]. Special care is nevertheless required to ensure that we can apply those routing techniques. (This was not needed in [6, 7] since these prior works used a different approach: each vertex simply loaded all the necessary edges from Λ in $\tilde{O}(n^{1/3})$ rounds. In comparison, we need to guarantee that the necessary edges from Λ_ℓ, for a fixed ℓ, can be loaded in negligible time.) We solve this difficulty by carefully defining the sets Λ_ℓ so that the the same edge is not requested by two distinct vertices at the same time (this is the contents of Lemma 10 in Section 4.2).

Another technical difficulty is how to handle vertices with very high degree. In the classical setting this was trivial, since it was enough to gather in $\tilde{O}(n^{1/3})$ rounds all the information about the edges of the network at one of these high-degree vertices. Since we want to construct a $\tilde{O}(n^{1/4})$-round quantum algorithm we cannot use this approach. Instead, we develop an approach based on the well-known protocol from the two-party quantum computation complexity of the disjointness function [4]. This is explained in Section 4.1.

Other related works. The triangle finding problem is also a central problem in quantum query complexity. While many quantum query algorithms have been designed in this setting [2, 19, 22, 23], they are based on quantum techniques (e.g., quantum walk search and learning graphs) that do not seem to lead to efficient algorithms in the distributed setting.

2 Preliminaries

Graph theory. All the graphs considered in this paper are undirected and unweighted. For any graph $G = (V, E)$ and any vertex $u \in V$, we denote $\deg(u)$ the degree of u and $\mathcal{N}(u)$ the set of neighbors of u. We write $n = |V|$ and $m = |E|$. For any set $E' \subseteq E$, we denote $\deg_{E'}(u)$ the number of edges in E' incident to u and write $\mathcal{N}_{E'}$ the set of all neighbors $v \in \mathcal{N}(u)$ such that $\{u, v\} \in E'$. We denote $\mathrm{diam}(G)$ the diameter of G and $\mathrm{mix}(G)$ the mixing time of G, i.e., the number of steps of a random walk over G needed to obtain a distribution close to the stationary distribution (we refer to [13] for a precise definition). For any positive integer t, we write $[t] = \{1, 2, \ldots, t\}$.

We will use the following lemma from [6] in our main algorithm.

▶ **Lemma 2** (Lemma 4.2 in [6]). *Consider a graph with m edges and n vertices. Let p be such that $p^2 m \geq 400(\log n)^2$. Suppose that the degree of any vertex of the graph is at most $mp/(20 \log n)$. Generate a subset S by letting each vertex join S independently with probability p. Then with probability at least $1 - 1/\mathrm{poly}(n)$, the number of edges in the subgraph induced by S is at most $6p^2 m$.*

Classical distributed computing. In the classical CONGEST model, the graph $G = (V, E)$ represents the topology of the network. The computation proceeds with round-based synchrony and each vertex can send one $O(\log n)$-bit message to each adjacent vertex per round. All links (corresponding to the edges of G) are reliable and suffer no faults. Each vertex has a distinct identifier from a domain $\mathcal{I}$ with $|\mathcal{I}| = \mathrm{poly}(n)$. It is also assumed that each vertex can access an infinite sequence of local random bits. Initially, each vertex knows nothing about the topology of the network except the set of edges incident to itself and the value of n.

Our quantum algorithm will be based on several classical distributed algorithms from the literature. A first crucial ingredient is the following recent result by Chang and Saranurak [7] that shows how to efficiently compute a good expander decomposition of the graph.

▶ **Theorem 3** ([7]). *In the classical CONGEST model, there is a $O(n^{0.1})$-round algorithm that computes with probability at least $1 - 1/\mathrm{poly}(n)$ a partition*

$$V = V_1 \cup V_2 \cup \cdots \cup V_s$$

of the vertex set V that satisfies the following two conditions:

- *for each $i \in [s]$, the subgraph induced by the vertex set V_i has mixing time $O(\mathrm{poly}(\log n))$;*
- *the number of inter-component edges (i.e., the number of edges with one endpoint in V_i and one endpoint in V_j, for $i \neq j$) is at most $|E|/10$.*

We will also use the following technical lemma from [6] that shows how to compute efficiently a new ID assignment that gives a good estimation of the degree of any vertex of the graph.

