THE QUANTUM QUERY COMPLEXITY OF TRIANGLE DETECTION IN THE ADJACENCY LIST MODEL

AMIN GILANI *, BLAKE HOLMAN †, JOHN KALLAUGHER ‡, AND OJAS PAREKH §

Abstract. The problem of determining whether a graph G on N nodes admits a triangle is one of the most well-studied graph problems in both the classical and quantum settings. Generally, algorithms are allowed to make queries to learn 1) the ith neighbor of a node, 2) the degree of a node, and 3) whether two nodes are connected. In the classical setting, almost all work on triangle-related problems uses all three types of queries. In the quantum setting, however, work has also been done using only queries about the connectivity between nodes. In this work, we take steps toward understanding the quantum query complexity of detecting whether G contains a triangle. In particular, we show that the quantum query complexity determining whether a node participates in a triangle is $\Theta(N)$, and the quantum query complexity of determining whether an edge participates in a triangle is $\Theta(N^{2/3})$.

1. Introduction. Three nodes u, v, w in a graph $G = (V, E)$ form a triangle if they are all adjacent in G. The problems of detecting, finding, counting, and listing triangles are well studied in a variety of classical settings [4, 5, 7, 10, 14] and have applications in networking, social systems, and bioinformatics [1, 3, 9, 11]. Classical algorithms for triangle-related problems generally use a specific model for accessing the graph, which allows for three types of queries:

1. *neighborhood queries* which take a node v and an index i and outputs the ith neighbor of v,
2. *degree queries* which take a node v and output the degree of v denoted d_v, and
3. *pair or edge queries* which take two nodes u and v and output whether they are connected.

The access model associated with only neighborhood and degree queries is called the *(adjacency) list model*, while the access model with only pair queries is called the *(adjacency) matrix model*. We call the query model with access to all three types of queries the *augmented list model*. The augmented list model is the most common for triangle problems. In the augmented list model Eden, Levi, Ron, and Seshadhri [5] give an algorithm which achieves the optimal query complexity of $\tilde{O}\left(\frac{N}{T^{1/3}} + \min\{\frac{m^{3/2}}{T}, \sqrt{m}\}\right)$ (here, the tilde hides polylog factors and polynomial dependency on the error parameter ε). The reason for this is that triangle problems are classically intractable in the list and matrix query models. A simple argument shows that even approximate triangle counting in the matrix model requires $\Omega(N^2)$ queries, and in the list model there is no sublinear query algorithm for approximately counting triangles [6].

Quantum algorithms can query the graph oracles on superpositions of inputs. As a result, quantum algorithms for triangle counting in the matrix and list models do have sublinear query complexity. The quantum query complexity of triangle finding in the matrix model is $\tilde{O}(N^{5/4})$, which can be easily modified to be a triangle counting algorithm with the same quantum query complexity. In the augmented list model, the quantum query complexity of triangle counting is $\tilde{O}\left(\frac{\sqrt{N}}{T^{1/6}} + \frac{m^{3/4}}{\sqrt{T}}\right)$. While triangle problems in the matrix and augmented list models have been well studied, prior to this work nothing was known about the quantum query complexity of triangle problems in the list model.

*University of Maryland, asgilani@umd.edu
†Purdue University, holman14@purdue.edu,
‡Sandia National Laboratories, jmkall@sandia.gov
§Sandia National Laboratories, odparek@sandia.gov

1.1. Our Contributions. We make progress towards characterizing the quantum query complexity of triangle problems in the list model. There are several challenges in working with the list model. For example, with pair queries it only takes 3 queries to determine whether some nodes u, v, and w form a triangle. However, in the list model, we must search the neighbors of these nodes to determine whether they are all connected. First, we consider the problem of determining whether a node participates in a triangle. We give an algorithm that uses $O(N)$ queries and a matching lower bound.

THEOREM 1.1. *In the adjacency list model, the quantum query complexity of determining whether a node in a graph on N nodes participates in a triangle is $\Theta(N)$.*

Next, we consider the problem of determining whether an edge $\{u, v\}$ participates in a triangle. The upper bound is simple, as it reduces to determining whether the neighborhoods of u and v are disjoint, which is well studied in the quantum setting [2]. The lower bound is considerably more difficult than the previous problem. We give a reduction from the *Element Distinctness Problem*, which gives a function f and asks whether f is one-to-one.

