Improved Quantum Algorithm for Triangle Finding via Combinatorial Arguments

François Le Gall
The University of Tokyo

Technical version available at arXiv:1407.0085 [quant-ph].

Background. Triangle finding is a graph-theoretic problem whose complexity is deeply connected to the complexity of several other computational tasks in theoretical computer science, such as solving path or matrix problems [3, 8, 9, 13, 18, 17, 19]. In its standard version (sometimes called *unweighted triangle finding*), it asks to find, given an undirected and unweighted graph $G = (V, E)$, three vertices $v_1, v_2, v_3 \in V$ such that $\{v_1, v_2\}$, $\{v_1, v_3\}$ and $\{v_2, v_3\}$ are edges of the graph.

Problems like triangle finding can be studied in the query complexity setting. In the usual model used to describe the query complexity of such problems, the set of edges E of the graph is unknown but can be accessed through an oracle: given two vertices u and v in V, one query to the oracle outputs one if $\{u, v\} \in E$ and zero if $\{u, v\} \notin E$. In the quantum query complexity setting, one further assume that the oracle can be queried in superposition. Besides its intrinsic interest, the triangle finding problem has been one of the main problems that stimulated the development of new techniques in quantum query complexity, and the history of improvement of upper bounds on the query complexity of triangle finding parallels the development of general techniques in the quantum complexity setting, as we explain below.

Grover search immediately gives, when applied to triangle finding as a search over the space of triples of vertices of the graph, a quantum algorithm with query complexity $O(n^{3/2})$. Using amplitude amplification, Buhrman et al. [7] showed how to construct a quantum algorithm for triangle finding with query complexity $O(n + \sqrt{nm})$ for a graph with m edges, giving an improvement for sparse graphs. Combining amplitude amplification with clever combinatorial arguments, Szegedy [16] (see also [12]) constructed a quantum algorithm for triangle finding with query complexity $\tilde{O}(n^{10/7}) = \tilde{O}(n^{1.428\cdots})$.[1]

The quantum technique that led to the next improvement was the concept of quantum walk search developed by Ambainis [1], which has turned out to be one of the most useful tools for the design of quantum algorithms for search problems. Magniez, Santha and Szegedy [12], using quantum walk search, constructed a quantum algorithm for triangle finding with improved query complexity $\tilde{O}(n^{13/10})$.

Besides Grover search and quantum walks, a third technique to design quantum query algorithms appeared recently when Reichardt [14] proved that the quantum query complexity of a problem can be found by solving a semi-definite positive program. While this optimization problem in general exponentially many constraints, Belovs [5] then developed a technique known as the learning graph approach to restrict the search space to candidates that automatically satisfy the constraints, thus giving an intuitive and efficient way to obtain a (not necessarily optimal) solution of the original optimization problem. Belovs [5] illustrated the power of this new technique by using it to improve the quantum query complexity of triangle finding to $O(n^{35/27}) = O(n^{1.296\cdots})$. Lee, Magniez and Santha [11] then showed, again using learning graphs, how to further improve this query complexity to $O(n^{9/7}) = \tilde{O}(n^{1.285\cdots})$, which was the best upper bound on the quantum complexity of triangle finding known before the present work. These two results based on learning graphs actually used a simple notion of learning graphs (referred to as "non-adaptive" learning graphs in [6]) where the queries done by the algorithm do not depend on the values of prior queries, which implies that the same upper bound $O(n^{9/7})$ holds for edge-weighted

[1] The $\tilde{O}(\cdot)$ notation removes $\mathrm{poly}(\log n)$ factors.

versions of the triangle finding problem[2] as well. Jeffery, Kothari and Magniez [10] showed how this complexity can also be achieved, up to polylogarithmic factors, using quantum walks by introducing the concept of nested quantum walks.