▶ **Lemma 4** (Lemma 4.1 in [6]). *In the classical CONGEST model, there is a $O(\mathrm{diam}(G)+\log n)$-round deterministic algorithm that computes a bijective map $\gamma\colon V\to\{1,\dots,|V|\}$ and a function $d\colon\{1,\dots,|V|\}\to\{0,1,\dots,\lfloor\log_2(n)\rfloor\}$ satisfying the following conditions:*

(i) $\gamma(u)\le\gamma(v)$ *implies* $\lfloor\log_2(\deg(u))\rfloor\le\lfloor\log_2(\deg(v))\rfloor$ *for any* $u,v\in V$*;*

(ii) $d(\gamma(u))=\lfloor\log_2(\deg(u))\rfloor$ *for all* $u\in V$*.*

More precisely, after running the algorithm each vertex u knows $\gamma(u)$ and can locally compute $d(y)$ for any $y\in\{0,1,\dots,|V|\}$.

We will also use the following result by Ghaffari, Kuhn and Su [13] about randomized routing in networks with small mixing time (see also the discussion in Section 3 of [7]).

▶ **Theorem 5** ([13]). *In the classical CONGEST model, there exists a $O(\mathrm{mix}(G)\cdot n^{o(1)})$-round algorithm that builds a distributed data structure. This data structure enables the vertices to implement the following routing task with probability at least $1-1/\mathrm{poly}(n)$ in $O(\mathrm{mix}(G)\cdot n^{o(1)})$ rounds: given a set of point-to-point routing requests, each given by the IDs of the corresponding source-destination pair and such that each vertex u is the source and the destination of at most $O(\deg(u))$ messages, delivers all the messages.*

Quantum distributed computing. We assume that the reader is familiar with the basic concepts of quantum computation and refer to, e.g., [24] for a good reference. The quantum CONGEST model is defined (see [10, 20] for details) as the quantum version of the classical CONGEST model, where the only difference is that each exchanged message consists of $O(\log n)$ quantum bits instead of $O(\log n)$ bits. In particular, initially the vertices of the network do not share any entanglement.

For the quantum CONGEST model, Le Gall and Magniez [20] introduced a framework for quantum distributed search, which can be seen as a distributed implementation of Grover's search [15], one of the most important centralized quantum algorithms. Let X be a finite set and $f\colon X\to\{0,1\}$ be a Boolean function over X. Let u be an arbitrary vertex of the network (e.g., an elected leader). Assume that vertex u can evaluate the function f in r rounds: assume that there exists an r-round distributed checking procedure $\mathcal{C}$ such that vertex u, when receiving as input $x\in X$, outputs $f(x)$. Now consider the following problem: vertex u should find one element $x\in X$ such that $f(x)=1$ (or report that no such element exists). The trivial strategy is to compute $f(x)$ for each $x\in X$ one by one, which requires $r|X|$ rounds. Ref. [20] showed that in the quantum CONGEST model this problem can be solved with probability at least $1-1/\mathrm{poly}(|X|)$ in $\tilde{O}(r\sqrt{|X|})$ rounds.

As explained in [20], the procedure $\mathcal{C}$ is often described as a classical (deterministic or randomized) procedure. It can then be quantized using standard techniques: one first transforms it to a reversible map using standard techniques [3] and then converts it into a quantum procedure.

3 Reduction to Triangle finding over Subnetworks

In this section we present the reduction by Chang, Pettie and Zhang [6] from triangle finding over the whole network to triangle finding over subnetworks with small mixing time. In this section again, $G=(V,E)$ represents the whole network on which we want to solve the triangle finding problem, and we write $n=|V|$.

Triangle finding over subnetworks with small mixing time. We now present the computational problem considered, which we denote FindTriangleInSubnetwork (the description is also summarized in Figure 1).

FindTriangleInSubnetwork

Input: a connected subgraph $G^{\text{in}} = (V^{\text{in}}, E^{\text{in}})$ of G and a set of edges $E^{\text{out}} \subseteq E$ joining vertices in V^{in} to vertices in $V \setminus V^{\text{in}}$ (each vertex $u \in G$ knows if $u \in V^{\text{in}}$ and gets $\mathcal{N}_{E^{\text{in}}}(u)$ and $\mathcal{N}_{E^{\text{out}}}(u)$)

Promise: (i) $\text{mix}(G^{\text{in}}) = \text{poly}(\log n)$
(ii) $\deg_{E^{\text{in}}}(u) \geq \deg_{E^{\text{out}}}(u)$ for all $u \in V^{\text{in}}$

Goal: detect if there exists a triangle $\{u, v, w\}$ with $u, v, w \in V^{\text{in}} \cup V^{\text{out}}$ and $\{u, v\}, \{u, w\}, \{v, w\} \in E^{\text{in}} \cup E^{\text{out}}$

Figure 1 Problem FindTriangleInSubnetwork.

