Exploring quantum analogs of the k-Canadian Traveler Problem

Nathan Sheffield

Massachusetts Institute of Technology

(Dated: September 23, 2023)

Abstract

The Canadian Traveler Problem concerns online s-t pathfinding in a subgraph of a known graph. It has been studied especially in the case where it is guaranteed that at most k edges of the known graph are missing in the true graph, which is referred to as the "k-Canadian Traveler Problem" or k-CTP. In this paper, we consider a relaxation of the problem in which edges can be queried arbitrarily. We show a lower bound of $\Omega(k \cdot d)$ queries in the classical setting, where d is the length of the shortest s-t path. We then give a straightforward quantum algorithm achieving $\tilde{O}(k \cdot \sqrt{d})$ query complexity. Analyzing the s-t connectivity span program algorithm of Belovs and Reichardt with the promise of at most k missing edges, we argue why one might expect a worst-case $\tilde{O}(\sqrt{k \cdot d})$ query complexity to be achievable. We prove this complexity in the special case where all paths in the graph are node-disjoint, show a corresponding lower bound, and then discuss potential approaches to work towards an algorithm on general graphs.

I. INTRODUCTION

The **Canadian Traveller problem** was first introduced by Papadimitriou and Yannakakis in 1991, who chose the name after hearing about truck drivers in remote parts of Canada who had to carefully plan routes because of the possibility of snow making roads impassable. The traditional formulation of the problem can be seen in FIG 1: a traveler is placed at a vertex s of a graph G', and aims to travel to a vertex t in as few steps as possible. The traveler has access to a map G, however G' is a subgraph of G that may be missing edges; the traveler discovers upon arriving at a vertex whether any of the edges leaving it are blocked.

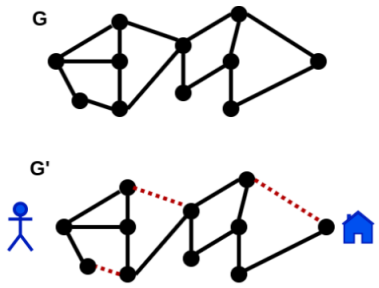

FIG. 1. A CTP instance; the traveler has access to the graph G, but may end up having to backtrack after discovering missing edges in G'.

The competitiveness of an algorithm for CTP is measured in terms of the true shortest s-t path in G'; the goal is to find an algorithm that minimizes the ratio of travel distance for the Canadian traveler to that an omniscient traveler with access to G'. Most work focuses on finding competitive strategies for k-CTP, where there are at most $k = O(1)$ blocked edges and k is known to the algorithm. This framework was first dealt with by Westphal, who gave a simple deterministic algorithm achieving competitive ratio $2k + 1$, as well as lower bounds of $2k + 1$ and $k + 1$ in the deterministic and randomized settings, respectively [1]. A subsequent paper by Xu et al showed a greedy strategy that improves on the $2k + 1$ bound in grid-like graphs [2]. Work by Bender and Westphal showed a tight $k + 1$ upper bound on randomized complexity in the case where all s-t paths are vertex-disjoint [3], and a 2014 result of Demaine et al showed a randomized strategy achieving $\left(1 + \frac{1}{\sqrt{2}}\right) k + 1$ competitive ratio on general graphs [4]. It remains open whether $k + 1$ is achievable in general.

II. STOCHASTIC S-T PATHFINDING MODEL

In order to consider k-CTP in a quantum setting, we'll define a straightforward related problem, which we refer to as the **stochastic s-t pathfinding model**:

Definition 1 (Stochastic s-t pathfinding)**.** The traveler is given a graph G, a start node s, a destination node t, and a promise that the true graph G' is equal to G but with up to k missing edges. The traveler has access to an oracle U, where

$$U(|i\rangle\,|b\rangle) = \begin{cases} |i\rangle\,|\bar{b}\rangle & \text{if the edge with index } i \text{ in } G' \text{ is unblocked} \\ |i\rangle\,|b\rangle & \text{otherwise} \end{cases}$$

In terms of k (the promised number of missing edges), and d (the length of the true shortest s-t path in G'), what is the minimum number of oracle queries required for the traveler to be able to identify an unblocked s-t path in the graph with success probability at least $\frac{2}{3}$?