The best known lower bound on the quantum query complexity of triangle finding is the trivial $\Omega(n)$. Belovs and Rosmanis [6] recently showed that any quantum algorithm (i.e., not necessarily based on learning graphs) solving the edge-weighted triangle finding problem requires $\Omega(n^{9/7}/\sqrt{\log n})$ queries. Since a non-adaptive learning graph does not treat differently the unweighted triangle finding problem and its weighted versions, as mentioned above, this lower bound for the weighted case implies that any quantum algorithm for unweighted triangle finding constructed using a non-adaptive learning graph requires $\Omega(n^{9/7}/\sqrt{\log n})$ queries as well, which matches, up to logarithmic factors, the best known upper bound described in the previous paragraph. Practically, this means that, in order to improve by more than a $1/\sqrt{\log n}$ factor the $O(n^{9/7})$-query upper bound on the quantum query complexity of triangle finding, one need to take in consideration the difference between the unweighted triangle finding problem and its edge-weighted version. Moreover, if the learning graph approach is used, then the learning graph constructed must be adaptive. While a concept of adaptive learning graph has been developed by Belovs and used to design a new quantum algorithm for the k-distinctness problem [4], so far no application of this approach to the triangle finding problem has been discovered.

Statement of our result. In this work we show that it is possible to overcome the $\Omega(n^{9/7}/\sqrt{\log n})$ barrier, and obtain the following result.

Theorem 1. *There exists a quantum algorithm that, given as input the oracle of an unweighted graph G on n vertices, outputs a triangle of G with probability at least $2/3$ if a triangle exists, and uses $\tilde{O}(n^{5/4})$ queries to the oracle.*

This result shows, for the first time, that in the quantum setting the standard (i.e., unweighted) triangle finding problem is easier than its edge-weighted versions, and thus sheds light on the fundamental difference between these two problems. Indeed, while in the classical time complexity setting evidences exist suggesting that the unweighted version is easier (see, e.g., [13, 18]), Theorem 1, combined with the lower bound by Belovs and Rosmanis [6], enables us to give a separation between the quantum query complexities of these two problems.

Naturally, our result exploits the difference between the triangle finding problem and its weighted versions. Our approach does not rely on learning graphs or nested quantum walks, the techniques that were used to obtain the previous best known upper bound. Instead, it relies on combinatorial ideas that exploit the fact that the graph is unweighted, as needed in any attempt to break the $\Omega(n^{9/7}/\sqrt{\log n})$ barrier, combined with Grover search, quantum search with variable costs [2], and usual quantum walks over Johnson graphs. Our quantum algorithm is highly adaptive, in that all later queries depend on the results of the queries done in at a preliminary stage by the algorithm. This gives another example of separation between the query complexity obtained by adaptive quantum query algorithms and the best query complexity that can be achieved using non-adaptive learning graphs (which is $\Omega(n^{9/7}/\sqrt{\log n})$ for triangle finding, as mentioned above), and thus sheds light on limitations of the non-adaptive learning graph approach for graph-theoretical problems such as triangle finding.

Overview of our quantum algorithm. Let $G = (V, E)$ denote the undirected and unweighted graph that is the input of the triangle finding problem, and write $n = |V|$. For any set $Y \subseteq V$, we use the notation $\mathcal{E}(Y) = \{\{u, v\} \mid u, v \in Y\}$ to represent the set of unordered pairs of elements in Y. For any vertex $u \in V$, we denote $N_G(u) = \{v \in V \mid \{u, v\} \in E\}$ the set of neighbors of u.

The algorithm first takes a set $X \subseteq V$ consisting of $\Theta(\sqrt{n}\log n)$ vertices chosen uniformly at random from V, and checks if there exists a triangle of G with a vertex in X. This can be checked, with

[2]Weighted versions of the triangle finding problem ask to find three vertices in the graph such that the sum of the weights of the corresponding three edges satisfy some condition (e.g., they sum to zero).

high probability, using Grover search in

$$O\left(\sqrt{|X| \times |\mathcal{E}(V)|}\right) = \tilde{O}\left(n^{5/4}\right)$$

queries. Define $S = \bigcup_{u \in X} \mathcal{E}(N_G(u))$. If no triangle has been reported, we know that any triangle of G must have an edge in the set $\mathcal{E}(V) \setminus S$. Note that the above preliminary step has already been used in prior works, in particular related to the design of combinatorial algorithms for Boolean matrix multiplication (e.g., [3, 15]) and even in the design of the $\tilde{O}(n^{10/7})$-query quantum algorithm for triangle finding in [12, 16]. We now explain how to check whether $\mathcal{E}(V) \setminus S$ contains an edge of a triangle or not, which is the novel contribution of our work.