The input of FindTriangleInSubnetwork is a connected subgraph $G^{\text{in}} = (V^{\text{in}}, E^{\text{in}})$ of G such that $\text{mix}(G^{\text{in}}) = \text{poly}(\log n)$, and a set of edges $E^{\text{out}} \subseteq E$ joining vertices in V^{in} to vertices in $V \setminus V^{\text{in}}$ that satisfies the following condition:

$$\deg_{E^{\text{in}}}(u) \geq \deg_{E^{\text{out}}}(u) \text{ for all } u \in V^{\text{in}}.$$

We write V^{out} the set of vertices in $V \setminus V^{\text{in}}$ that appear as an endpoint of an edge in E^{out}. The goal is to detect if there is a triangle in the subgraph of G induced by the edge set $E^{\text{in}} \cup E^{\text{out}}$. Note that such a triangle is either a triangle of G^{in}, or consists of two vertices in V^{in} and one vertex in V^{out}. Note that, while G^{in} (the subnetwork induced by E^{in}) has small mixing time, in general nothing can be said about the mixing time of the subnetwork induced by $E^{\text{in}} \cup E^{\text{out}}$.

The reduction. Chang, Pettie and Zhang [6] proved that when a good expander decomposition of the network is known, triangle finding over the whole network can be efficiently reduced to solving several instances of FindTriangleInSubnetwork. Combined with Theorem 3, this gives an efficient reduction from triangle finding to FindTriangleInSubnetwork, which we state in the following theorem. For completeness we include a sketch of the proof (we refer to [6, 7] for the details).

▶ **Theorem 6** ([6, 7]). *Assume that there exists an r-round distributed algorithm $\mathcal{A}$ that solves the problem* FindTriangleInSubnetwork *with probability at least $1-1/n^3$ and uses only the edges in $E^{in} \cup E^{out}$ for communication. Then there exists a $O(r \log n + n^{0.1})$-round distributed algorithm that solves the triangle finding problem over the whole graph $G = (V, E)$ with probability at least $1 - 1/\text{poly}(n)$.*

Sketch of the proof. The first step of the reduction computes a good decomposition of the whole network G in $O(n^{0.1})$ rounds using Theorem 3. Let $V = V_1 \cup V_2 \cup \cdots \cup V_s$, for some integer $s \leq n$, denote the decomposition and E^{inter} denote the set of inter-component edges. By definition, the set E^{inter} satisfies $|E^{\text{inter}}| \leq 0.1|E|$.

For any index $i \in [s]$, let us write $G_i = (V_i, E_i)$ the subgraph of G induced by V_i. We say that a vertex $u \in V_i$ is good if $\deg_{E_i}(u) \geq \deg_{E^{\text{inter}}}(u)$; otherwise we say that u is bad. Let E_i^{inter} be the set of edges in E^{inter} that are adjacent to a good vertex of G_i. Let E_i^{new} be the set of edges in E_i that are adjacent to a bad vertex of G_i. Define the set

$$E^{\text{new}} = E^{\text{inter}} \cup E_1^{\text{new}} \cup \cdots \cup E_s^{\text{new}}$$

and observe that $|E^{\text{new}}| \leq |E^{\text{inter}}| + 2|E^{\text{inter}}| \leq 0.3|E|$.

The triangles in G can be classified into the following four types.

- Type 1: triangles with three vertices in a same component G_i.
- Type 2: triangles with two vertices in a same component G_i and the third vertex in another component G_j, in which the two vertices in V_i are good.
- Type 3: triangles with two vertices in a same component G_i and the third vertex in another component G_j, in which at least one of the two vertices in V_i is bad.
- Type 4: triangles with three vertices in distinct components.

