Improved Quantum Query Complexity Bounds for Some Graph Problems

Prafulla Dhariwal, Vinay Mayar

Abstract

We prove improved quantum query complexity bounds for some graph problem. Our results are based on a new quantum algorithm in [1] that could be used to improve query complexity upper bounds. We prove a lower bound of $\Omega\left(k^{1/2}n^{3/2}\right)$ queries to an adjacency matrix for the k-source shortest paths problem for unweighted graphs, which matches the upper bound proved in [1]. We also prove an upper bound of $O(n^{1.5})$ queries for finding the minimum vertex cover in a bipartite graph G if we are given the maximum matching in G. In [1], Lin and Lin proved that the latter could be found in $O(n^{1.75})$ queries, which gives us an $O(n^{1.75})$ upper bound for the minimum vertex cover problem for bipartite graphs. We also discuss the implications of their results on the query complexity of other related graph problems.

Keywords: quantum computation, quantum query complexity, complexity theory, shortest path, vertex cover, maximum matching, graph algorithms

1. Introduction

A key motivation for the study of quantum complexity theory is to determine when a quantum computer can give a speedup over classical computers. Currently, only a few problems have been shown to be computable in polynomial time by a quantum algorithm while requiring exponential time classically. One of the ways we can analyze the computational speedup of quantum computers is by comparing the number of bits of the input that a quantum algorithm has to check to compute a function to the number of bits a classical algorithm has to check. The minimum number of bits that an algorithm must check to compute a function is the query complexity of the function. The query complexity of a function is often closely related to its time complexity and gives insight into the relative power of quantum computers. In this paper we analyze the quantum query speedups for some graph problems.

Quantum query complexity bounds already exist for many graph problems. Dürr et al. prove almost tight bounds for CONNECTIVITY, STRONG CONNECTIVITY, SINGLE-SOURCE SHORTEST PATHS, and MINIMUM SPANNING TREE [2]. Classically, an algorithm must look at almost the entire input to solve these problems, but quantum algorithms can do better. To determine the quantum query complexity of a problem, we find an upper bound for the quantum query complexity, usually by devising and analyzing an algorithm, and a lower bound by a variety of methods. Once the gap between the upper and lower bounds has been removed, we know the quantum query complexity of the problem. There remains a gap between the best upper and lower bounds for the quantum query complexity of k-SOURCE SHORTEST PATHS, MAXIMUM MATCHING, and MINIMUM VERTEX COVER. The goal of this paper is to reduce these gaps.

Email addresses: prafulla@mit.edu (Prafulla Dhariwal), vmayar@mit.edu (Vinay Mayar)

1.1. Our Contributions

In Section 3, we prove a lower bound of $\Omega\left(k^{1/2}n^{3/2}\right)$ queries to an adjacency matrix for the k-source shortest paths problem for unweighted graphs, which matches the upper bound proved in [1]. In Section 4, we prove an upper bound of $O(n^{1.5})$ queries for finding the minimum vertex cover in a bipartite graph G if we are given the maximum matching in G. In [1], Lin and Lin proved that a maximum matching for a bipartite graphs can be found in $O(n^{1.75})$ queries. Combined with our reduction, this proves a new upper bound of $O\left(n^{1.75}\right)$ for finding the minimum vertex cover in bipartite graphs. From results in [2], we get a lower bound of $\Omega(n^{1.5})$ for the same. Though not tight, our upper bound automatically decreases if a better upper bound is found for MAXIMUM MATCHING in bipartite graphs.

2. Preliminaries

2.1. Graph Theory

Let $G = (V, E)$ be an undirected graph, where V and E denote the set of vertices and edges of G. Let $n = |V|$ be the number of vertices and $m = |E|$ be the number of edges. In the problems considered in this paper, we will be given a graph as input and will need to query the existence of edges between vertices. We consider two models for querying edges in a graph.

Adjacency Matrix We are given the adjacency matrix $A \in \{0,1\}^{n\times n}$ of G with $A_{i,j} = 1$ iff $\{i,j\} \in E$.

Adjacency List We are given the degrees of the vertices and for every vertex $i \in V$ an array f_i of its neighbors. The value $f_i(j)$ is the j-th neighbor of i, in some arbitrary but fixed ordering.

2.2. Quantum Query Complexity

The quantum query complexity of a graph algorithm A is the number of queries to the adjacency matrix or to the adjacency list made by A. We will use the following tool, a result by Lin and Lin, to prove quantum query complexity upper bounds.

