Improved Quantum Query Algorithms for Triangle Finding and Associativity Testing*

Troy Lee[†1], Frédéric Magniez[‡2], and Miklos Santha[§1,2]

[1]Centre for Quantum Technologies, National University of Singapore, Singapore 117543
[2]CNRS, LIAFA, Univ Paris Diderot, Sorbonne Paris-Cité, 75205 Paris, France

Quantum query complexity is a black-box model of quantum computation, where the resource measured is the number of queries to the input needed to compute a function. This model captures the great algorithmic successes of quantum computing like the search algorithm of Grover [5] and the period finding subroutine of Shor's factoring algorithm [10], while at the same time is simple enough that one can often show tight lower bounds.

Recently, there have been very exciting developments in quantum query complexity. Reichardt [9] showed that the general adversary bound, formerly just a lower bound technique for quantum query complexity [6], is also an upper bound. This characterization opens a new avenue for designing quantum query algorithms. The general adversary bound can be written as a relatively simple semidefinite program, thus by providing a feasible solution to the minimization form of this program one can upper bound quantum query complexity.

This plan turns out to be quite difficult to implement as the minimization form of the adversary bound has exponentially many constraints. Even for simple functions it can be challenging to directly write down a feasible solution, much less worry about finding a solution with good objective value. To surmount this problem, Belovs [2] introduced the beautiful model of learning graphs, which can be viewed as the minimization form of the general adversary bound with additional structure imposed on the form of the solution. This additional structure makes learning graphs easier to reason about by ensuring that the constraints are *automatically* satisfied, leaving one to worry about optimizing the objective value.

Learning graphs have already proven to be a very successful model for designing quantum query algorithms. In his original paper [2], Belovs gave an algorithm for triangle finding with complexity $O(n^{35/27})$, improving the quantum walk algorithm of Magniez et al. [8] with complexity $O(n^{1.3})$. Belovs' triangle algorithm was later generalized to detecting constant-sized subgraphs [11, 7], giving an algorithm of complexity $o(n^{2-2/k})$ for determining if a graph contains a k-vertex subgraph H, again improving the [8] bound of $O(n^{2-2/k})$. All these algorithms use the most basic model of learning graphs, that we also use in this paper. A more general model of learning graphs (introduced, though not used in Belovs' original paper) was used to give an $o(n^{3/4})$ algorithm for k-element distinctness, when the inputs are promised to be of a certain form [3]. Recently, Belovs further generalized the learning graph model and removed this promise to obtain an $o(n^{3/4})$ algorithm for the general k-distinctness problem [1].

*To appear in SODA 2013. Full version available at http://arxiv.org/abs/1210.1014
†troyjlee@gmail.com
‡frederic.magniez@univ-paris-diderot.fr
§miklos.santha@liafa.jussieu.fr

In this paper, we continue to show the power of the learning graph model. We give an algorithm for detecting a triangle in a graph making $O(n^{9/7})$ queries. This lowers the exponent of Belovs algorithm from about 1.296 to under 1.286. For the problem of testing if an operation $\circ : S \times S \to S$ is associative, where $|S| = n$, we give an algorithm making $O(n^{10/7})$ queries, the first improvement over the trivial application of Grover search with complexity $O(n^{3/2})$. Previously, Dörn and Thierauf [4] gave a quantum walk based algorithm to test if $\circ : S \times S \to S'$ is associative that improved on Grover search but only when $|S'| < n^{3/4}$.

Our main contributions are two-fold. On the technical side, our improved algorithms result by allowing more freedom in the choice of parameters in learning graphs for detecting constant-sized subgraphs, compared to previous works [2, 11, 7]. Optimizing over this larger parameter space leads to our improvements in complexity. The basic model of learning graphs that we use has the interesting property that the complexity depends only on the structure of the 1-certificates, and not on the underlying alphabet. This means that our subgraph detection algorithms immediately give algorithms for detecting subgraphs with specified edge colors in a colored graph. This allows the extension to problems like associativity testing, which can be viewed as detecting the presence of a handful of constant-size colored and directed graphs in a directed colored graph.

On the conceptual side, we make the design of learning graph algorithms more accessible to the uninitiated. Arguments analyzing the complexity of learning graphs can be tedious and repetitive—a bit like coding in assembly. We provide a high-level language with simple commands like "load vertex a" or "load edge (a,b)" and rules for the order in which they can be executed. This informal terminology was introduced by Belovs, and our main theorem rigorously compiles this high-level language into a learning graph and gives the resulting complexity. One can now design quantum query algorithms solely using the high-level language and not worry about the nitty gritty learning graph implementation.

Our contribution. We now explain the new ideas in our learning graphs in more detail, using triangle detection as an example. We first review the quantum walk algorithm of [8], and the learning graph algorithm of Belovs [2]. For this high-level overview we just focus on the database of edge slots of the input graph G that is maintained by the algorithm. A quantum walk algorithm explicitly maintains such a database, and the nodes of a learning graph are labeled by sets of queries which we will similarly interpret as the database of the algorithm.

In the quantum walk algorithm [8] the database consists of an r-element subset of the n-vertices of G and all the edge slots among these r-vertices. That is, the presence or absence of an edge in G among a complete r-element subgraph is maintained by the database. In the learning graph algorithm of Belovs, the database consists of a random subgraph with edge density $0 \leq s \leq 1$ of a complete r-element subgraph. In this way, on average, $O(sr^2)$ many edge slots are queried among the r-element subset, making it cheaper to set up this database. This saving is what results in the improvement of Belovs' algorithm. Both algorithms finish by using search plus graph collision to locate a vertex that is connected to the endpoints of an edge present in the database, forming a triangle.

