View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Oracles with Costs

Shelby Kimmel[1,2], Cedric Yen-Yu Lin[2], and Han-Hsuan Lin[2]

1 **Joint Center for Quantum Information and Computer Science, University of Maryland, US**
2 **Center for Theoretical Physics, Massachusetts Institute of Technology, US**

Abstract

While powerful tools have been developed to analyze quantum query complexity, there are still many natural problems that do not fit neatly into the black box model of oracles. We create a new model that allows multiple oracles with differing costs. This model captures more of the difficulty of certain natural problems. We test this model on a simple problem, Search with Two Oracles, for which we create a quantum algorithm that we prove is asymptotically optimal. We further give some evidence, using a geometric picture of Grover's algorithm, that our algorithm is exactly optimal.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Quantum Algorithms, Query Complexity, Amplitude Amplification

Digital Object Identifier 10.4230/LIPIcs.TQC.2015.1

1 Introduction

The standard oracle model is a powerful paradigm for understanding quantum computers. Tools such as the adversary semidefinite program [12, 13], learning graphs [5, 6], and the polynomial method [4] allow us to accurately characterize the quantum query complexity [1, 7] of many problems of interest.

However, the oracle model does not capture the full power or challenges of quantum computing. For example, problems such as k-SAT do not fit easily into the oracle model. Additionally, while the query complexity of the hidden subgroup problem is known to be polynomial in the size of the problem [11], for some non-abelian groups there is no efficient algorithm.

In this paper, we describe a variation of the oracle model. We have access to two oracles, rather than a single oracle[1], but one oracle is more expensive to use. In the standard oracle model, the figure of merit is the query complexity, which is the minimum number of queries needed to an oracle to evaluate a function. In our model, the figure of merit is the *cost complexity*, which is the minimum cost needed to evaluate a function using multiple oracles with different costs.

To motivate this model, we consider the following fact: in some search problems we want to find an element in a set that satisfies a property that is expensive to test. However, often another less expensive test is available that can narrow down the search range but is not conclusive. We give three examples of problems where such less expensive, less conclusive tests are natural. In each example, Test 1 is more expensive to run but is conclusive, while Test 2 is cheaper to run but allows some non-solutions to pass.

[1] The model can easily be extended to more than two oracles, but for simplicity, we limit ourselves to two.


© Shelby Kimmel, Cedric Yen-Yu Lin, and Han-Hsuan Lin;
licensed under Creative Commons License CC-BY
10th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2015).
Editors: Salman Beigi and Robert König; pp. 1–26

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- In the problem of k-SAT on n bits, we would like to find an assignment $x \in \{0,1\}^n$ such that all clauses are satisfied. Consider an algorithm for k-SAT that runs two types of tests on a possible assignment x:
  1. Check whether all clauses are satisfied.
  2. Check whether some subset of the clauses are satisfied.
- Given a graph A and a set of graphs $\{B_1, \cdots, B_p\}$, we would like to find a graph B_i isomorphic to A. Consider an algorithm that runs two types of tests on a graph B_i:
  1. Check whether B_i is isomorphic to A (say by brute force search).
  2. Check whether the adjacency matrices of B_i and A have the same spectrum.
- In the decision variant of the traveling salesman problem, given a positively weighted N-graph G and a positive number b, we would like to find a tour of the vertices of G that uses cost no more than b. Given a partial tour of length $N/2$, we can run two types of tests:
  1. Check whether the partial tour can be completed to an N-vertex tour that has cost at most b, by using brute force search.
  2. Check whether the sum of the weights of the $N/2$ edges traversed in the partial tour is bigger than b.

In all three examples, the two tests can be implemented as unitaries $\mathcal{O}_1, \mathcal{O}_2$ that act as $\mathcal{O}_i|x\rangle|y\rangle = |x\rangle|y \oplus f_i(x)\rangle$. Here $f_i(x) = 1$ if assignment x passes Test i and $f_i(x) = 0$ otherwise. These two unitaries will play the role of oracles with different costs.

None of the problems listed above are typically thought of as oracle problems, because in each problem, there is more information than can easily be incorporated into a single oracle. However, with multiple oracles, the information can be distributed among different oracles. Using different costs for different oracles allows us to include information about the time required to access information. We see that cost complexity can capture certain aspects of a problem that can not be easily accounted for in the standard oracle model; we hope this model will provide new insight into problems previously thought beyond the tools of query algorithms. We note that we do not expect these techniques to allow us to solve NP-complete problems in polynomial time. Rather, our goal is to potentially improve upon existing exponential time algorithms, and create connections between standard oracle problems and problems that seem far from typical oracle problems.

Problems such as those described above can easily be recast into an oracle problem, which we call Search with Two Oracles (STO). In this work, we focus on the problem of STO. We tightly characterize the quantum cost complexity of this problem, and give several techniques for putting lower bounds on quantum cost complexity. We also show that the cost complexity of STO is the same whether or not the oracles can be accessed using a control operation; that is, accessing the oracles in superposition gives no added power.