For each index $i \in [s]$, we use Algorithm $\mathcal{A}$ to solve FindTriangleInSubnetwork on instance $(G^{\text{in}}, E^{\text{out}})$ with $G^{\text{in}} = G_i$ and $E^{\text{out}} = E_i^{\text{inter}}$. A crucial point is that this can be done for all i's in parallel by "doubling" the bandwidth (i.e., by using $2r$ rounds in total), since Algorithm $\mathcal{A}$ on instance $(G^{\text{in}}, E^{\text{out}})$ only uses the edges in $E_i \cup E_i^{\text{inter}}$ for communication. Indeed, the communication networks are disjoint for all instances, except for the intercomponent edges that can be shared by two instances. This detects all the triangles of types 1 and 2 in the graph.

Another crucial observation is that all remaining potential triangles (i.e., the triangles of type 3 and 4) have their three edges contained in the set E^{new}. It is thus enough to recurse on this set, i.e., to repeat the same methodology with E replaced by E^{new}. Since $|E^{\text{new}}| \leq 0.3|E|$, after $O(\log n)$ levels of recursion the algorithm finishes. The overall complexity of this second part is thus $O(r \log n)$ rounds. ◀

4 Main Quantum Algorithm

In the classical CONGEST model, Chang, Pettie and Zhang [6] have shown that the problem FindTriangleInSubnetwork can be solved with high probability in $\tilde{O}(n^{1/3})$ rounds using only the edges in $E^{\text{in}} \cup E^{\text{out}}$ for communication, which leads to a $\tilde{O}(n^{1/3})$-round classical algorithm for triangle finding via Theorem 6. Our main technical result is the following theorem.

▶ **Theorem 7.** *In the quantum CONGEST model, there is a $\tilde{O}(n^{1/4})$-round quantum algorithm that solves the problem* FindTriangleInSubnetwork *with probability at least $1 - 1/n^3$ and uses only the edges in $E^{in} \cup E^{out}$ for communication.*

Theorem 1 then immediately follows from Theorem 6 and Theorem 7.

The goal of this section is to prove Theorem 7. For brevity we write $\mathcal{V} = V^{\text{in}} \cup V^{\text{out}}$ and $\mathcal{E} = E^{\text{in}} \cup E^{\text{out}}$. We also write $\bar{n} = |\mathcal{V}|$ and $\bar{m} = |\mathcal{E}|$. Note that $\bar{m} \geq \bar{n}/2$ since the graph $(\mathcal{V}, \mathcal{E})$ is connected. Let $S \subseteq \mathcal{V}$ be the subset of all vertices $u \in \mathcal{V}$ such that

$$\deg_{\mathcal{E}}(u) \geq \bar{m}/\sqrt{\bar{n}}.$$

Observe that $|S| \leq 2\sqrt{\bar{n}}$.

In Section 4.1 below we present a $\tilde{O}(n^{1/4})$-round quantum algorithm that detects the existence of a triangle containing at least one vertex from S. We then describe, in Section 4.2, our main technical contribution: a $\tilde{O}(n^{1/4})$-round quantum algorithm that detects the existence of a triangle under the assumption $S = \emptyset$. The quantum algorithm of Theorem 7 then follows by combining these two algorithms, since (as already observed in prior works [6, 7]) detecting whether there exists a triangle with no vertex in S reduces to the case $S = \emptyset$.

Let us provide some explanations about why detecting the existence of a triangle with no vertex in S reduces to the case $S = \emptyset$. One natural approach is to focus on the subgraph where all the nodes in S are removed. This approach nevertheless does not immediately work since this may make the graph disconnected and may significantly reduce the number of edges (in which case Lemma 2 may not anymore be applicable). Instead, we now briefly describe a method that keeps the graph connected and the number of edges (almost) unchanged. The idea is simply to "virtually" replace each node $u \in S$ by a star of degree $\sqrt{\bar{n}}$ and spread the $\deg_{\mathcal{E}}(u)$ incident edges of u evenly into the leaves of the star so that each leaf has $\deg_{\mathcal{E}}(u)/\sqrt{\bar{n}} < \bar{m}/\sqrt{\bar{n}}$ incident edges. Since $|S| \leq 2\sqrt{\bar{n}}$, this can be done by introducing only $\Theta(\bar{n})$ virtual nodes.