Zhu [11] and Lee et al. [7] extended the triangle finding algorithm of Belovs to finding constant sized subgraphs. While the algorithm of Zhu again maintains a database of a random subgraph of an r-vertex complete graph with edge density s, the algorithm of Lee et al. instead used a more structured database. Let H be a k-vertex subgraph with vertices labeled from $[k]$. To determine if G contains a copy of H, the database of the algorithm consists of $k-1$ sets $A_1, \ldots, A_{k-1}$ of size r and for every $\{i, j\} \in H - \{k\}$ the edge slots of G according to a sr-regular bipartite graph between A_i and A_j. Again both algorithms finish by using search plus graph collision to find a vertex connected to edges in the database to form a copy of H.

In this work, our database is again the edge slots of G queried according according to the union of regular bipartite graphs whose overall structure mimics the structure of H. Now, however, we allow optimization

	Setup	load a_1	load a_2	load $\{a_1, a_2\}$	find a_3 + graph collision
[8]	r^2	$\sqrt{n}r$	$n\sqrt{r}$	n	$n^{3/2}/r^{1/3}$
[2]	sr^2	$\sqrt{n}sr$	$ns\sqrt{r}$	n	$n^{3/2}/(\sqrt{s}r^{1/3})$
This paper	$r_1 r_2$	$\sqrt{n}r_2$	$n\sqrt{r_1}$	n	$n^{3/2}/(r_1 r_2)^{1/6}$

Table 1: A comparison of the costs in the triangle finding algorithms of [8, 2] and this paper. The vertices of the triangle are a_1, a_2, a_3. Parameters $r, r_1, r_2 \in [n]$ specify set sizes in the algorithms, and $s \in [0,1]$ specifies an edge density. In all algorithms, loading a_1 is low order. By taking $r_2 > r_1$ we can increase the complexity of this step (though it remains low order) and decrease the complexity of the final step, resulting in our improved complexity.

over all parameters of the database—we allow the size of the set A_i to be a parameter r_i that can be independently chosen; similarly, we allow the degree of the bipartite graph between A_i and A_j to be a variable d_{ij}. This greater freedom in the parameters of the database allows the improvement in triangle finding from $O(n^{35/27})$ to $O(n^{9/7})$. Instead of an r-vertex graph with edge density s, our algorithm uses as a database a complete unbalanced bipartite graph with left hand side of size r_1 and right hand side of size r_2. Taking $r_1 < r_2$ allows a more efficient distribution of resources over the course of the algorithm. As before, the algorithm finishes by using search plus graph collision to find a vertex connected to endpoints of an edge in the database. A comparison of the costs of each step for the quantum walk algorithm [8], the algorithm of Belovs [2], and the algorithm given here can be found in Table 1.

The extension to functions of the form $f : [q]^{n \times n} \to \{0,1\}$, like associativity, comes from the fact that the basic learning graph model that we use depends only on the structure of a 1-certificate and not on the values in a 1-certificate. This property means that an algorithm for detecting a subgraph H can be immediately applied to detecting H with specified edge colors in a colored graph.

If an operation $\circ : S \times S \to S$ is non-associative, then there are elements a, b, c such that $a \circ (b \circ c) \neq (a \circ b) \circ c$. A certificate consists of the 4 (colored and directed) edges $b \circ c = e, a \circ e, a \circ b = d$, and $d \circ c$ such that $a \circ e \neq d \circ c$. The graph of this certificate is a 4-path with directed edges, and using our algorithm for this graph gives complexity $O(|S|^{10/7})$.

We provide a high-level language for designing algorithms within our framework. The algorithm begins by choosing size parameters for each A_i and degree parameters for the bipartite graph between A_i and A_j. Then one can choose the order in which to load vertices a_i and edges (a_i, a_j) of a 1-certificate, according to the rules that both endpoints of an edge must be loaded before the edge, and at the end all edges of the certificate must be loaded. Our main theorem shows how to implement this high-level algorithm as a learning graph and gives the resulting complexity.

With larger subgraphs, optimizing over the set size and degree parameters to obtain an algorithm of minimal complexity becomes unwieldy to do by hand. Fortunately, this can be phrased as a linear program and we provide code to compute a set of optimal parameters [1].

A full version of our paper can be found on the arXiv at `http://arxiv.org/abs/1210.1014`.

[1] code is available at `https://github.com/troyjlee/learning_graph_lp`

References

[1] A. Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *Prooceedings of 53rd Annual IEEE Symposium on Foundations of Computer Science*, 2012.

[2] A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of 44th Symposium on Theory of Computing Conference*, pages 77–84, 2012.

[3] A. Belovs and T. Lee. Quantum algorithm for k-distinctness with prior knowledge on the input. Technical Report arXiv:1108.3022, arXiv, 2011.

[4] S. Dörn and T. Thierauf. The quantum complexity of group testing. In *Proceedings of the 34th conference on current trends in theory and practice of computer science*, pages 506–518, 2008.

[5] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM Symposium on the Theory of Computing*, pages 212–219, 1996.

[6] Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of 39th ACM Symposium on Theory of Computing*, pages 526–535, 2007.

[7] T. Lee, F. Magniez, and M. Santha. A learning graph based quantum query algorithm for finding constant-size subgraphs. Technical Report arXiv:1109.5135, arXiv, 2011.

[8] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007.

[9] Ben W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of 22nd ACM-SIAM Symposium on Discrete Algorithms*, pages 560–569, 2011.

[10] P. Shor. Algorithms for quantum computation: Discrete logarithm and factoring. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[11] Y. Zhu. Quantum query complexity of subgraph containment with constant-sized certificates. Technical Report arXiv:1109.4165v1, arXiv, 2011.