This problem captures a fair amount of the standard k-CTP problem: we're asking for the difficulty of finding an s-t path with limited information, with the goal of doing well when the true shortest path was short. However, our definition notably differs from the standard k-CTP in that it allows for edges to be queried from arbitrary parts of the graph, instead of enforcing that the traveler must perform a walk and query only adjacent edges [5]. We'll now proceed to show that the classical lower bounds for k-CTP also hold in this model.

A. Classical Lower Bounds

To lower-bound the number of queries required by a classical algorithm, we'll consider a graph with **node-disjoint paths**, as in the lower bound from [1]. Consider a graph in which s and t are connected by $k+1$ disjoint paths, each of length d, shown in FIG 2. Now, suppose a deterministic classical algorithm has queried $k \cdot d - 1 = \Omega(k \cdot d)$ edges. It can be seen that an adaptive adversary can ensure that there remain two paths, each of which might be either blocked or unblocked, so that the deterministic algorithm cannot guarantee a solution.

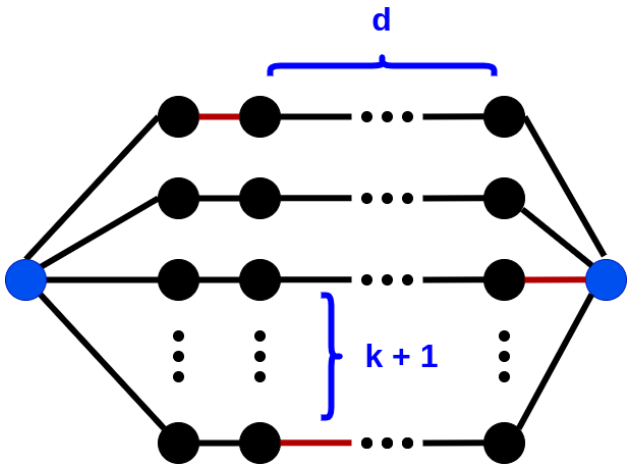

FIG. 2. A graph on which classical algorithms require $\Omega(k \cdot d)$ queries

The lower bound for randomized classical complexity is similar. We appeal to Yao's principle, arguing that a worst-case bound on a randomized algorithm can be obtained from an average-case bound on deterministic algorithms [6]. Consider the case where G is as in FIG 2. We choose the distribution of G' to be uniform across all subgraphs of G where k of the $k+1$ paths have exactly 1 missing edge. Suppose a deterministic algorithm has made $\frac{kd}{100} = \Omega(k \cdot d)$ edge queries. The probability that a given query discovers a blocked edge is upper bounded by $\frac{2}{d}$ if the query is to a path that has received at most $\frac{d}{2}$ previous queries. Even if it were the case that the $\left(\frac{d}{2}+1\right)$st query to a path always revealed a blocked edge, this would give us an upper bound of $\frac{2k}{50}$ on the expected number of edges revealed. By Markov's inequality, this means the probability that the algorithm has already discovered more than $\frac{k}{5}$ blocked edges is at most $\frac{1}{5}$. Now, assuming at most $\frac{k}{5}$ blocked edges have been discovered, consider the probability that the algorithm has revealed more than half the edges in the true unblocked path. All $\frac{4k}{5}$ as-yet-unblocked paths are a priori equally likely to be the true unblocked path, so the choice of which ones to start making more than $\frac{d}{2}$ queries is necessarily arbitrary, and thus with probability $\frac{39}{40}$ the algorithm has revealed no more than half the edges in the true unblocked path. But now, in this case, even if the algorithm knew the true unblocked path was one of the ones it had made at most $\frac{d}{2}$ queries to, it would have at least $\frac{k}{2}$ options to choose from, each of whose probability of being the true unblocked path differs by at most a factor of two. So, it will guess correctly with probability at most $\frac{4}{k}$. But $\frac{1}{5} + \frac{1}{40} + \frac{4}{k} < \frac{2}{3}$ in the limit of large k, which shows a lower bound of $\Omega(k \cdot d)$ randomized query complexity.