We also attempt to exactly bound (rather than asymptotically bound) the cost complexity of STO. Usually, one is not particularly interested in proving exact optimality, but we have several reasons for wanting to explore this problem. Few quantum algorithms are known to be exactly optimal; Grover's algorithm and parity are two examples [10, 4]. STO is a very simple extension of a standard search problem, so it seems like a good candidate problem for obtaining another exact lower bound. Proving that our algorithm is exactly optimal would provide evidence that amplitude amplification is exactly optimal in the case of no additional structure (i.e. when we treat the base algorithm as a black box). Additionally, while we can obtain asymptotically tight bounds for the problem of STO, for a simple extension of STO to $\log N$ oracles (where N is the size of the search space), these techniques fail. However, if

we could obtain tighter bounds for STO, we should be able to get a better characterization of the cost complexity for these more complex problems.

Finally, we compare the quantum cost complexity of STO to the classical cost complexity. We show a polynomial reduction in cost for the quantum version. Moreover, we show that the optimal quantum and classical algorithms behave qualitatively differently, highlighting the power of quantum algorithms.

In Section 2, we describe cost complexity and define STO. In Section 3, we describe optimal quantum algorithms for STO, and in Section 4, we put lower bounds on the cost complexity of STO. Finally, we look at the classical cost complexity of STO in Section 5.

2 Cost Complexity, STO, and Relation to Previous Work

Cost complexity is very closely related to query complexity. For background on query complexity, see [1, 7].

We first define cost complexity. In the following, we use the notation $[N] \equiv \{1, \ldots, N\}$. Given the input $(f_1, f_2) \in D$, which is a pair of functions $f_1, f_2 : [N] \to \{0, 1\}$, we want to calculate F where $F : D \to \{0, 1\}$. Let f_1 be associated with cost c_1 and f_2 be associated with cost c_2. Depending on the type of algorithm (e.g. classical, quantum), these two functions are accessed in different ways.

In the classical setting, consider a randomized classical algorithm $\mathcal{A}_c$ for F that makes q_1 queries to f_1, and q_2 queries to f_2. Then the cost of this algorithm is

$$\text{Cost}(\mathcal{A}_c) = q_1 c_1 + q_2 c_2. \tag{1}$$

Let $\mathcal{A}_{c,\epsilon}$ be the set of randomized classical algorithms that solve F with success probability at least $1 - \epsilon$ on all inputs in D. Then the *classical randomized cost complexity (RCC)* of F is

$$RCC_\epsilon(F) = \min_{\mathcal{A}_c \in \mathcal{A}_{c,\epsilon}} \text{Cost}(\mathcal{A}_c). \tag{2}$$

In the quantum setting, let $\mathcal{O}_1$ and $\mathcal{O}_2$ be unitaries acting on the Hilbert space $\mathbb{C}^N$ with standard basis states $|i\rangle$ for $i \in [N]$ as $\mathcal{O}_j|i\rangle = (-1)^{f_j(i)}|i\rangle$ for $j \in 1, 2$. Consider a quantum algorithm $\mathcal{A}_q$ that at each time step, can apply $\mathcal{O}_1$ or $\mathcal{O}_2$ or some other unitary that is independent of f_1 and f_2, and which makes q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. Then the cost of the algorithm $\mathcal{A}_q$ is

$$\text{Cost}(\mathcal{A}_q) = q_1 c_1 + q_2 c_2. \tag{3}$$

Let $\mathcal{A}_{q,\epsilon}$ be the set of quantum algorithms that solve F with success probability at least $1 - \epsilon$ on all inputs in D. Then the *quantum cost complexity (QCC)* of F is

$$QCC_\epsilon(F) = \min_{\mathcal{A}_q \in \mathcal{A}_{q,\epsilon}} \text{Cost}(\mathcal{A}_q). \tag{4}$$

Finally, we consider quantum algorithms that can access oracles in superposition. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be as above, and let $\mathcal{O}_0 = \mathbb{I}$, the $N \times N$ identity matrix. We now consider a quantum algorithm that has access to a controlled operation $C\mathcal{O}$ that acts on the the Hilbert space $\mathbb{C}^3 \otimes \mathbb{C}^N \otimes \mathbb{C}^V$ ($\mathbb{C}^V$ is a workspace register) with standard basis states $|b\rangle|i\rangle|v\rangle$ for $i \in [N]$, $v \in [V]$, and $b \in \{0, 1, 2\}$ as $C\mathcal{O}|b, i\rangle = |b\rangle\mathcal{O}_b|i\rangle|v\rangle$. Suppose the encoded functions are f_1 and f_2. Then if an algorithm $\mathcal{A}_{qs}$ applies $C\mathcal{O}$ a total of T times over the course of the algorithm to states

$$|\eta^t_{f_1,f_2}\rangle = \sum_{b=0}^{2} \sum_{i=1}^{N} \sum_{v=1}^{V} \alpha^t_{f_1,f_2}(b, i, v)|b, i, v\rangle \tag{5}$$

for $t \in [T]$, the cost of the algorithm is

$$\mathrm{Cost}(\mathcal{A}_{qs}) = \max_{f_1,f_2} \sum_{t=1}^{T} \kappa(\eta^t_{f_1,f_2}) \text{ where}$$
$$\kappa(\eta^t_{f_1,f_2}) = \begin{cases} c_1 \text{ if } \sum_{i,v} |\alpha^t_{f_1,f_2}(1,i,v)|^2 \neq 0, \\ c_2 \text{ if } \sum_{i,v} |\alpha^t_{f_1,f_2}(1,i,v)|^2 = 0 \text{ and } \sum_{i,v} |\alpha^t_{f_1,f_2}(2,i,v)|^2 \neq 0, \\ 0 \text{ if } \sum_{i,v} |\alpha^t_{f_1,f_2}(1,i,v)|^2 = 0 \text{ and } \sum_{i,v} |\alpha^t_{f_1,f_2}(2,i,v)|^2 = 0. \end{cases} \tag{6}$$