THEOREM 1.2. *In the adjacency list model, the quantum query complexity of determining whether an edge in a graph on N nodes participates in a triangle is $\Theta(N^{2/3})$.*

2. Preliminaries. In this section, we give the background and notation on graphs and their quantum query models.

2.1. Graphs and Query Models. Let $G = (V, E)$ be a graph on N nodes and m edges. We always use the convention that $V = \{0, \ldots, N-1\}$. A vertex u is a neighbor of v if u and v are connect in G, and we let $\Gamma(v)$ denote the neighbors of v. Additionally, for any $S \subset V$, we let $\Gamma(S) = \bigcup_{v \in S} \Gamma(v)$. Additionally, we let $\Gamma(v, i)$ to be the ith neighbor of v according to an arbitrary but fixed order for each vertex. The degree of a vertex v is denoted $d_v = |\Gamma(v)|$ and we let $d_{\max} = \max_{v \in V} d_v$.

There are three main query models which allow us to gain information about G in many different ways. In *adjacency matrix model* (or just matrix model), we have access to an oracle which, given nodes u and v, tell us whether or not $\{u, v\} \in E$. In the *adjacency list model* we have access to two oracles: a degree oracle and a neighborhood oracle. The degree oracle takes a node v and returns its degree d_v. The neighborhood oracle takes in a vertex v and an index i, returning the ith (starting from zero) neighbor with respect to an arbitrary but fixed order. If $i \geq d_v$, then the oracle returns a special symbol $\perp$. The last query model is the *augmented adjacency list model* which is simply the union of the matrix and adjacency list model.

2.2. Quantum Computation. A classical bit is either in the state 0 or the state 1. We can represent these states via a vector such that the zero state is $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the one state is $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. A qubit, however can be in any state of the form

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

for $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. In order to gain information about $|\psi\rangle$, we must *measure* ψ. Upon measuring, $|\psi\rangle$ collapses to $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. We will also use $\langle\psi|$ to represent $|\psi\rangle^\dagger$ and $\langle\psi|\phi\rangle$ to represent $\langle\psi| |\phi\rangle$.

More generally, for a bit-string $s = s_1, \ldots, s_n \in \{0, 1\}^n$, we can represent s as a vector v of length 2^n, where $v_s = 1$ (here we are indexing with respect to the integer corresponding

to s) and all other entries are zero. Equivalently,

$$v = |s\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_n\rangle ,$$

where $\otimes$ denotes the tensor product between two vectors. Intuitively, writing v as a tensor product of single-qubit states allows us to view v as the bit-string it represents. Additionally, we use the following equivalent notation:

$$|a\rangle \otimes |b\rangle = |a\rangle\,|b\rangle = |a, b\rangle = |ab\rangle .$$

A quantum circuit acting on n qubits can be represented as a $2^n \times 2^n$ matrix U. Quantum circuits are inherently reversible, meaning that after applying U to some state $|\phi\rangle$, there exists a quantum circuit U^{-1} which undoes the effect of U. Moreover, due to the natural rules of measurement above, it is clear that U must maintain the 2-norm of $|\psi\rangle$. We call such matrices *unitary*. Since U is unitary, it can be shown that $U^\dagger = U^{-1}$.

We make use of the following quantum subroutine. Given quantum query access to a function $f : [N] \to \{0, 1\}$, Grover's search algorithm [8] can find a marked item x (meaning $f(x) = 1$) or report that none exist using $O(\sqrt{N})$ queries to f (as opposed to classical algorithms which require $\Omega(N)$ queries).

3. Quantum Algorithms for Triangle Problems in the Adjacency List Model. The first problem we consider is determining whether a vertex v participates in a triangle. In the matrix model, this can be done straightforwardly using Grover's search algorithm to find pairs of nodes $u, w \in V$ in which u, v, and w are all adjacent in G, using this uses $O(N)$ queries. In the adjacency list model, however, we cannot simply check whether pairs of vertices are connected in constant time. We give an $O(N)$ algorithm for the problem.

LEMMA 3.1. *The problem of determining whether a node participates in a triangle in a graph on N nodes has quantum query complexity $O(N)$.*

Consider the following method:

1. Query the neighborhood oracle to find the neighborhood $u_1, \ldots, u_{d_v}$ of v.
2. Search over the neighbors of u_i to find w which is also a neighbor of v.

The first step costs $O(N)$ queries, as we are classically recording each of v's neighbors. Since we have stored the neighbors of v, it doesn't take any additional queries to determine whether a neighbor w of u_i is a member of $\Gamma(v)$. So, the query complexity of the second step is $O(\sqrt{|\Gamma(\Gamma(v)))|}) = O(N)$, resulting in Lemma 3.1.

Next, we consider the query complexity of determining whether an edge $\{u, v\}$ admits a triangle. Again this can be done in the adjacency model by searching for a node w which is connected to both u and v in $O(\sqrt{N})$ queries. We show that in the adjacency model we can determine whether a vertex v forms a triangle with u and v or report that none exists using $O(N^{2/3})$ quantum queries.