For any set $Y \subseteq V$ and any $w \in V$, let us define the set $\Delta_G(X, Y, w) \subseteq \mathcal{E}(Y)$ as follows:

$$\Delta_G(X, Y, w) = \mathcal{E}(Y \cap N_G(w)) \setminus S.$$

It is easy to see that, with high probability on the choice of X, this set will be "sparse". We will not give the precise definition of sparsity here, but instead describe our algorithm in the following ideal (but still typical) situation: there exists a positive constant c such that

$$|\Delta_G(X, Y, w)| \leq \frac{c|Y|^2}{\sqrt{n}} \quad \text{for all } Y \subseteq V \text{ and } w \in V. \tag{1}$$

Remember that we now want to check if $\mathcal{E}(V) \setminus S$ contains an edge of a triangle. Our key observation is the following. Given a vertex $w \in V$ and a set $B \subseteq V$ of size $\lceil \sqrt{n} \rceil$ *such that* $\Delta_G(X, B, w)$ *is known*, we can check if there exists a pair $\{v_1, v_2\} \in \mathcal{E}(B) \setminus S$ such that $\{v_1, v_2, w\}$ is a triangle of G with

$$O\left(\sqrt{|\Delta_G(X, B, w)|}\right) = O\left(\sqrt{\frac{c|B|^2}{\sqrt{n}}}\right) = O(n^{1/4})$$

queries using Grover search and Condition (1), since such $\{v_1, v_2\}$ exists if and only if $\Delta_G(X, B, w) \cap E \neq \emptyset$. The remarkable point here is that, if there were no sparsity condition on $\Delta_G(X, B, w)$ then this search would require $\Theta(\sqrt{|B|^2}) = \Theta(\sqrt{n})$ queries. This improvement from $\sqrt{n}$ to $n^{1/4}$ is one of the main reasons why we obtain an algorithm for triangle finding with query complexity $\tilde{O}(n^{5/4})$ instead of $O(n^{3/2})$ using straightforward quantum search.

The main difficulty when trying to exploit the above observation is that we not only want now to find a vertex w and a set B for which there exists $\{v_1, v_2\} \in \mathcal{E}(B) \setminus S$ such that $\{v_1, v_2, w\}$ is a triangle, we also need to construct the set $\Delta_G(X, B, w)$, which requires additional queries. To deal with this problem, we use a quantum walk over a Johnson graph, which enables us to implement the construction of $\Delta_G(X, B, w)$ concurrently to the search of B and w. By carefully analyzing the resulting quantum walk algorithm, we can show that the improvement by a factor $n^{1/4}$ described in the previous paragraph is still preserved as long as we have enough prior information about the set S when executing the walk.

The difficulty now is that loading enough information about S during the execution of the quantum walk is too costly. Moreover, constructing S before executing the quantum walk requires $\Theta(n^{3/2})$ queries, which is too costly as well. To solve this difficulty, we first search, using another quantum walk on another Johnson graph, a set $A \subseteq V$ of size $\lceil n^{3/4} \rceil$ such that $\left(\bigcup_{w \in V} \Delta_G(X, A, w)\right) \cap E \neq \emptyset$, and concurrently construct the set $\mathcal{E}(A) \setminus S$. We then do exactly as in the previous paragraph, but taking B as a subset of A instead of as a subset of V. Since $\Delta_G(X, B, w)$ can be created efficiently from the knowledge of $\mathcal{E}(A) \setminus S$, and $\mathcal{E}(A) \setminus S$ is available in the memory of the new quantum walk, the problem mentioned in the previous paragraph is solved. By carefully designing the new quantum walk, we can show that its query complexity is sufficiently small.