Let $\mathcal{A}_{qs,\epsilon}$ be the set of quantum algorithms using $C\mathcal{O}$ that solve F with success probability at least $1-\epsilon$ on all inputs in D. Then the *controlled quantum cost complexity (ConQCC)* of F is

$$ConQCC_\epsilon(F) = \min_{\mathcal{A}_{qs} \in \mathcal{A}_{qs,\epsilon}} \mathrm{Cost}(\mathcal{A}_{qs}). \tag{7}$$

The controlled quantum cost complexity is closely related to the time required in the model of variable times introduced by Ambainis in [2].

Note that

$$ConQCC_\epsilon(F) \leq QCC_\epsilon(F) \leq RCC_\epsilon(F). \tag{8}$$

For any of the cost complexities described above, if we do not include a subscript ϵ, then the cost is assumed to apply for the case $\epsilon = 1/3$.

Now that we have defined cost complexity, we introduce the problem of STO as a testbed for tools and ideas that can hopefully be applied to more complex problems. More formally, we give the definition of STO:

▶ **Definition 1** (Search with Two Oracles (STO)). Let N and M be known positive integers and let $S \subseteq [N]$ be an unknown set. There might or might not exist a special item i_*. If i_* exists, then one is promised that $i_* \in S$ and $|S| = M$. If i_* doesn't exist, the size of S is arbitrary. Let f_* and f_S be two functions with domain $[N]$ and range $\{0,1\}$ such that

$$f_*(i) = \begin{cases} 1 & \text{if } i = i_* \\ 0 & \text{if } i \neq i_* \text{ or } i_* \text{ doesn't exist.} \end{cases} \qquad f_S(i) = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S. \end{cases} \tag{9}$$

Then $\mathrm{STO}(f_*, f_S) = 1$ if i_* exists, and 0 otherwise. c_* is the cost associated with f_* and c_S is the cost associated with f_S, with $c_* \geq c_S$.

c_S and c_* are assumed to depend on N and M, but our results hold for any form of that dependence, so we leave off any explicit relationship.

Cost complexity, and STO in particular, are related to several existing oracle problems. In the problem of STO, the function f_S can be thought of as providing extra information or advice about the function f_*. There have been several studies in which access to a single oracle is supplemented with some extra information that can come in the form of another oracle or classical information, e.g. [14, 15]. Previous works [3, 14] have considered multiple oracles, but not with costs. Furthermore, the additional advice oracles considered in these works tend to be somewhat unnatural, and are tailored to the specific problems considered. As mentioned, *ConQCC* is related to the model of variable costs studied by Ambainis, in which he considered a single oracle that has different costs for querying different items [2]. We also note that Cerf et al. [9] consider similar quantum algorithms in the context of constraint satisfaction problems, but they do not approach the problem from an oracular perspective.

3 Quantum Algorithms for STO

We now describe quantum algorithms for solving STO[2]. These algorithms use the oracles $\mathcal{O}_*$ and $\mathcal{O}_S$ directly, rather than the controlled version (i.e. $C\mathcal{O}$) of these oracles. All of our algorithms can be viewed as examples of amplitude amplification. Recall

▶ **Theorem 2** (Amplitude Amplification [8]). *Let $T \subset [N]$, $\alpha \in [0,1]$, and let $\mathcal{O}^T$ be an quantum oracle that marks the elements of T. We define*

$$|T\rangle = \frac{1}{\sqrt{|T|}} \sum_{i \in T} |i\rangle. \tag{10}$$

Given an algorithm $\mathcal{A}$ that acts on a state $|\psi_0\rangle$ and produces a state $|\psi_{\mathcal{A}}\rangle$ such that $|\langle T|\psi_{\mathcal{A}}\rangle| = p$, one can create a new algorithm $\mathcal{B}$ that applies $\mathcal{O}^T$, $\mathcal{A}$, and $\mathcal{A}^{-1}$ each

$$\tau = \left\lceil \frac{\arcsin\sqrt{1-\alpha} - \arcsin p}{2\arcsin p} \right\rceil \tag{11}$$

times, and which acts on the initial state $|\psi_0\rangle$ and produces a state $|\psi_{\mathcal{B}}\rangle$ such that

$$|\psi_{\mathcal{B}}\rangle = \sqrt{1-\alpha}|T\rangle + \sqrt{\alpha}|T^{\perp}\rangle, \tag{12}$$

where $\langle T|T^{\perp}\rangle = 0$ and $|T^{\perp}\rangle \in Span(|T\rangle, |\psi_A\rangle)$.