LEMMA 3.2. *The problem of determining whether an edge participates in a triangle in a graph on N nodes has quantum query complexity $O(N^{2/3})$.*

This problem is similar to the *element distinctness problem.*

DEFINITION 3.3. *Let* ELEMENTDISTINCTNESS *be the problem of determining whether a function $f : [N] \to [M]$ with $M = \Theta(N)$ is one-to-one.*

Ambainis gave a quantum algorithm that solves element distinctness using $O\left(N^{2/3}\right)$ queries [2]. In our case, we have an edge $\{u, v\}$ and a list consisting of the entries (with duplicates) of $\Gamma(u)$ and $\Gamma(v)$ of size at most $2d_{\max}$. The proof of Lemma 3.2 follows by applying the element distinctness algorithm to determine whether the edge is a triangle edge.

4. Lower bounds in the Adjacency List Model. In this section, we give lower bounds for triangle-related problems in the adjacency list model.

4.1. Triangle Vertex Determination. First, we consider the problem of deciding whether a vertex participates in a triangle. We give a reduction from the problem of deciding whether a bitstring of length M contains a one, which is known to require $\Omega(\sqrt{M})$ quantum queries. In the previous section, we gave an algorithm that tests whether a node is a triangle node using $O(N)$ queries.

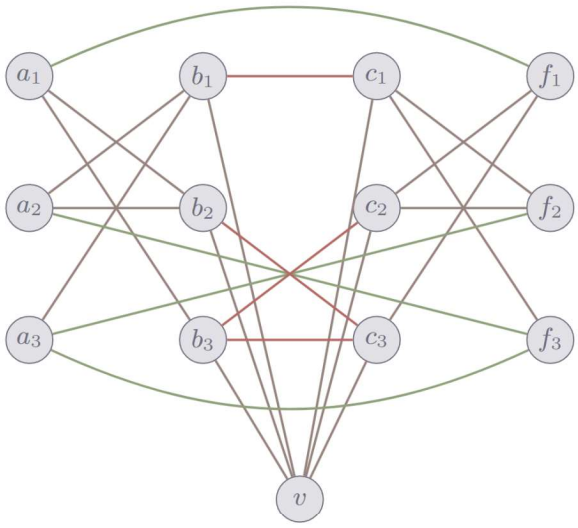

FIG. 4.1. *Above is the graph associated with the string* $x' = 10011$ *from Lemma 4.1. The associated matrix is* $X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$. *Green edges between* a_i *and* f_j *and red edges between* b_i *and* c_j *correspond to the case where* $X_{ij} = 1$. *The red edges create the triangle* $b_i c_j v$.

LEMMA 4.1. *In the adjacency list model, determining whether a node* v *in a graph* G *is part of a triangle requires* $\Omega(N)$ *queries.* .

Proof. Given some $x' \in \{0,1\}^M$, let N be the smallest integer such that $\binom{N}{2} \geq M$ and let $x = x' \| 0^{\binom{N}{2}-M}$, where $\|$ denotes string concatenation. Now define the symmetric $N \times N$ matrix X which encodes two copies of x in the following way. First, we construct the upper triangular matrix X' which takes the values x_i in the natural order. We let $X = X' + X'^T - \text{diag}(X')$. Next, we construct a graph $G = (V, E)$ on $4N + 1$ nodes. Let $A = \{a_1, \ldots, a_N\}$, $B = \{b_1, \ldots, b_N\}$, $C = \{c_1, \ldots, c_N\}$, and $F = \{f_1, \ldots, f_N\}$. Finally, we let $\mathrm{V} = A \cup B \cup C \cup F \cup \{v\}$. For each $u \in B \cup C$, $\{u, v\}$ is an edge in G. Next, if $X_{ij} = 1$ then $\{a_i, f_j\}, \{a_j, f_i\}, \{b_i, c_j\}, \{b_j, c_i\} \in E$. If $X_{ij} = 0$, then $\{a_i, b_j\}, \{a_j, b_i\}, \{c_i, f_j\}, \{c_j, f_i\} \in E$.

Then we can define the jth neighbor of a_i to be f_j if $X_{ij} = 1$ and b_j otherwise. The jth neighbor of b_i is c_j if $X_{ij} = 1$ and a_j otherwise. The jth neighbors of c_i and f_i are defined analogously. Since each node in $A \cup B \cup C \cup F$ has degree N, we do not have to query x to simulate the degree oracle. Lastly, each query to the neighborhood can be answered with one query to x.