B. Quadratic improvement in d

Given access to quantum computation, however, it is possible to do better than $O(k \cdot d)$ in general. A simple application of Grover's algorithm will in fact allow us to achieve $\tilde{O}(k \cdot \sqrt{d})$. Specifically, we will use the fact that, given a domain of size N, there exists a quantum search algorithm requiring $O(\sqrt{n \log 1/\varepsilon})$ queries that finds a solution with probability at least $1-\epsilon$ if one exists [7]. The following algorithm solves k-CTP in the stochastic connectivity model with $\tilde{O}(k\sqrt{d})$ query complexity (it is roughly analogous to the classical BACKTRACK algorithm proposed in [1]):

Algorithm 1 Quantum "backtrack" algorithm

$M \leftarrow G$ represents the map known to the traveler
$r \leftarrow 0$ represents the number of blocked edges the traveler has revealed
while $r < k$ **do**
    Find the shortest path, P, in M
    Quantum search on P long enough find a blocked edge with $1 - \frac{1}{4k}$ probability if it exists
    if a blocked edge e has been found **then**
        $M \leftarrow M \setminus \{e\}$
        $r \leftarrow r + 1$
    else
        Return P
Return the shortest unblocked path in M

This algorithm runs for at most k iterations, and on each iteration makes an error (i.e. returns a path with a blocked edge) with probability at most $\frac{1}{4k}$, so the total probability of error is at most $\frac{1}{4}$. Since we always search the shortest unblocked path in the graph, we are guaranteed to only run searches on paths of lengths less than or equal to the true shortest path length d. So, the total number of required queries is $O(k\sqrt{d \log 4k}) = \tilde{O}(k\sqrt{d})$.

We might wonder whether this quadratic speedup in d is in fact optimal, or whether it is possible to improve upon this somewhat trivial algorithm. The following analysis hints that it may be possible to achieve $\tilde{O}(\sqrt{k \cdot d})$, but fails to show an algorithm doing so in general.

C. Alternate analysis of the span program for s-t connectivity

Our key tool will be Belovs and Reichardt's span program for s-t connectivity. A span program is a model of computation introduced by Karchmer and Wigderson for understanding classical complexity [8]. They have since become an important tool in understanding quantum complexity due to a result of Reichardt that span program witness sizes provide upper bounds for quantum query complexity [9]. We state this result, and the definition of span programs, in appendix A.

Belovs and Reichardt subsequently developed an span program for checking s-t connectivity, which they analyzed to find an efficient quantum connectivity algorithm. This span program is as follows:

- The vector space V has one basis element associated to each vertex in the graph
- The target vector is chosen to be $|t\rangle - |s\rangle$, where s and t are the source and destination vertices, respectively
- An input x encodes for each pair of vertices (u, v) whether the edge (u, v) exists in the graph. $I^{(0)}_{(u,v)} = \{\}$, $I^{(1)}_{(u,v)} = |v\rangle - |u\rangle$

In their analysis, Belovs and Reichardt observe that, if the domain of x is restricted such that s and t are connected only if they're connected by a path of length $\leq d$, the max positive witness size is $O(d)$ (witnessed by an unblocked s-t path; the sum along this path is $|t\rangle - |s\rangle$), and the max negative witness size is $O(n^2)$ (witnessed by an s-t cut; the sum of the vectors in t's connected component minus the sum of the vectors in s's connected component has dot product 1 with $|t\rangle - |s\rangle$ but 0 with all available vectors), so a quantum algorithm can solve this problem with $O(n\sqrt{d})$ queries [10].

We are interested, however, in the case where we know that G' is equal to G with up to k missing edges. In this case, we can analyze the span program slightly differently. Once again, with a promise that the shortest path from s to t is of length at most d if it exists, we get maximum positive witness size at most d. But now, if we have sets $I_{(u,v)}$ only for $(u, v) \in G$, we are guaranteed to only ever have up to k unavailable vectors. So, the number of unavailable vectors crossing an s-t cut is at most k, and each has dot product at most 2 with with the witness vector, so the maximum positive witness size is $4k$. Thus, given a

promise that no more than k edges are missing from G, we can check s-t connectivity with $O(\sqrt{k \cdot d})$ queries.