This gives us the following Corollary:

▶ **Corollary 3.** *Let $\mathcal{A}$ and τ be as in Theorem 2, and assume $\mathcal{O}^T$ has cost c_T while $\mathcal{A}$ and $\mathcal{A}^{-1}$ have cost $c_{\mathcal{A}}$. Then there exists a algorithm $\mathcal{B}$ that applies $\mathcal{O}^T$, $\mathcal{A}$ and $\mathcal{A}^{-1}$ not in superposition, and produces the state $|T\rangle$ with probability $1-\epsilon$ such that*

$$\mathrm{Cost}(\mathcal{B}) = \tau\left(c_T + 2c_{\mathcal{A}}\right). \tag{13}$$

In the following, we describe three algorithms for STO. We consider the limit that $M, N/M \to \infty$ to simplify our analysis, but this limit still captures the essential behavior of the algorithms. We use the following notation:

$$\begin{aligned} |N\rangle &= \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle, \\ |S\rangle &= \frac{1}{\sqrt{M}} \sum_{i \in S} |i\rangle. \end{aligned} \tag{14}$$

We have a slight abuse of notation, since $|N\rangle$ could refer either to the equal superposition state, or the N^{th} standard basis state. However, whenever we write $|N\rangle$, we will always mean the equal superposition state.

The first algorithm we consider ignores $\mathcal{O}_S$ and performs a Grover search for i_* using $\mathcal{O}_*$:

[2] For the purpose of describing these algorithms, we assume that i_* exists. A single application of O_* at the end of the algorithm can be used to check (with appropriate probability) whether or not i_* exists, at a cost of c_*.

▶ **Algorithm 1** (Grover's Search). *Prepare the state $|N\rangle$ at cost 0. Set $\mathcal{A}$ equal to the identity. Then by Corollary 3 there exists an algorithm $\mathcal{B}$ that produces the state $|i_*\rangle$ with probability $1-\epsilon$ with cost*

$$c_* \left\lceil \frac{\arcsin\sqrt{1-\epsilon} - \arcsin\frac{1}{\sqrt{N}}}{2\arcsin\frac{1}{\sqrt{N}}} \right\rceil. \tag{15}$$

In the limit of $N \to \infty$, the cost becomes

$$c_* \arcsin\sqrt{1-\epsilon}\sqrt{N}. \tag{16}$$

However, if $\mathcal{O}_S$ comes to us cheaply, we would like to take advantage of it: The following algorithm first rotates $|N\rangle$ to $|S\rangle$ (using $\mathcal{O}_S$), and then rotates $|S\rangle$ to $|i_*\rangle$ (using both $\mathcal{O}_S$ and $\mathcal{O}_*$).

▶ **Algorithm 2.** *Prepare the state $|N\rangle$ at cost 0. Set $\mathcal{A}$ equal to the identity. Since $|\langle N|S\rangle| = \sqrt{M/N}$, by Corollary 3 there exists an algorithm $\mathcal{B}$ that with probability 1 produces the state $|S\rangle$ at cost*

$$c_S \left\lceil \frac{\left(\frac{\pi}{2} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil. \tag{17}$$

Now $|\langle i_*|S\rangle| = \sqrt{1/M}$, so using Corollary 3 again, there exists an algorithm $\mathcal{C}$ that with probability $1-\epsilon$ produces the state $|i_*\rangle$ at cost

$$\left\lceil \frac{\arcsin\sqrt{1-\epsilon} - \arcsin\frac{1}{\sqrt{M}}}{2\arcsin\frac{1}{\sqrt{M}}} \right\rceil \left(c_* + 2c_S \left\lceil \frac{\left(\frac{\pi}{2} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil \right). \tag{18}$$

Dropping terms of size at most $O(M^{-1/2})$ or $O\left((M/N)^{1/2}\right)$ of the zeroth order terms, the cost becomes

$$\frac{\arcsin\sqrt{1-\epsilon}}{4}\left(2c_*\sqrt{M} + \pi c_S\sqrt{N}\right). \tag{19}$$

Combining Algorithms 1 and 2, we have that

$$\begin{aligned} QCC(STO) &= O\left(\min\left\{c_*\sqrt{N}, c_*\sqrt{M} + c_S\sqrt{N}\right\}\right) \\ &= O\left(\max\left\{c_*\sqrt{M}, c_S\sqrt{N}\right\}\right). \end{aligned} \tag{20}$$

In Section 4, we will show that this cost (Eq. (20)) is asymptotically optimal. This means that Algorithm 2 is always asymptotically optimal, although Algorithm 1 has lower cost when $c_* \approx c_S$. However, it turns out that there is an algorithm that has lower cost than either Algorithm 1 or 2. In Section 4, we give evidence that this final algorithm, which we call the Hybrid Algorithm, is not just asymptotically optimal, but exactly optimal.

The two algorithms we have so far presented can be summarized as follows: Algorithm 1 directly performs Grover rotations to rotate $|N\rangle$ to $|i_*\rangle$, while Algorithm 2 first rotates $|N\rangle$ to $|S\rangle$, then rotates $|S\rangle$ to $|i_*\rangle$. The final algorithm we consider, the Hybrid Algorithm, first rotates $|N\rangle$ to some superposition of $|N\rangle$ and $|S\rangle$, and then rotates to $|i_*\rangle$.