Now, v, b_i, and c_j form a triangle if and only if $X_{ij} = 1$. So, given an algorithm for the triangle vertex problem which uses at most q quantum queries, then we can find a 1 in x

using $O(\sqrt{q})$ queries. Since we know that the problem of deciding whether x contains a 1 has query complexity $\Omega(\sqrt{N})$, we know that $q = \Omega\left(\sqrt{\binom{N}{2}}\right) = \Omega(N)$. □

Theorem 1.1 follows from Lemmas 3.1 and 4.1.

4.2. Triangle Edge Determination. In this section we show that in the adjacency list model, the problem of determining whether an edge is a triangle edge has quantum query complexity $\Omega(N^{2/3}/\log N)$, completing the proof of Theorem 1.2. To accomplish this, we show that the triangle edge problem requires just as many queries (asymptotically) as the *element distinctness problem*, which asks whether a function $f : [N] \to [M]$ is one-to-one (i.e. has no collision). Prior work showed that the promise problem in which $M = N$ and there is at most one collision in f also has quantum query complexity $\Omega(N^{2/3})$ [13]. We will let UElementDistinctness denote this special case of element distinctness.

The reduction from UElementDistinctness to the problem of determining whether an edge is a triangle edge relies on the following ideas. Suppose we have disjoint sets $W = \{w_1, \ldots, w_{\ell/2}\}$ and $Z = \{z_1, \ldots, z_{\ell/2}\}$ which are both subsets of the domain of our function f. Suppose further that between W and Z there is either a single collision pair (and no more) w_i and z_j such that $f(w_i) = f(z_j)$ and $i \neq j$ or there are none. Now we want to set up a graph with an edge $\{u, v\}$ such that $\{u, v\}$ participates in a triangle if and only if some w_i and z_j form a collision. A natural approach is to have a set of N vertices $L_1, \ldots, L_N$ with an edge from u to L_i whenever $f(w_j) = i$ and an edge from v to L_i whenever $f(z_j) = i$. At a glance, this seems to work, as the resulting graph G has a triangle whenever W and Z have a collision. The problem, however, is that determining the degree of L_i is essentially the same as checking whether a list contains a certain element, meaning we cannot simulate it with a constant number of queries to f. To circumvent this problem, we add an additional N nodes $R_1, \ldots, R_N$ and connect L_i to R_j whenever $f(w_j) \neq i$ and $f(z_j) \neq i$. Now we can straightforwardly implement the neighborhood and degree oracles with only a constant number of queries to f.

To increase our success probability, we split up our input $X = \langle x_1, \ldots, x_N \rangle$ into more than two sublists, each of size $\ell = \Theta(N)$ so that the probability that a collision pair is in the same sublist is small. We can now construct the graph mentioned above on every pair of these sublists to determine whether f has a collision with probability $\gg 1/2$.

Lemma 4.2. *The problem of determining whether an edge of a graph on N nodes is a triangle edge requires $\Omega\left(N^{2/3}\right)$ queries.*

Proof. Let $f : [N] \to [N]$ be an instance of UElementDistinctness. Let $\mathcal{A}$ be an $A(N)$-query algorithm for triangle edge detection. We construct the $B(N)$-query algorithm $\mathcal{B}$, which does the following on input f. Let π be a random permutation of $[N]$, and let $F = \langle f(\pi(1)), \ldots f(\pi(N)) \rangle$. Now we divide F into N/ℓ sublists $F^i = \langle F_{\ell(i-1)+1}, \ldots, F_{\ell i} \rangle$ for $\ell = \frac{N}{3}$. The probability that no sublist contains a collision pair and any two sublists which share a value share it at different indices is at least $1 - \left(\frac{\ell}{N}\right)^2 \cdot \frac{N}{\ell} - \frac{1}{\ell} = 2/3 - o(1)$. For now, assume this is the case.

The graph $G_{ij} = (V, E)$ is constructed as follows. The vertex set V of size $2N + 2$ consists of two sets of nodes corresponding to the range $L_1, \ldots, L_N$ and $R_1, , \ldots, R_N$ as well

as two helper nodes u and v. The edge set is defined such that

$$\begin{aligned} E = &\{\{u, v\}\} \\ &\cup \{\{u, L_a\} : a \in F^i\} && u \text{ is adjacent to nodes corresponding to } F^i \\ &\cup \{\{v, L_a\} : a \in F^j\} && v \text{ is adjacent to nodes corresponding to } F^j \\ &\cup \{\{L_a, R_b\} : F^i[b], F^j[b] \neq a\} && L_a \text{ is adjacent to } R_b \text{ if } F^i[b] \neq a \text{ and } F^j[b] \neq a. \end{aligned}$$