To summarize, we obtain a (four-level) recursive procedure involving quantum walks that checks if $\mathcal{E}(V) \setminus S$ contains an edge of a triangle, and thus checks if G contains a triangle. Several technical difficulties arise when analyzing the performance of this recursive quantum algorithm and showing that its query complexity is $\tilde{O}(n^{5/4})$, especially when Condition (1) does not hold. They are dealt with by using additional quantum techniques, such as quantum search with variable costs [2], estimating the size of the involved sets by random sampling, and proving several concentration bounds.

References

[1] AMBAINIS, A. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing 37*, 1 (2007), 210–239.

[2] AMBAINIS, A. Quantum search with variable times. *Theory of Computing Systems 47*, 3 (2010), 786–807.

[3] BANSAL, N., AND WILLIAMS, R. Regularity lemmas and combinatorial algorithms. *Theory of Computing 8*, 1 (2012), 69–94.

[4] BELOVS, A. Learning-graph-based quantum algorithm for k-distinctness. In *Proceedings of the 53rd Symposium on Foundations of Computer Science* (2012), pp. 207–216.

[5] BELOVS, A. Span programs for functions with constant-sized 1-certificates: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing* (2012), pp. 77–84.

[6] BELOVS, A., AND ROSMANIS, A. On the power of non-adaptive learning graphs. In *Proceedings of the 28th Conference on Computational Complexity* (2013), pp. 44–55.

[7] BUHRMAN, H., DÜRR, C., HEILIGMAN, M., HØYER, P., MAGNIEZ, F., SANTHA, M., AND DE WOLF, R. Quantum algorithms for element distinctness. *SIAM Journal on Computing 34*, 6 (2005), 1324–1330.

[8] CZUMAJ, A., AND LINGAS, A. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM Journal on Computing 39*, 2 (2009), 431–444.

[9] ITAI, A., AND RODEH, M. Finding a minimum circuit in a graph. *SIAM Journal on Computing 7*, 4 (1978), 413–423.

[10] JEFFERY, S., KOTHARI, R., AND MAGNIEZ, F. Nested quantum walks with quantum data structures. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms* (2013), pp. 1474–1485.

[11] LEE, T., MAGNIEZ, F., AND SANTHA, M. Improved quantum query algorithms for triangle finding and associativity testing. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms* (2013), pp. 1486–1502.

[12] MAGNIEZ, F., SANTHA, M., AND SZEGEDY, M. Quantum algorithms for the triangle problem. *SIAM Journal on Computing 37*, 2 (2007), 413–424.

[13] PATRASCU, M. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd Symposium on Theory of Computing* (2010), pp. 603–610.

[14] REICHARDT, B. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th Symposium on Foundations of Computer Science* (2009), pp. 544–551.

[15] SCHNORR, C.-P., AND SUBRAMANIAN, C. R. Almost optimal (on the average) combinatorial algorithms for Boolean matrix product witnesses, computing the diameter (extended abstract). In *Proceedings of the 2nd workshop on Randomization and Approximation Techniques in Computer Science* (1998), pp. 218–231.

[16] SZEGEDY, M. On the quantum query complexity of detecting triangles in graphs. arXiv:quant-ph/0310107, 2003.

[17] VASSILEVSKA WILLIAMS, V., AND WILLIAMS, R. Subcubic equivalences between path, matrix and triangle problems. In *Proceedings of the 51th Symposium on Foundations of Computer Science* (2010), pp. 645–654.

[18] VASSILEVSKA WILLIAMS, V., AND WILLIAMS, R. Finding, minimizing, and counting weighted subgraphs. *SIAM Journal on Computing 42*, 3 (2013), 831–854.

[19] WILLIAMS, R. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Symposium on Theory of Computing* (2014), pp. 664–673.