▶ **Algorithm 3** (Hybrid Algorithm). *Prepare the state $|N\rangle$ at cost 0. Set $\mathcal{A}$ equal to the identity. Since $|\langle N|S\rangle| = \sqrt{M/N}$, by Theorem 2 and Corollary 3 there exists an algorithm $\mathcal{B}$ that produces a state $|\psi_{\mathcal{B}}\rangle$ at cost*

$$c_S \left\lceil \frac{\left(\arcsin\sqrt{1-\alpha} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil . \tag{21}$$

where

$$|\psi_{\mathcal{B}}\rangle = \sqrt{1-\alpha}|S\rangle + \sqrt{\alpha}|S^{\perp}\rangle. \tag{22}$$

By Theorem 2, $|S^{\perp}\rangle$ is a linear combination of $|S\rangle$ and $|N\rangle$ but is orthogonal to $|S\rangle$. Therefore, $|S^{\perp}\rangle$ is a superposition of all elements not in S, and so $\langle i_*|S^{\perp}\rangle = 0$. Thus

$$\frac{\sqrt{1-\alpha}}{\sqrt{M}} = \langle\psi_{\mathcal{B}}|i_*\rangle. \tag{23}$$

Applying Corollary 3 again, we can create an algorithm $\mathcal{C}$ that has cost

$$\left\lceil \frac{\arcsin\sqrt{1-\epsilon} - \arcsin\frac{\sqrt{1-\alpha}}{\sqrt{M}}}{2\arcsin\frac{\sqrt{1-\alpha}}{\sqrt{M}}} \right\rceil \left(c_* + 2c_S \left\lceil \frac{\left(\arcsin\sqrt{1-\alpha} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil \right) \tag{24}$$

and produces the state $|i_*\rangle$ with probability $1-\epsilon$. In Appendix A, we show there is a choice of α such that, dropping terms of size at most $O(M^{-1/2})$ or $O((M/N)^{1/4})$ that of the zeroth order terms, the cost is

$$Cost(Hybrid) = \frac{c_S\sqrt{N}\arcsin\sqrt{1-\epsilon}}{2}\sec\left(\phi_{opt} + \sqrt{\frac{M}{N}}\right), \tag{25}$$

where ϕ_{opt} is given by

$$\phi_{opt} = \max\begin{cases} 0 \\ \phi : \tan\left(\phi + \sqrt{\frac{M}{N}}\right) = \phi + \frac{c_*}{c_S}\sqrt{\frac{M}{N}}. \end{cases} \tag{26}$$

When c_S is close to c_*, this algorithm approximates Algorithm 1. When c_S is very small compared to c_*, it approximates Algorithm 2. Otherwise, it, in effect, interpolates between the two algorithms.

4 Lower Bound on Quantum Cost Complexity of STO

Several techniques give asymptotically tight lower bounds on the quantum cost complexity of STO. We will briefly sketch two approaches for bounding the quantum cost complexity (QCC), and then discuss a bound on controlled quantum cost complexity ($ConQCC$) in detail. The fact that so many approaches give good lower bounds is encouraging; this means many techniques from (or variations on) the standard query complexity toolbox can be applied.

Our lower bound on $ConQCC$(STO) is asymptotically tight with the algorithms of Section 3, i.e. Eq. (20), even though those algorithms do not use controlled oracles. Because algorithms that use controlled versions of the oracles are more powerful than oracles that

can not access controlled versions (see Eq. (8)), this result proves that not only are our algorithms for STO asymptotically optimal, but having access to a controlled version of the oracles for STO does not give an advantage.

When discussing lower bounds on the cost of STO, we will often refer to the SEARCH problem. We call SEARCH the problem in which one is given a function $f_* : [N] \to \{0, 1\}$ such that there is exactly zero or one element i_* such that $f_*(i_*) = 1$, and one would like to determine if there is such an element i_*; in other words, SEARCH is computing $\mathrm{OR}(f_*)$ with a promise on f_*.

Here are brief descriptions of two methods for lower bounding QCC. We describe them in the context of STO, but they could be applied more generally.

Oracle Simulation: Suppose one only has an oracle $\mathcal{O}_*$. Then one could use this to simulate an oracle $\mathcal{O}_S$ by applying $\mathcal{O}_*$, and then subsequently randomly choosing $M - 1$ items to mark. If $M \ll N$, with high probability, the chosen $M - 1$ items will not include $\mathcal{O}_*$, and this simulated oracle will act identically to a true $\mathcal{O}_S$. Now any algorithm for STO that uses this simulated oracle will actually only use $\mathcal{O}_*$ to find the marked item i_*, and so the problem reduces to SEARCH. Well-known quantum lower bounds on SEARCH [7] then give a lower bound on the total number of queries to either $\mathcal{O}_*$ or the simulated $\mathcal{O}_S$, which in turn can be used to put a lower bound on the cost. For more details on oracle simulation, see Section 5, in which we use oracle simulation to bound the classical cost complexity of STO.

Adversary Method: One can create an adversary matrix whose rows and columns are indexed by pairs of oracles (f_*, f_S). This matrix can be used to create a progress function, and then one can bound the progress that either oracle $\mathcal{O}_*$ or $\mathcal{O}_S$ can make. This gives lower bounds on the queries needed to $\mathcal{O}_*$ and $\mathcal{O}_S$ to evaluate STO, which in turn can be used to lower bound the cost of STO. In Appendix C, we detail how to create this bound for STO.