III. QUANTUM ALGORITHM FOR NODE-DISJOINT PATH GRAPHS

This s-t connectivity result seems initially like it should directly give us the desired complexity for our problem. However, there's a couple of caveats. The first is that our complexity is in terms of a *promise for d*. That is, if we know ahead of time that s and t will have a path of length at most d if they're connected, then we can check connectivity in few queries with bounded error probability. However, if we don't a priori know for certain a bound on their distance, the fact that they might "happen" to be close anyway doesn't help us – we needed to have known before we ran the algorithm. The second issue is that while this algorithm can *check* connectivity, it won't actually return a path when s and t are connected. There does not seem to be an obvious way to reduce the search problem to that decision problem.

These caveats mean that we are not able in this paper to provide an $\tilde{O}(\sqrt{k \cdot d})$ algorithm for the general case. However, we will show how to overcome these issues in the case where all paths are node-disjoint (which was the case where we found our classical lower bounds), which gives some hope for a general algorithm. Both of the methods we use will be variants of binary search, meaning that our final algorithm will be performing two nested binary searches.

A. Checking connectivity without a promise of d

Lemma 1 (Better s-t connectivity complexity on node-disjoint paths)**.** Suppose we're given a graph G where all simple s-t paths have disjoint vertices from each other, and a promise that G' consists of G with at most k blocked paths [11]. There exists a quantum algorithm that tests s-t connectivity in G' using $O(\sqrt{k \cdot d^*} \log d^*)$ queries to the adjacency matrix G', where d^* is the length of the true shortest s-t path in G', or the maximum length s-t path in G if no connection exists in G'.

Proof. Consider a binary search as in Algorithm 2.

Since the connectivity span program has bounded two-sided error, given an appropriate

Algorithm 2 Connectivity algorithm for node-disjoint path graphs

$d \leftarrow 1$
while d is less the twice the longest path length in G **do**
    $G_d \leftarrow$ the subgraph of of G consisting of all s-t paths with length at most d
    for $C \cdot \log d$ iterations **do**
        Run the connectivity span program on G_d, with promises of $\leq k$ missing edges and shortest path length $\leq d$ if a path exists
    if the majority of iterations say G_d is connected **then**
        Return `TRUE`
    else
        $d \leftarrow 2 \cdot d$
Return `FALSE`

choice of constant C we can ensure that each execution of the while loop has probability at most $\frac{1}{3} \cdot 2^{-d}$ of giving an erroneous result. So, the total error probability is bounded by $\frac{1}{3}$. By our analysis of the span program algorithm, each execution of the while loop requires $O(\sqrt{k \cdot d} \log d)$ oracle queries. If s and t are connected by a path of length d^*, the total number of queries required is therefore of order at most

$$\sum_{i=0}^{\lceil \log_2 d^* \rceil} \sqrt{k \cdot 2^i} \cdot i \leq 2\sqrt{k \cdot d^*} \log d^* \left(\sum_{i=0}^{\lceil \log_2 d^* \rceil} 2^{-i/2} \right) = O(\sqrt{k \cdot d} \log d^*)$$

By an identical argument, if s and t are disconnected, the total number of queries is $O(\sqrt{k \cdot d} \log d^{\max})$, where $d^{\max}$ is the longest s-t path in G. □

We now use this improved connectivity decision algorithm as a subroutine for a path search algorithm.

B. Search to decision reduction

Lemma 2 (Solution to stochastic connectivity k-CTP on node-disjoint path graphs)**.** Let G be a graph where all simple s-t paths have disjoint vertices from each other, and let G' be a subgraph of G with at most k deleted edges, and shortest s-t path length d^*. There exists an algorithm that, given G, for all G', identifies an unblocked s-t path in G' with probability at least $\frac{2}{3}$, and makes $O(\sqrt{k \cdot d^*} \log d^* \log \log k)$ queries to the edge oracle for G'.

Proof. Once again, the solution is to binary search, but now over subsets of the paths as opposed to guesses of the shortest path length. Note importantly that we need only consider the $(k+1)$ shortest paths in G, because at least one of those must be unblocked in G'. The procedure is stated precisely in Algorithm 3.