Theorem 1 ([1], Thm. 9)**.** *Suppose there is a classical algorithm $\mathcal{A}$ that computes $f(x)$ in T queries, and there is a guessing algorithm $\mathcal{G}$ that guesses the result of each query (0 or 1), making no more than an expected G mistakes for all inputs x. Then we can design a quantum algorithm that uses $O\left(\sqrt{TG}\right)$ queries to compute $f(x)$ with bounded error.*

This result relates classical query complexity to quantum query complexity and proves useful when an efficient classical algorithm for a problem exists. For more information about this result, see Appendix A.

3. *k*-Source Shortest Paths

We are given an unweighted graph G and a set S of sources with $|S| = k$. The k-source shortest paths problem is to find the shortest path from each vertex in s to each vertex in G. When $k = 1$, this is the single-source shortest paths problem, and when $k = n$ this is the all-pairs shortest paths problem. The best classical algorithm for SINGLE-SOURCE SHORTEST PATHS uses breadth-first or depth-first search and has query complexity $\Theta\left(n^2\right)$. Dürr et al. provide a quantum algorithm for SINGLE-SOURCE SHORTEST PATHS for weighted graphs with query complexity $O(n^{3/2}\log^{3/2} n)$ [2]. They also prove a lower bound of $\Omega\left(n^{3/2}\right)$, leaving a gap of log factors.

3.1. Upper Bound

In [1], Lin and Lin give an upper bound of $O(k^{1/2}n^{3/2})$ queries to an adjacency matrix for the k-source shortest paths problem for unweighted graphs. This improves the previous known bounds by removing the log factors. They apply Theorem 1 by using a depth-first search algorithm k times for $\mathcal{A}$ and a guessing algorithm which always guesses that the queried vertex pair is not an edge. Since depth-first search visits at most $n-1$ vertices (on any input), the algorithm makes at most $G = k(n-1)$ mistakes total and has the trivial query complexity $T = O\left(n^2\right)$. By Theorem 1 the corresponding bounded-error quantum algorithm therefore has query complexity $O(\sqrt{TG}) = O\left(k^{1/2}n^{3/2}\right)$.

3.2. Lower Bound

In [2], Dürr et al. prove a lower bound of $\Omega(\sqrt{nm})$ in the adjacency list model for SINGLE-SOURCE SHORTEST PATHS. Cai et al. prove a lower bound of $\Omega(n^2)$ for ALL-PAIRS SHORTEST PATHS [3]. We prove a lower bound of $\Omega(k^{1/2}n^{3/2})$ for k-SOURCE SHORTEST PATH by a reduction from MINIMUM FINDING, which is the problem of finding the minimum entry of each row of a matrix. Dürr et al. prove the following lower bound for MINIMUM FINDING:

Theorem 2 ([2], Thm. 19)**.** *In a matrix with c rows of d columns, finding the minimum value in each row takes $\Omega\left(c\sqrt{d}\right)$ queries.*

This theorem still holds when we restrict the matrix elements to be non-negative or to the set $\{0, 1\}$.

Theorem 3. *Finding the shortest paths from k source vertices in a graph requires $\Omega\left(\sqrt{\frac{km^2}{n}}\right)$ queries in the adjacency list model and $\Omega(k^{1/2}n^{3/2})$ queries in the adjacency matrix model.*

Proof. Let M be a matrix with ke rows and d columns, each entry $M_{i,j}$ being non-negative. By theorem 2, finding the minimum in each row of M takes $\Omega(ke\sqrt{d})$ queries. We construct the following weighted graph G (Figure 1).

Vertices: k vertices $u_i, i \in [k]$ in the first layer, kd vertices $g_{i,j}, i \in [k], j \in [d]$ in the second layer, and e vertices $v_r, r \in [e]$ in the third layer.

Edges: For each $i \in [k], j \in [d]$, there is an edge of weight 0 from u_i to $g_{i,j}$. For each $i \in [k], j \in [d], r \in [e]$, there is an edge from $g_{i,j}$ to e_r of weight $M_{(i-1)e+r,j}$.

The vertices u_i are the sources. Finding the shortest path from a particular u_i to a particular v_r is equivalent to findng the minimum value in the row $(i-1)e + r$ of M. Thus, finding the shortest path from each source to each vertex v_r requires yields the minimum of each row of M. By Theorem 2, this requires $\Omega(ke\sqrt{d})$ queries. G has $n = k+kd+e$ vertices and $m = kd+kde$ edges. Given n and m, we can choose d and e appropriately to get the required number of vertices and edges. As $m = \Theta(kde)$, and $n > kd$, we have a lower bound of $\Omega\left(\sqrt{\frac{km^2}{n}}\right)$ for k-SOURCE SHORTEST PATHS in the adjacency list model. Since m can be $\Theta(n^2)$, this gives a bound of $\Omega(k^{1/2}n^{3/2})$ for the adjacency matrix model. Theorem 2 holds even if the matrix M is restricted to contain only elements from $\{1, \infty\}$, which gives us the corresponding lower bound for unweighted graphs. □