4.1 Finding a triangle containing (at least) one high-degree vertex

In this subsection we describe how to detect, in $\tilde{O}(n^{1/4})$ rounds, the existence of a triangle with edges in $\mathcal{E}$ that contains at least one vertex from S. We will use the following lemma, which is a straightforward application of the framework for distributed quantum search described in Section 2, but can also be seen as an adaptation of the quantum protocol by Buhrman, Cleve and Wigderson [4] for the disjointness function in two-party quantum communication complexity.

► **Lemma 8.** *Consider any two adjacent vertices u and v, each owning a set $T_u \subseteq V$ and a set $T_v \subseteq V$, respectively. There is a quantum algorithm that checks if $T_u \cap T_v \neq \emptyset$ with high probability in $\tilde{O}(\sqrt{\min\{|T_u|, |T_v|\}})$ rounds. Moreover, this algorithm only uses communication along the edge $\{u, v\}$.*

Proof. Consider the subnetwork consisting only of the two vertices u and v and the edge $\{u, v\}$. Set $X = T_u$ and define the function

$$f(x) = \begin{cases} 1 & \text{if } x \in T_v, \\ 0 & \text{otherwise,} \end{cases}$$

for any $x \in X$. Obviously, for any $x \in X$, vertex u can compute the value $f(x)$ in 2 rounds. We can thus apply the quantum distributed search framework of Section 2 with vertex u acting as a leader, which enables u to check whether there exists $x \in X$ such that $x \in T_v$ in $\tilde{O}(\sqrt{|T_u|})$ rounds.

By symmetry there also exists a quantum algorithm that enables vertex v to decide whether $T_u \cap T_v \neq \emptyset$ in $\tilde{O}(\sqrt{|T_v|})$ rounds. Combining these two algorithms gives the claimed complexity. ◄

We now explain how our quantum algorithm works. First of all, observe that since each vertex u receives as input $\mathcal{N}_{E^{\text{in}}}(u)$ and $\mathcal{N}_{E^{\text{out}}}(u)$, each vertex knows whether it is in S or not. Each vertex first tells this to all its neighbors. This requires 1 round of communication.

Each vertex $u \in \mathcal{V}$ then computes, locally, the set

$$T_u = \mathcal{N}_{\mathcal{E}}(u) \cap S = (\mathcal{N}_{E^{\text{in}}}(u) \cup \mathcal{N}_{E^{\text{out}}}(u)) \cap S.$$

Note that for each edge $\{u, v\} \in \mathcal{E}$, there exists $w \in S$ such that $\{u, v, w\}$ is in a triangle with three edges in $\mathcal{E}$ if and only if $T_u \cap T_v \neq \emptyset$. Thus, for each edge $\{u, v\} \in \mathcal{E}$, the vertices u and v use the quantum algorithm of Lemma 8 to decide whether $T_u \cap T_v \neq \emptyset$ or not. Since this algorithm only communicates through the edge $\{u, v\}$, it can be applied in parallel to all edges $\{u, v\} \in \mathcal{E}$. This gives overall round complexity $\tilde{O}(\sqrt{|S|}) = \tilde{O}(n^{1/4})$.

4.2 Finding a triangle with only low-degree vertices

In this subsection we assume that the inequality

$$\deg_{\mathcal{E}}(u) < \bar{m}/\sqrt{\bar{n}}.$$

holds for all vertices $u \in \mathcal{V}$, i.e., we assume that $S = \emptyset$. We show how to detect, in $\tilde{O}(n^{1/4})$ rounds, the existence of a triangle with edges in $\mathcal{E}$ in this case as well.

Partitioning the set $\mathcal{V}$. Let us write $t = \left\lfloor \sqrt{\bar{n}}/(30 \log \bar{n}) \right\rfloor$. We randomly partition the set $\mathcal{V}$ into t subsets $\mathcal{V}_1, \ldots, \mathcal{V}_t$ as follows: each vertex $u \in \mathcal{V}$ selects an integer i uniformly at random in the set $[t]$ and joins the set $\mathcal{V}_i$. Vertex $u \in \mathcal{V}$ then tells its neighbors the value i, which can be done in 1 round. Each vertex therefore learns in which sets its neighbors have been included.

For any $i, j \in [t]$, let $E(\mathcal{V}_i, \mathcal{V}_j)$ denote all the edges in $\mathcal{E}$ with one endpoint in $\mathcal{V}_i$ and one endpoint in $\mathcal{V}_j$. Our analysis will rely on the following lemma.