Algorithm 3 Pathfinding algorithm for node-disjoint path graphs

$S \leftarrow G$ represents the subgraph we recurse over
Remove all but the shortest $k+1$ s-t paths from S
while S consists of more than 1 s-t path **do**
    $p \leftarrow$ number of s-t paths in S
    $S_{\text{low}} \leftarrow$ subgraph of S consisting of shortest $\left\lfloor \frac{p}{2} \right\rfloor$ paths
    $S_{\text{high}} \leftarrow$ subgraph of S consisting of longest $\left\lceil \frac{p}{2} \right\rceil$ paths
    for $C \cdot \log\log k$ iterations **do**
        Run our connectivity algorithm on S_{low}, with a promise of $\leq \left\lfloor \frac{p}{2} \right\rfloor$ blocked paths
    if the majority of iterations say S_{low} is connected **then**
        $S \leftarrow S_{\text{low}}$
    else
        $S \leftarrow S_{\text{high}}$
Return the single path in S

Through appropriate choice of C, we ensure that the algorithm errs with probability at most $\frac{1}{3\log(k+1)}$ each execution of the while loop (since our connectivity algorithm has bounded error). Thus, over the $\log(k+1)$ executions of the while loop, we accumulate total error probability bounded by $\frac{1}{3}$. Now, note that since we always check connectivity on the shorter half of the paths first, whenever we check connectivity on a subgraph not containing the shortest unblocked path, it has longest path shorter than the shortest unblocked path. So, running our connectivity algorithm always takes $O(\sqrt{\frac{p}{2} \cdot d^*} \log d^*)$ queries. Since p is halved every execution of the while loop, the total query complexity is of order

$$\sqrt{k \cdot d^*} \log d^* \log\log k \left(\sum_{i=0}^{\log k} \frac{k}{2^i} \right) = O(\sqrt{k \cdot d^*} \log d^* \log\log k)$$

□

So we have found an $\tilde{O}(\sqrt{k \cdot d})$ algorithm for this model of k-CTP on graphs with node-

disjoint paths. These techniques do not apply to general graphs, but the fact that we've shown good performance on the graphs from which classical lower bounds were derived suggests there could be a possibility of improvement in general. In the next section, we'll show that as long as k and d are similar in magnitude, no polynomial improvement on this algorithm is possible.

C. Quantum lower bound

To show that a $\Omega(\sqrt{k \cdot d})$ lower bound holds in the case of node-disjoint paths, we apply a strategy called the **adversary method**, introduced by Ambainis in 2000 [12]. This method is effectively the dual of the span program method for upper bounds. Ambainis's statement of the bound is described in Appendix B. Recall the node-disjoint path graph of FIG 2, with $k+1$ paths all of length d. Using notation as in Ambainis's statement, we let X be the set of inputs where all paths are blocked in exactly one edge except for the top path, and Y be the set of inputs where all paths are blocked in exactly one edge except for a single path among the bottom k paths. $R(x, y)$ holds when x and y differ in exactly 2 places. For any x, there are $k \cdot d$ ways to reach a y by modifying 2 bits (choose one of the d edges along the top path to block; choose a path among the bottom k to unblock), and symmetrically $k \cdot d$ x's in relation with any given y. Then, note that for any given index, there are at most $\max(k, d)$ y's in relation to x such that $x_i \neq y_i$, and vice versa for x's in relation to a given y. So, the basic adversary method gives an $\Omega(\min(k, d))$ lower bound on query complexity for our model of k-CTP. We suspect with a more sophisticated adversary argument it would be possible to show $\Omega(\sqrt{k \cdot d})$.

IV. DISCUSSION, APPLICATIONS AND NATURAL FURTHER DIRECTIONS

The question of how to get from our algorithm on node-disjoint paths to an equally good algorithm on general graphs is a not an obvious one. However, one intermediate case that may be worth studying is that of **apex trees**. An apex tree consists of a tree rooted at t, where each leaf of the tree is then connected by a single edge to s. An algorithm for k-CTP on apex trees is the main subroutine used in the current best classical algorithm to solve it generally [4], so this would seem to be a natural next step to consider.