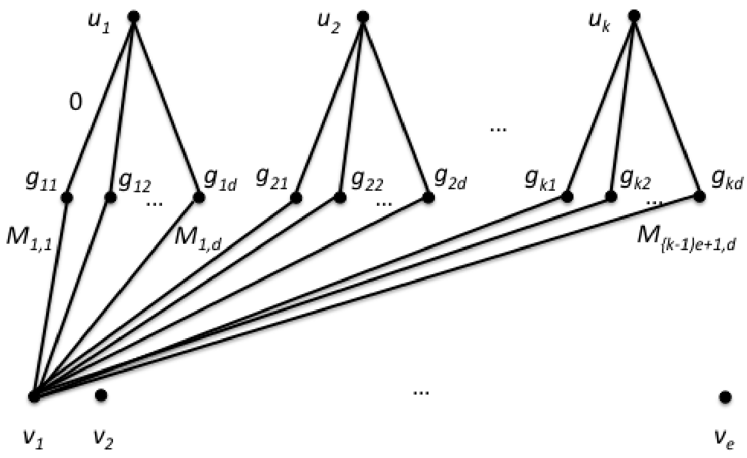

Figure 1: The graph for the reduction from MINIMUM FINDING to VERTEX COVER. The source vertices u_i are each connected to the d vertices in the corresponding group $g_{i,j}$ of vertices in the middle level. The vertices v_r in the bottom level are each connected to each vertex $g_{i,j}$ by an edge of weight $M_{(i-1)e+r,j}$.

This proves the optimality of the algorithm in [1].

4. Maximum Matching and Minimum Vertex Cover in Bipartite Graphs

In MAXIMUM MATCHING, we are given as input a graph G and are asked to find the set $M \subset E$ of maximum cardinality such that no two edges share a common endpoint. A related problem is MINIMUM VERTEX COVER: given a graph G, find the set $S \subset V$ of minimum cardinality such that each edge in the graph is adjacent to at least one vertex in S. For the case of bipartite graphs, König's theorem states that the minimum vertex cover has the same cardinality as the maximum matching. Lin and Lin provide a $O\left(n^{1.75}\right)$ query algorithm for MAXIMUM MATCHING for bipartite graphs [1]. This algorithm can be used to determine the size $|S|$ of the minimum vertex cover of a bipartite graph. Furthermore, we can construct a minimum vertex cover S given a maximum matching in a bipartite graph [4, pp. 74]. We will use Theorem 1 to show that we construct S given M using at most $O\left(n^{1.5}\right)$ additional queries to the adjacency matrix. Since finding the maximum matching takes $O(n^{1.75})$ queries, this gives an $O(n^{1.75})$ algorithm for MINIMUM VERTEX COVER in a bipartite graph.

4.1. Upper Bound

We are given the maximum matching M for a bipartite graph G. An unmatched vertex is a vertex that is not adjacent to any edge in M. An alternating path is a path in G that alternates between using edges in M and edges not in M. We use the following theorem:

Theorem 4 ([4], pp. 74)**.** *Let $G = (V, E)$ be a bipartite graph where the vertex set V is partitioned into a left set L and a right set R. Let M be a maximum matching for G. Let U be the set of unmatched vertices in L, and let Z be the set of vertices that are either in U or connected to U by alternating paths. Then, a minimum vertex cover of G is $S = (L \setminus Z) \cup (R \cap Z)$.*

We are given L, R, M. From M and L, we can find the set U without any queries. We can find Z by doing a breadth-first search from the set U, and alternating between using matched and unmatched edges in the search. To get an upper bound for the number of queries to find Z, we use Theorem 1. Our classical algorithm will make at most $T = O\left(n^2\right)$ queries (the trivial upper bound) to the adjacency matrix. Our guessing algorithm will make at most $G = O(n)$ mistakes. This gives an upper bound of $O(\sqrt{TG}) = O(n^{1.5})$ for the query complexity.

Algorithm 1. *Classical algorithm to find the minimum vertex cover S given the maximum matching M in a bipartite graph $G = (L \cup R, E)$:*

1. Find the set U of vertices not matched by M.
2. Initialize the set $S_0 = U$ and $L = V \setminus S_0$. Set $i = 1$.
3. Repeat while both L and S_{i-1} are non-empty:
   Case 1. i is odd.
   (a) For all vertices $v \in L$:
      i. Query edges (v, u) with $u \in S_{i-1}$ such that $(v, u) \notin M$. The moment a query returns 1, add v to S_i and remove v from L.