▶ **Lemma 9.** *With probability* $1 - 1/\mathrm{poly}(n)$*, the following statement is true: for all* $i, j \in [t]$*,*

$$|E(\mathcal{V}_i, \mathcal{V}_j)| = O\left(\frac{\bar{m}(\log \bar{n})^2}{\bar{n}}\right).$$

Proof. Let us first consider the case $i = j$. We apply Lemma 2 over the graph generated by the vertex set $\mathcal{V}$ and using the probability $p = 1/t$. Note that

$$p^2 \bar{m} = \frac{\bar{m}}{(\left\lfloor \sqrt{\bar{n}}/(30 \log \bar{n}) \right\rfloor)^2} \geq \frac{\bar{n}/2}{(\sqrt{\bar{n}}/(30 \log \bar{n}))^2} \geq 400(\log \bar{n})^2$$

and

$$\frac{\bar{m}p}{20 \log \bar{n}} = \frac{\bar{m}}{20 \log \bar{n} \left\lfloor \sqrt{\bar{n}}/(30 \log \bar{n}) \right\rfloor} \geq \frac{\bar{m}}{\sqrt{\bar{n}}},$$

which implies that the two conditions in Lemma 2 are satisfied.

In the case $i \neq j$, we apply Lemma 2 over the graph generated by the vertex set $\mathcal{V}$ again, but using the probability $p = 2/t$. The conclusion is the same. ◀

Partitioning the triples of indices. Let us write Λ the set of all triples (i, j, k) with $i, j, k \in [t]$, i.e., $\Lambda = [t] \times [t] \times [t]$. Let us partition this set into t sets $\Lambda_1, \ldots, \Lambda_t$, each containing t^2 triples, as follows. For each $\ell \in [t]$, define the set Λ_ℓ as:

$$\Lambda_\ell = \big\{(i, j, 1 + (i + j + \ell \bmod t)) \mid (i, j) \in [t] \times [t]\big\}.$$

Our analysis will rely on the following lemma, which immediately follows from the definition of the sets Λ_ℓ.

▶ **Lemma 10.** *The following statements are true for all* $\ell \in [t]$ *and all triples* $(i, j, k) \in \Lambda_\ell$*:*
- *there is no index* $i' \in [t] \setminus \{i\}$ *such that* $(i', j, k) \in \Lambda_\ell$*;*
- *there is no index* $j' \in [t] \setminus \{j\}$ *such that* $(i, j', k) \in \Lambda_\ell$*;*
- *there is no index* $k' \in [t] \setminus \{k\}$ *such that* $(i, j, k') \in \Lambda_\ell$*.*

Assigning the triples to vertices. For each $\ell \in [t]$, we assign the $t^2 = \Theta(\bar{n}/(\log \bar{n})^2)$ triples in Λ_ℓ to the vertices in V^{in}. The assignment should be made carefully, so that each vertex is assigned a number of triples proportional to its degree and, additionally, all the vertices know to which vertex each triple in Λ_ℓ is assigned. To achieve this goal we use the same approach as in [6], which is based on the ID assignment of Lemma 4.

We first apply Lemma 4 to the subnetwork G^{in} in order to obtain an ID assignment $\gamma\colon V^{\mathrm{in}} \to \{1, \ldots, |V^{\mathrm{in}}|\}$ and the degree estimator function

$$d\colon \{1, \ldots, |V^{\mathrm{in}}|\} \to \{0, 1, \ldots, \lfloor \log_2 |V^{\mathrm{in}}| \rfloor\}$$

satisfying the properties in the lemma. This requires $O(\mathrm{diam}(G^{\mathrm{in}}) + \log n) = O(\mathrm{mix}(G^{\mathrm{in}}) + \log n) = \mathrm{poly}(\log n)$ rounds. For any vertex $u \in V^{\mathrm{in}}$, define the quantity

$$r_u = \frac{2^{d(\gamma(u))}}{\bar{m}/\bar{n}}.$$

Note that since $d(\gamma(u)) = \lfloor \log_2(\deg_{E^{\mathrm{in}}}(u)) \rfloor$, the quantity r_u is an approximation of the ratio between $\deg_{E^{\mathrm{in}}}(u)$ and the average degree of the subgraph $(\mathcal{V}, \mathcal{E})$. Observe that