First, we will show that neighborhood queries can be answered using $O(1)$ queries to f. The bth neighbor of nodes u and v can be answered by simply querying the bth entry of F^i and F^j, respectively. The bth neighbor of L_a is

$$\Gamma(L_a, b) = \begin{cases} R_b & \text{if } F^i[b], F^j[b] \neq a \\ u & \text{if } F^i[b] = a \\ v & \text{if } F^j[b] = a, \end{cases}$$

so any neighbor query can be answered with $O(1)$ queries to the list. Recall that we are assuming that $F^i[b] \neq F^j[b]$ for all b. For the R_a nodes, we let $\Gamma(R_a, b) = L_{b'}$, where $b' = b + \left|\left\{r \leq i : F^i[a] = r \text{ or } F^j[a] = k\right\}\right|$, which can be answered by querying $F^i[a]$ and $F^j[a]$. By our assumption $d_{R_a} = N - \left|\left\{r \in [N] : F^i[a] = r \text{ or } F^j[a] = r\right\}\right|$. So, answering the degree queries also only takes $O(1)$ queries to f. Nodes u and v have degree ℓ and each L_a has degree N. Thus, the probability $\mathcal{B}$ successfully find a collision is at least $2/3 - o(1)$ which can be amplified using majority voting. Furthermore, $B(N) = O(A(N))$, so $A(N) = \Omega(N^{2/3}/\log N)$. □

Theorem 1.2 follows from Lemmas 3.2 and 4.2.

5. Conclusion and Open Problems. In this work, we show that the quantum query complexity of the problem of determining whether a node participates in a triangle is $\Theta(N)$ and for edges is $\Theta(N^{2/3})$. The natural next step is to give nontrivial upper or lower bounds on the quantum query complexity of determining whether a graph has a triangle. A promising avenue may be to use similar techniques as we did to lower bound the edge triangle problem. We can generalize the element distinctness problem to the case where we are given any function $f : [N] \to [M]$ and are asked to determine whether f is one-to-one. In this case, the problem of determining whether at least 1 of N instances of this new problem is one-to-one has query complexity $\Omega(N^{7/6})$ [12]. Likewise, it is still open whether there is a $o(N^{3/2})$ algorithm for determining whether a graph admits a triangle in the adjacency list model.

REFERENCES

[1] M. Al Hasan and V. S. Dave, *Triangle counting in large networks: a review*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8 (2018), p. e1226.

[2] A. Ambainis, *Quantum walk algorithm for element distinctness*, SIAM Journal on Computing, 37 (2007), pp. 210–239.

[3] B. Andreopoulos, C. Winter, D. Labudde, and M. Schroeder, *Triangle network motifs predict complexes by complementing high-error interactomes with structural information*, BMC bioinformatics, 10 (2009), p. 196.

[4] A. Azad, A. Buluç, and J. Gilbert, *Parallel triangle counting and enumeration using matrix algebra*, in 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IEEE, 2015, pp. 804–811.

[5] T. Eden, A. Levi, D. Ron, and C. Seshadhri, *Approximately counting triangles in sublinear time*, SIAM Journal on Computing, 46 (2017), pp. 1603–1646.

[6] M. Gonen, D. Ron, and Y. Shavitt, *Counting stars and other small subgraphs in sublinear-time*, SIAM Journal on Discrete Mathematics, 25 (2011), pp. 1365–1411.

[7] O. Green, P. Yalamanchili, and L.-M. Munguía, *Fast triangle counting on the gpu*, in Proceedings of the 4th Workshop on Irregular Applications: Architectures and Algorithms, 2014, pp. 1–8.
[8] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 212–219.
[9] P. W. Holland and S. Leinhardt, *A method for detecting structure in sociometric data*, American Journal of Sociology, 76 (1970), pp. 492–513.
[10] R. Pagh and C. E. Tsourakakis, *Colorful triangle counting and a mapreduce implementation*, Information Processing Letters, 112 (2012), pp. 277–281.
[11] A. Portes, *Social capital: Its origins and applications in modern sociology*, Annual Review of Sociology, 24 (1998), pp. 1–24.
[12] B. W. Reichardt, *Reflections for quantum query algorithms*, in Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, USA, 2011, Society for Industrial and Applied Mathematics, p. 560–569.
[13] A. Rosmanis, *Adversary lower bound for element distinctness with small range*, arXiv preprint arXiv:1401.3826, (2014).
[14] T. Schank and D. Wagner, *Finding, counting and listing all triangles in large graphs, an experimental study*, in Experimental and Efficient Algorithms, S. E. Nikoletseas, ed., Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 606–609.