   Case 2. i is even.
   (a) For all vertices $v \in L$, use M to check if there is a $(v, u) \in M$ such that $u \in S_{i-1}$. If so, add v to S_i and remove v from L.
4. Set $Z = V \setminus L = \bigcup_i S_i$
5. Output $S = (L \setminus Z) \cup (R \cap Z)$.

Proof of correctness: We started with $S_0 = U$. If we define distance from U to be the length of the smallest alternating path required to reach a vertex v, then a simple inductive argument shows that S_i is the set of all vertices at distance i from U. Thus, we see that the union of all S_i is exactly the set Z of all vertices reachable from U using alternating paths. Then, by Theorem 4, set S is the required minimum vertex cover.

Theorem 5. *Finding the minimum vertex cover, given the maximum matching in a bipartite graph G takes $O\left(n^{1.5}\right)$ queries in the adjacency matrix model.*

Proof. Algorithm 1 needs to query at most $T = O\left(n^2\right)$ entires in the adjacency matrix. (This is the trivial query complexity upper bound for graph problems: if we have already queried an edge, we never have to repeat that query, and there are n^2 entries in the matrix in total.) We define our guessing algorithm to always guess that there is no edge. We observe that whenever we make a mistake on an edge (v, u) in step 3, the vertex v gets removed from L. Thus, as there are at most n vertices, we can make at most n mistakes. Thus, $G \leq n$. By Theorem 1, we have an upper bound of $O\left(\sqrt{TG}\right) = O\left(n^{1.5}\right)$ queries. □

Since the conversion from M to S can be done with $O\left(n^{1.5}\right)$ queries and M can be determined with $O\left(n^{1.75}\right)$ queries, we have an algorithm to determine S using $O\left(n^{1.75}\right)$ queries.

4.2. Lower Bound

In [5, pp. 4], Zhang proves a lower bound of $\Omega\left(n^{1.5}\right)$ for the decision problem of whether a bipartite graph G has a perfect matching. If we could find the minimum vertex cover in $o\left(n^{1.5}\right)$, we also get the size of the minimum vertex cover, which by König's theorem is the size of the maximum matching. Then, without making any extra queries, we can solve the decision problem of whether G has a perfect matching. Hence, we also get a lower bound of $\Omega\left(n^{1.5}\right)$ for finding the size of the minimum vertex cover in a bipartite graph.

5. Maximum Independent Set

An independent set in a graph is a set of vertices no two of which are adjacent. MAXIMUM INDEPENDENT SET is the problem of finding the largest such set. Since the maximum independent set is the complement of a vertex cover, Algorithm 1 can be extended to give a maximum independent set. Thus, the quantum query complexity of MAXIMUM INDEPENDENT SET in a bipartite graph is $O\left(n^{1.75}\right)$.

6. Discussion and Open Problems

Our results improve the bounds for some important graph problems. A summary of our and known results, as well as possible open problems, is given in Table 1.

Problem	Query Complexity	Open Problems
k-SOURCE SHORTEST PATHS	$\Theta(k^{1/2}n^{3/2})$	Proving bounds for adjacency list model. Generalizing to weighted graphs. Finding the time complexity of the algorithm given in [1].
MAXIMUM MATCHING	$\Omega(n^{1.5}), O(n^{1.75})$	Reducing the gap between the two bounds. Finding bounds for general graphs, possibly by applying Theorem 1 to the Micali-Vazirani algorithm for maximum matching in a general graph. Finding the time complexity of the algorithm given in [1].
MINIMUM VERTEX COVER	$\Omega(n^{1.5}), O(n^{1.75})$	Same as above.

Table 1: A table of relevant bounds. The bounds for MAXIMUM MATCHING and MINIMUM VERTEX COVER are for bipartite graphs, and the bounds for k-SOURCE SHORTEST PATHS are for general graphs.

Other open problems:

1. Can Theorem 1 be applied to more problems to improve or reproduce their query complexity bounds? We tried applying it to:
   (a) GRAPH COLLISION. The explicit algorithm used in the proof of Theorem 1 uses results proved by Kothari in [6]. In the same paper, Kothari uses his span program construction to remove the log factors in the upper bound for GRAPH COLLISION.

(b) PLANARITY TESTING.
(c) TRIANGLE FINDING.
(d) QUANTUM LOCAL SEARCH. The standard black box of making queries to the function isn't very useful as we need $\{0,1\}$ black-boxes. We experimented with blackboxes like "Is $f(x) < f(y)$?" and "Is $f(x)$ a local minimum?".