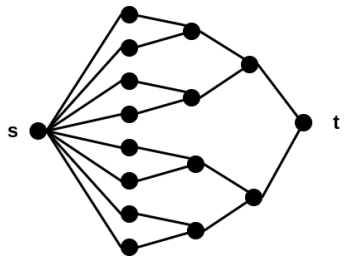

FIG. 3. A complete binary s-t apex tree

Apex trees are a good challenge to our method, because although they're very structured, the number of s-t paths grows exponentially in d. So, simple binary search doesn't seem feasible. One observation worth making, though, is that it's possible we can get a better analysis of our span program connectivity algorithm in some cases. As observed by Jeffery et al, the witness complexity of the s-t connectivity span program is determined by the effective resistance and capacitance of the input graph [13]. In particular, this means that if we can guarantee that whenever s and t are connected there are many different unblocked s-t paths, we can achieve improved runtime for checking connectivity, which seems relevant in the case of complete apex trees where, unless there are blockages near the root, there will be many s-t paths.

Even if we are unable to find generally applicable methods, it can also be worth studying special cases of independent interest. For instance, in a 2019 paper Jeffery and Kimmel observed a reduction of the problem of evaluating Boolean formulas composed of ANDs and ORs to s-t connectivity testing on a certain class of planar graphs [14] [15]. In the light of this interpretation, our result about node-disjoint paths yields the following corollary:

Lemma 3. Given a positive CNF formula ϕ on variables $x_1 \dots x_n$, if it is promised that the values of $x_1 \dots x_n$ differ in at most k places from known values $y_1 \dots y_n$, and $y_i \geq x_i$ for all i, there exists an algorithm that with bounded error probability finds the smallest violated clause in $\tilde{O}(\sqrt{k \cdot d})$ quantum queries to x, where d is the size of this clause.

Thinking of clauses as paths and literals as edges, this follows directly from our above work. Investigation of our k-CTP model on broader classes of graphs could yield stronger formula evaluation results. Another potentially worthwhile direction would be to try and solve the node-disjoint path case when it's possible for G' to have up to k edges not in G but in an even larger pre-specified graph H – that is, we allow for addition as well as removal of

specific edges. If we could prove query complexity upper bounds on particular cases of that problem, they could translate to query complexity upper bounds on CNF formula evaluation without positivity or monotonicity constraints.

V. CONCLUSION

In this work, we defined a variant of path-finding problem in which the graph is known ahead of time up to a constant number of edge deletions. We demonstrated an quantum improvement from $\Omega(k \cdot d)$ query complexity in the classical case to $O(k\sqrt{d \log k})$ in general, and $O(\sqrt{k \cdot d} \log d \log \log k)$ when all paths are node-disjoint. We showed an almost-tight lower bound in the node-disjoint case when $k \approx d$, but left open the question of the complexity of the general case. We then discussed potential strategies to approach the general case, and considered connections between this problem and Boolean formula evaluation.

This research is motivated by the fundamental importance of path-finding in graphs across algorithms theory and practice. Prior work has established that quantum computation can allow path-finding in polynomially less time and fewer accesses to the graph than the classical limits. We believe that further work studying hardness of path-finding under parameterizations of the input is valuable for improving our understanding of what allows this quantum speedup and what additional restrictions a quantum computer can exploit to do better. This paper represents the initial steps towards understanding a particular direction of this parameterization, inspired by potential applications like packet routing in an unreliable communication network. There remains much related work to be done, both in resolving the questions raised in this paper, and in exploring path-finding through different novel lenses.

Appendix A: Span program definitions

Definition 2 (Span programs [8])**.** A **span program** consists of a finite vector space V, a target unit vector $|T\rangle \in V$, n 0-sets $I_1^{(0)}, \ldots, I_n^{(0)} \subseteq V$, and n 1-sets $I_1^{(1)}, \ldots, I_n^{(1)} \subseteq V$. This program is said to accept an input $x \in \{0,1\}^n$ if

$$|T\rangle \in \text{span} \bigcup_{i \in \{1,\ldots,n\}} I_i^{(x_i)}$$

Theorem 1 (Relationship between span programs and query complexity [9])**.** If a span program accepts an input x, we say that the **positive witness size** for the program on x is the minimum square magnitude $\langle w|w\rangle$ among vectors $|w\rangle$ such that $A|w\rangle = |T\rangle$, where A is the matrix whose columns are the available vectors under x.