$$\sum_{u \in V^{\mathrm{in}}} r_u \geq \sum_{u \in V^{\mathrm{in}}} \frac{\deg_{E^{\mathrm{in}}}(u)/2}{\bar{m}/\bar{n}} = \frac{|E^{\mathrm{in}}|}{\bar{m}/\bar{n}} \geq \bar{n}/2,$$

since $|E^{\mathrm{in}}| \geq \bar{m}/2$. Now define the quantity

$$q_u = \begin{cases} 0 & \text{if } r_u \leq 1/4, \\ \lceil r_u \rceil & \text{otherwise,} \end{cases}$$

and observe that

$$\sum_{u \in V^{\mathrm{in}}} q_u \geq \sum_{u \in V^{\mathrm{in}}} r_u - \frac{|V^{\mathrm{in}}|}{4} \geq \frac{\bar{n}}{4} \geq t^2. \tag{1}$$

We can now explain the assignment of the triples from Λ_ℓ. We fix an arbitrary order (known to all the vertices of the network) on the triples of each Λ_ℓ. For concreteness, let us choose the lexicographic order. We start by assigning to the vertex $u_1 \in V^{\mathrm{in}}$ such that $\gamma(u_1) = 1$ the first q_{u_1} triples of Λ_ℓ in the lexicographic order, then assign to the vertex $u_2 \in V^{\mathrm{in}}$ such that $\gamma(u_2) = 2$ the next q_{u_2} triples from Λ_ℓ in the lexicographic order, and repeat the process until all the triples of Λ_ℓ have been assigned (Equation (1) guarantees that all triples are assigned by this process). For each vertex $u \in V^{\mathrm{in}}$, let us write $\Lambda_\ell^u \subseteq \Lambda_\ell$ the set of triples assigned to u.

A crucial observation is that each vertex of the network can locally compute, for any $\ell \in [t]$ and any triple $(i, j, k) \in \Lambda_\ell$, the ID of the vertex to which (i, j, k) is assigned, since each vertex knows the value $d(y)$ for all $y \in \{0, 1 \ldots, |V^{\mathrm{in}}|\}$.

Description of the quantum search algorithm. Consider the function

$$f \colon [t] \to \{0, 1\}$$

defined as follows. For any $\ell \in [t]$, we have $f(\ell) = 1$ if and only if there exists a triple $(i, j, k) \in \Lambda_\ell$ such that the graph G has a triangle with one vertex in the set $\mathcal{V}_i$, one vertex in $\mathcal{V}_j$, one vertex in $\mathcal{V}_k$ and its three edges in $\mathcal{E}$. Our quantum algorithm implements the quantum distributed search framework described in Section 2 with $X = [t]$ to detect if there exists one index $\ell \in [t]$ such that $f(\ell) = 1$. This approach obviously detects the existence of a triangle with three edges in $\mathcal{E}$, i.e., it solves our problem. The complexity of this approach, as explained in Section 2, is $\tilde{O}(\sqrt{t}\delta) = \tilde{O}(n^{1/4}\delta)$ rounds, where δ is the round complexity of the checking procedure. We present below a checking procedure with round complexity $\delta = \tilde{O}(\mathrm{mix}(G^{\mathrm{in}}))$. Since $\mathrm{mix}(G^{\mathrm{in}}) = \mathrm{poly}(\log n)$ from our assumption on G^{in}, the overall round complexity is $\tilde{O}(n^{1/4})$, as claimed.

Description of the checking procedure. We now describe a classical randomized checking procedure that enables the leader, on an input $\ell \in [t]$, to evaluate the value $f(\ell)$. As explained in Section 2 such a classical procedure can then be converted into a quantum procedure using standard techniques. In the checking procedure, the leader first broadcasts the information "ℓ" to all the vertices of the network. This can be done in $\mathrm{diam}(G^{\mathrm{in}}) \leq \mathrm{mix}(G^{\mathrm{in}})$ rounds. Then each vertex $u \in V^{\mathrm{in}}$ checks, for each $(i,j,k) \in \Lambda_\ell^u$, whether there exists a triangle with one vertex in $\mathcal{V}_i$, one vertex in $\mathcal{V}_j$, one vertex in $\mathcal{V}_k$ with three edges in $\mathcal{E}$. In order to do that, vertex u simply needs to collect all the edges in $E(\mathcal{V}_i,\mathcal{V}_j) \cup E(\mathcal{V}_i,\mathcal{V}_k) \cup E(\mathcal{V}_j,\mathcal{V}_k)$ for each $(i,j,k) \in \Lambda_\ell^u$. From Lemma 9 and from the definition of the set Λ_ℓ, this requires