2. One limitation of applications of Theorem 1 is that they apply only to unweighted graphs. This is because the guessing algorithm makes binary guesses. If Lin and Lin's results were extended to allow non-binary queries, one could prove upper bounds for VERTEX COVER and MAXIMUM MATCHING on weighted bipartite graphs and k-SOURCE SHORTEST PATHS and MINIMUM SPANNING TREE on general weighted graphs. If the query, for example to an adjacency matrix, were not binary, as in the case of weighted graphs, we could simulate a guessing algorithm with a guessing algorithm for each bit of the input.

7. Acknowledgements

We would like to thank Cedric Yen-Yu Lin and Han-Hsuan Lin for helpful discussions and for suggesting open problems related to their work in [1]. Special thanks to Prof. Scott Aaronson for guiding us in this project for his class, Quantum Complexity Theory.

Appendix A. The Bomb Query Model

The bomb query model is a model of quantum computation inspired by the Elitzur-Vaidman bomb tester [7]. The premise of this model is that all oracle queries are replaced by a queries to a bomb circuit which executes a controlled query to the oracle, measures the result, and explodes if a 1 was queried [1]. Explosion terminates the algorithm, preventing successful evaluation of the function. Lin and Lin characterized the relation between the bomb query complexity $B(f)$ of a function f, which is the minimum number of bomb queries to the input required to compute f, and the quantum query complexity $Q(f)$.

Theorem 6 ([1], Thm. 1)**.** $Q(f) = \Theta\left(\sqrt{B(f)}\right)$.

Lin and Lin then prove the existence of an efficient bomb query algorithm to compute f using a randomized algorithm $\mathcal{A}$ for f and a guessing algorithm $\mathcal{G}$ that guesses queries to the input.

Theorem 7 ([1], Thm. 8)**.** *Suppose there is a classical algorithm $\mathcal{A}$ that computes $f(x)$ in T queries, and there is a guessing algorithm $\mathcal{G}$ that guesses the result of each query (0 or 1), making no more than an expected G mistakes for all inputs x. Then we can design a bomb query algorithm that uses $B(f) = O(TG)$ queries to compute $f(x)$ with bounded error.*

Thus, by Theorem 6, this gives a non-constructive upper bound to $Q(f) = \left(\sqrt{TG}\right)$.

Inspired by this result, Lin and Lin then give a constructive proof by explicitly providing a quantum algorithm that achieves the $O\left(\sqrt{TG}\right)$ bound by using $\mathcal{A}$ and $\mathcal{G}$ as subroutines. The quantum algorithm performs a Grover Search for the location of the first mistake made by the guessing algorithm, and then repeats the process for the remaining queries. To remove the log factors from their construction, they use a span program construction given by Kothari in [6].

Lin and Lin's results yield improved bounds for many problems. In particular, they improve the quantum query complexity upper bound for SINGLE-SOURCE SHORTEST PATHS in unweighted graphs to $O\left(n^{1.5}\right)$ in the adjacency matrix model, thus removing the log factors from previously known results. They also give a bound of $O\left(n^{1.75}\right)$ queries for MAXIMUM MATCHING for bipartite graphs. A common technique used by these algorithm's is to create a situation where the mistakes in the guessing algorithm lead to additions of edges to a tree. This limits the number of mistakes to $O(n)$, as there are $n-1$ edges in a tree.

References

[1] C. Y.-Y. Lin, H.-H. Lin, Upper Bounds on Quantum Query Complexity Inspired by the Elitzur-Vaidman Bomb Tester (2014). `arXiv:1410.0932`.

[2] C. Durr, M. Heiligman, P. Hoyer, M. Mhalla, Quantum query complexity of some graph problems (2004). `arXiv:quant-ph/0401091`.

[3] F. Cai, S. Tayu, S. Ueno, On the Quantum Query Complexity of All-Pairs Shortest Paths, in: Proceedings of the 2007 IEICE General Conference, 2007.

[4] J. Bondy, U. Murty, Graph Theory with Applications, North Holland, 1976.

[5] S. Zhang, On the Power of Ambainis's Lower Bounds (2003). `arXiv:quant-ph/0311060`.

[6] R. Kothari, An Optimal Quantum Algorithm for the Oracle Identification Problem, in: E. W. Mayr, N. Portier (Eds.), 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), Vol. 25 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 482–493. `arXiv:1311.7685v2`.

[7] A. Elitzur, L. Vaidman, Quantum Mechanical Interaction-Free Measurements, Foundations of Physics 23 (7) (1993) 987–997. `doi:10.1007/BF00736012`.