If the span program rejects an input x, we say that the **negative witness size** for x is the minimum value of $\langle w| BB^\dagger |w\rangle$, where the columns of B are all vectors in any $I_i^{(b)}$, and $|w\rangle$ is some vector such that $\langle w|T\rangle = 1$ but $\langle w|v\rangle = 0$ for all available vectors v under x.

If there exists a span program computing a function P on binary inputs x, there exists a two-sided bounded-error quantum algorithm to compute P while making

$$O\left(\sqrt{\max_{x \text{ accepted}} \text{pos witness size of } x \max_{x \text{ rejected}} \text{negative witness size of } x}\right)$$

queries to bits of x.

Appendix B: Statement of basic adversary bound from [12]

Theorem 2 (Ambainis, adversary bound)**.** Let $f(x_1, \ldots, x_N)$ be a function of n $\{0,1\}$-valued variables and X, Y be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ be such that

1. For every $x \in X$, there exist at least m different $y \in Y$ such that $(x, y) \in R$.
2. For every $y \in Y$, there exist at least m' different X such that $(x, y) \in R$.
3. For every $x \in X$ and $i \in \{1, \ldots, n\}$, there are at most l different $y \in Y$ such that $(x, y) \in R$ and $x_i \neq y_i$
4. For every $y \in Y$ and $i \in \{1, \ldots, n\}$, there are at most l' different $x \in X$ such that $(x, y) \in R$ and $x_i \neq y_i$

Then, any quantum algorithm computing f uses $\Omega(\sqrt{\frac{mm'}{ll'}})$ queries.

---

[1] S. Westphal, A note on the k-canadian traveller problem, Information Processing Letters **106**, 87 (2008).

[2] Y. Xu, M. Hu, B. Su, B. Zhu, and Z. Zhu, The canadian traveller problem and its competitive analysis, Journal of Combinatorial Optimization **18**, 195 (2009).

[3] M. Bender and S. Westphal, An optimal randomized online algorithm for the k-canadian traveller problem on node-disjoint paths, J. Comb. Optim. **30**, 87–96 (2015).

[4] E. D. Demaine, Y. Huang, C.-S. Liao, and K. Sadakane, Canadians should travel randomly, in *Automata, Languages, and Programming*, edited by J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014) pp. 380–391.

[5] An alternative definition, which we call the **location register model**, enforces these locality constraints by giving the oracle a portion of the Hilbert space not accessible to the algorithm, which encodes the traveler's location. This model seems to lend itself to interesting variants of quantum walks, and Prof. Chuang observes there may be strategies involving Elitzur-Vaidman measurements [16]. However, as our results here have been very limited, we will not discuss them in this paper.

[6] A. C.-C. Yao, Probabilistic computations: Toward a unified measure of complexity, 18th Annual Symposium on Foundations of Computer Science (sfcs 1977) , 222 (1977).

[7] C. Dürr, M. Heiligman, P. HOyer, and M. Mhalla, Quantum query complexity of some graph problems, SIAM Journal on Computing **35**, 1310 (2006).

[8] M. Karchmer and A. Wigderson, On span programs, in *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference* (1993) pp. 102–111.

[9] B. W. Reichardt, Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function, in *2009 50th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, 2009).

[10] A. Belovs and B. W. Reichardt, Span programs and quantum algorithms for st-connectivity and claw detection, in *Algorithms – ESA 2012* (Springer Berlin Heidelberg, 2012) pp. 193–204.

[11] Note that technically ensuring at most k blocked edges is stronger than ensuring at most k blocked paths in the node-disjoint path case, however our analysis of the span program still holds under this assumption.

[12] A. Ambainis, Quantum lower bounds by quantum arguments (2000), arXiv:quant-ph/0002066 [quant-ph].

[13] M. Jarret, S. Jeffery, S. Kimmel, and A. Piedrafita, enQuantum algorithms for connectivity and related problems 10.4230/LIPICS.ESA.2018.49 (2018).

[14] S. Jeffery and S. Kimmel, Quantum algorithms for graph connectivity and formula evaluation (2019), arXiv:1704.00765 [quant-ph].

[15] I originally had a large section of the paper on this topic because I thought I'd come up with this idea, but stumbled upon this prior work a couple of days ago. Such is life.

[16] L. Vaidman, The elitzur-vaidman interaction-free measurements (2008), arXiv:0801.2777 [quant-ph].