$$\tilde{O}\left(\frac{\bar{m}}{\bar{n}} \times q_u\right) = \tilde{O}\left(\frac{\bar{m}}{\bar{n}} \times \lfloor r_u \rfloor\right) = \tilde{O}\left(\deg_{E^{\mathrm{in}}}(u)\right)$$

incoming messages. Conversely, let us consider the number of outgoing messages needed to gather the information about the edges. Lemma 10 guarantees that the information about each edge only needs to be communicated to one vertex, which implies that each vertex u is the source of $\deg_{E^{\mathrm{in}}}(u)$ messages. Theorem 5 thus implies that the checking procedure can be implemented in $O(\mathrm{mix}(G^{\mathrm{in}}) \cdot n^{o(1)})$ rounds. The leader then checks if one of the vertices in V^{in} found a triangle, which can be done in $O(\mathrm{diam}(G^{\mathrm{in}})) = O(\mathrm{mix}(G^{\mathrm{in}}))$ rounds.

In order to reduce the complexity from $O(\mathrm{mix}(G^{\mathrm{in}}) \cdot n^{o(1)})$ to $\tilde{O}(\mathrm{mix}(G^{\mathrm{in}}))$ we simply need to modify slightly the routing scheme from [13], exactly as done in the classical case (see Section 3 of [7]).

Acknowledgments

The authors are grateful to anonymous reviewers for helpful comments. TI was partially supported by JST SICORP grant No. JPMJSC1606 and JSPS KAKENHI grant No. JP19K11824. FLG was supported by JSPS KAKENHI grants Nos. JP15H01677, JP16H01705, JP16H05853, JP19H04066 and by the MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) grant No. JPMXS0118067394. FM was partially supported by the ERA-NET Cofund in Quantum Technologies project QuantAlgo and the French ANR Blanc project QuData.

References

1 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling views: A new lower bound technique for distributed computations under congestion. *CoRR*, abs/1711.01623, 2017. `arXiv:1711.01623`.

2 Aleksandrs Belovs. Span programs for functions with constant-sized 1-certificates: extended abstract. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 77–84, 2012.

3 Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.

4 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 63–68, 1998.

5 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. *Distributed Computing*, 32(6):461–478, 2019.

6 Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840, 2019.

7 Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 66–73, 2019.

8 Danny Dolev, Christoph Lenzen, and Shir Peled. “Tri, Tri Again": Finding triangles and small subgraphs in a distributed setting - (extended abstract). In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 195–209, 2012.

9 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.

10 Michael Elkin, Hartmut Klauck, Danupon Nanongkai, and Gopal Pandurangan. Can quantum communication speed up distributed computation? In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 166–175, 2014.

11 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1150–1162, 2012.

12 Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. What can be observed locally? In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 243–257, 2009.

13 Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 131–140, 2017.

14 Mohsen Ghaffari and Jason Li. New distributed algorithms in almost mixing time via transformations from parallel algorithms. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 31:1–31:16, 2018.

15 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996.

16 Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *The Electronic Journal of Combinatorics*, 24(4):P4.21, 2017.

17 Taisuke Izumi and François Le Gall. Quantum distributed algorithm for the All-Pairs Shortest Path problem in the CONGEST-CLIQUE model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 84–93, 2019.

18 Taisuke Izumi and François Le Gall. Triangle finding and listing in CONGEST networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 381–389, 2017.

19 François Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 216–225, 2014.

20 François Le Gall and Frédéric Magniez. Sublinear-time quantum computation of the diameter in CONGEST networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 337–346, 2018.

21 François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. Quantum advantage for the LOCAL model in distributed computing. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 49:1–49:14, 2019.

22 Troy Lee, Frédéric Magniez, and Miklos Santha. Improved quantum query algorithms for triangle finding and associativity testing. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1486–1502, 2013.

23 Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007.

24 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2011.

25 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. On the distributed complexity of large-scale graph computations. In *Proceedings of the Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 405–414, 2018.

26 Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015.