4.1 Lower Bound on Controlled Quantum Cost Complexity of STO

In this section, in order to lower bound $ConQCC(\mathrm{STO})$, we consider a new problem in the standard query model, which we call Expanded Search with Two Oracles (ESTO). We show that if we had an algorithm $\mathcal{A}$ which could use the control oracle $C\mathcal{O}$ to solve STO with cost $c_{\mathcal{A}}$, then we could create a new algorithm $\mathcal{A}'$ to solve ESTO using $O(c_{\mathcal{A}})$ queries. We then use the adversary method to lower bound the query complexity of ESTO, which in turn puts a lower bound on $ConQCC(\mathrm{STO})$. This strategy is inspired by Ambianis's approach for lower bounding the variable times search problem [2].

We first describe the problem ESTO. We suggest referencing Figure 1 during the description of the problem for a graphical interpretation. Let N, M, c_* and c_S be as in STO. Without loss of generality, we can assume $c_*, c_S \gg 1$. If they are not, we can multiply both costs by some large factor K. Then the final cost is exactly a factor of K larger than it would have been with the original costs. (If $c_S = 0$, this approach does not work, but in that case, STO reduces to SEARCH). We define

$$m_* = \max\left\{i : \left\lceil \frac{\pi}{4}\sqrt{i} \right\rceil + 1 \leq c_*, i \in \mathbb{Z}\right\},$$
$$m_S = \max\left\{i : \left\lceil \frac{\pi}{4}\sqrt{i} \right\rceil + 1 \leq c_S, i \in \mathbb{Z}\right\},$$

ESTO queries an unknown function $f : [N(m_S + m_*)] \to \{0, 1\}$. We consider $\mathcal{D}_1 = \{1, \ldots, Nm_*\}$ to be the "first part" of the domain of f, and $\mathcal{D}_2 = \{Nm_*+1, \ldots, N(m_*+m_S)\}$

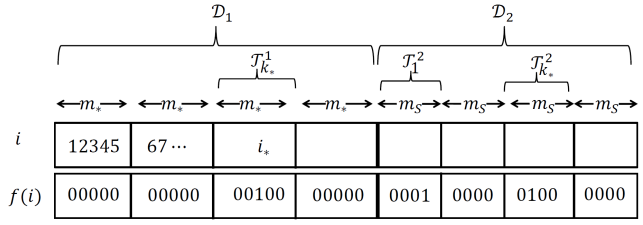

Figure 1 A diagram of a function f for which $\mathrm{ESTO}(f) = 1$. The domain of f is divided into two parts $\mathcal{D}_1$ and $\mathcal{D}_2$. Each of these sets are further divided into N sets of size m_* and m_S respectively. These sets are labeled $\mathcal{T}^1_k$ for sets in $\mathcal{D}_1$, and $\mathcal{T}^2_k$ for sets in $\mathcal{D}_2$. We see there is exactly one value of $i \in \mathcal{D}_1$ with value 1, and it is in the set $\mathcal{T}^1_{k_*}$. In the case shown in this figure, $S = \{1, k_*\}$, so both $\mathcal{T}^2_{k_*}$ and $\mathcal{T}^2_1$ contain exactly one marked item.

to be the "second part" of the domain. We further divide $\mathcal{D}_1$ ($\mathcal{D}_2$) into N blocks of m_* (m_S) elements respectively, where the elements $\mathcal{T}^1_k = \{(k-1)m_* + 1, \ldots, km_*\}$ constitute the k^{th} block of $\mathcal{D}_1$, and the elements $\mathcal{T}^2_k = \{Nm_* + (k-1)m_S + 1, \ldots, Nm_* + km_S\}$ constitute the k^{th} block of $\mathcal{D}_2$.

We are promised that there is either exactly zero or one value $i_* \in \mathcal{D}_1$ such that $f(i_*) = 1$. If there is such an i_*, we label the block it is in by k_*, so $i_* \in T^1_{k_*}$. Furthermore, if i_* exists, there is a set $S \in [N]$ such that $|S| = M$, $k_* \in S$, and for each $k \in S$ there is exactly one value of $i \in \mathcal{T}^2_k$ such that $f(i) = 1$. Given such a function f, $\mathrm{ESTO}(f) = 1$ if there is an item $i_* \in \mathcal{D}_1$ such that $f(i_*) = 1$, and 0 otherwise.

Given an algorithm $\mathcal{A}$ for STO that uses the control oracle $C\mathcal{O}$ and has cost $c_{\mathcal{A}}$, we can create an algorithm $\mathcal{A}'$ to solve ESTO that uses $2c_{\mathcal{A}}$ queries. Let $y^b_j = 1$ for $b \in \{1, 2\}$ if there is an element $i \in \mathcal{T}^b_j$ such that $f(i) = 1$, and 0 otherwise. Then by Claim 2 in [2], there is an algorithm $\mathcal{B}$ that takes $|b, j\rangle|0\rangle|0\rangle \to |b, j\rangle|y^b_j\rangle|\psi^b_j\rangle$ for some state $|\psi^b_j\rangle$ and uses c_* queries if $b = 1$ and c_S queries if $b = 2$. At the cost of doubling the number of queries, we can uncompute the final register. Thus there is an algorithm $\mathcal{B}'$ that takes $|b, j\rangle|0\rangle \to |b, j\rangle|y^b_j\rangle$ and uses $2c_*$ queries if $b = 1$ and $2c_S$ queries if $b = 2$. We also allow for $b = 0$, in which case the algorithm $\mathcal{B}'$ applies the identity.

Then we can solve ESTO using our algorithm $\mathcal{A}$ for STO. In STO we are searching for a specific element $i^* \in [N]$ with certain properties, in ESTO, the search is for a specific block $k^* \in [N]$ with analogous properties. We replace an application of the controlled oracle $C\text{-}\mathcal{O}$ to the state $|b, i\rangle$ with $b \in \{0, 1, 2\}$ and $i \in [N]$ with an application of the algorithm $\mathcal{B}'$ to the state $|b, i\rangle$, (which corresponds to searching the block $\mathcal{T}^b_i$, for $b \in \{1, 2\}$ and $i \in [N]$, or doing nothing if $b = 0$). The number of queries required by $\mathcal{B}'$ will be twice cost of the equivalent query made by $\mathcal{A}$. Due to the specific structure of f, this algorithm will solve ESTO with a number of queries equal to $2c_{\mathcal{A}}$.

Now all that is left is to put a lower bound on the number of queries needed to solve ESTO. We use Ambainis's adversary bound:

▶ **Theorem 4** (Basic Adversary Bound [1]). *Let $F(f(1), \ldots, f(N))$ be a function of N $\{0,1\}$-valued variables $f(i)$, and let X, Y be two sets of inputs such that $F(f) \neq F(g)$ if $f \in X$ and $g \in Y$. Let $R \subset X \times Y$ be such that*

- *For every $f \in X$, there exist at least μ different $g \in Y$ such that $(f, g) \in R$.*
- *For every $g \in Y$, there exist at least μ' different $f \in X$ such that $(f, g) \in R$.*
- *For every $f \in X$ and $i \in [N]$, there are at most l different $g \in Y$ such that $(f, g) \in R$ and $f(i) \neq g(i)$.*

- *For every $g \in Y$ and $i \in [N]$, there exist at least l' different $f \in X$ such that $(f, g) \in R$ and $f(i) \neq g(i)$.*

Then, any quantum algorithm computing F with error at most ϵ on all valid inputs uses at least

$$\frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2}\sqrt{\frac{\mu\mu'}{ll'}} \tag{27}$$

queries.

For the sets X and Y, we only consider functions f where in each block $\mathcal{T}_j^b$, there is at most 1 marked item. We denote by $f_{k_*,i_*,S,S'}$ a function where $i_* \in \mathcal{D}_1$ is the marked item, k_* is the block where the i_* sits (or $i_* = k_* = 0$ if there is no marked item in $\mathcal{D}_1$), S is the set of blocks in $\mathcal{D}_2$ that have exactly one marked item in each block, and S' is a list of the $|S|$ items that are marked in the second part of the domain.

Let X be the set of all functions $f_{k_*,i_*,S,S'}$ with $k_* \neq 0$, $i_* \neq 0$, $|S| = M$, and $k_* \in S$. From our definition of ESTO, these are functions for which the algorithm should output 1. Let Y be the set of functions $f_{0,0,T,T'}$ with $|T| = M - 1$. Then R is defined by $(f_{k_*,i_*,S,S'}, f_{0,0,T,T'}) \in R$ if and only if $T \subset S$, $T' \subset S'$, and $k_* \notin T$. With this definition of R, we have $\mu = 1$ while $\mu' = (N - M + 1)m_* m_S$. Likewise $l = 1$ while $l' = \max\{m_S, m_*\} = m_*$ since $c_* \geq c_S$. Theorem 4 then gives that the number of queries required to solve ESTO, is at least

$$\frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2}\sqrt{(N - M + 1)m_S}. \tag{28}$$

Eq. (28) does not tell the full story; we can repeat this procedure with the set X the same as before, but now the set Y includes all functions $f_{0,0,S,S'}$ such that $|S| = M$. Then we choose $(f_{k_*,i_*,S,S'}, f_{0,0,T,T'}) \in R$ if and only if $T = S$ and $T' = S'$. With this definition of R, we have $\mu = 1$, while $\mu' = Mm_*$. Likewise $l = 1$ while $l' = 1$. Again using Theorem 4, we have that the number of queries required to solve ESTO is at least

$$\frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2}\sqrt{Mm_*}. \tag{29}$$

Since $c_*, c_S \gg 1$, we have $m_* = \Omega((c_*)^2)$ and $m_S = \Omega((c_S)^2)$, so combining Eq. (28) and Eq. (29), and using the fact that a lower bound on the query complexity of ESTO gives a lower bound on the controlled quantum cost complexity of of STO, we have

$$ConQCC_\epsilon(\text{STO}) \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{4} \times \max\left\{\sqrt{Mm_*}, \sqrt{(N - M + 1)m_S}\right\} \tag{30}$$

$$= \Omega\left(\max\left\{\sqrt{M}c_*, \sqrt{(N - M + 1)}c_S\right\}\right). \tag{31}$$

With Eq. (20), this bound proves our algorithms are asymptotically optimal. In Figure 2, we compare the bound given by the reduction to ESTO with the Hybrid Algorithm. Even though the functions are asymptotically tight, the forms of these two bounds are quite different.

4.2 Exact Lower Bound for Cost Complexity of STO

In the introduction, we mentioned several reasons for wanting to prove exact optimality of our algorithm for STO. Aside from finding an example besides Grover's algorithm of an exactly optimal algorithm, proving our algorithm for STO is optimal would have several

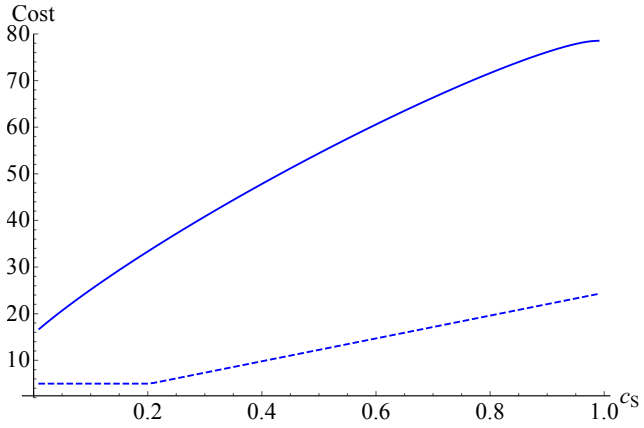

Figure 2 The solid line is the cost of the hybrid algorithm, while the dashed line is the lower bound on the cost given by Eq. (30). The cost is calculated with $c_* = 1$, $N = 10^4$, $M = 400$ and $\epsilon = 0$ while c_S is varied.

other implications. First, the algorithms described in Section 3 are all based on amplitude amplification, so if we can prove these approaches are optimal, that would give evidence that amplitude amplification is an exactly optimal algorithm for certain types of unstructured search problems.

Second, if we consider an extension of STO to many oracles, we can no longer prove asymptotic optimality of our amplitude amplification algorithm. Note that in amplitude amplification, (see Theorem 2), the inner algorithm ($\mathcal{A}$) is applied two times for each application of the oracle that identifies the target state (if $\mathcal{A} = \mathcal{A}^{-1}$). This factor of two is not accounted for in our lower bound of Section 4.1. While this factor of two can be swept under the rug using asymptotic notation, if we consider a problem with k nested oracles, and try to apply a similar strategy as for STO and use nested amplitude amplification, the innermost algorithm will accumulate an extra factor of 2^k in the number of times it must be applied. Using a strategy similar to Section 4.1 to lower bound this problem will not catch that factor of 2^k, for the same reason the factor of 2 is not characterized by the oracle simulation and adversary method. In the case of $k = \log N$ nested oracles, our bounds will no longer be asymptotically tight. Thus, if we can find an exact bound in the case of STO, we might be able to extend it to get asymptotically tight bounds for the case of nested oracles, providing evidence that multiple nestings of amplitude amplification are optimal for certain problems.

We have found that proving an exactly tight lower bound for STO is a challenge, and in fact we can only prove the hybrid algorithm is optimal in a limited setting. The difficulty in proving optimality even in this limited case provides insight into the difficulty of the more general case.

The restricted setting we investigate is to only consider *Grover-like* algorithms.

▶ **Definition 5.** A *Grover-like* algorithm with oracles $\{\mathcal{O}_1, \ldots, \mathcal{O}_l\}$ that act on an N-dimensional Hilbert space must:

- Use only an N-dimensional Hilbert space as its workspace,
- Initialize in the equal superposition state $|N\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle$,
- Use only the unitaries $\{\mathcal{O}_1, \ldots, \mathcal{O}_l\}$ and $G = \mathbb{I} - 2|N\rangle\langle N|$, and
- End with a measurement on the standard basis.

If we consider *Grover-like* algorithms for SEARCH, the state of the system is restricted to a 2-dimensional subspace spanned by $|N\rangle$ and $|i_*\rangle$. Since $G^2 = \mathcal{O}_*^2 = \mathbb{I}$, the only possible algorithm is alternating G and $\mathcal{O}_*$, and one can easily track the progress of the state through the two dimensional space towards $|i_*\rangle$, thus trivially proving that in this setting, Grover's algorithm is exactly optimal.

We will see in the proof of Theorem 6 that for STO, the picture becomes much more complicated. In fact, even in the restricted setting of *Grover-like* algorithms, we need an additional assumption to prove optimality. In particular, we show

▶ **Theorem 6** (Exact Lower Bound). *The cost of every Grover-like algorithm for STO that succeeds with probability at least* $1 - \epsilon$ *for a constant* ϵ *is at least*

$$\frac{c_S\sqrt{N}\arcsin\sqrt{1-\epsilon}}{2}\sec\left(\phi_{opt} + \sqrt{M/N}\right), \tag{32}$$

where ϕ_{opt} *satisfies*

$$\phi_{opt} = \max\begin{cases}0,\\ \phi : \tan\left(\phi + \sqrt{\frac{M}{N}}\right) = \phi + \frac{c_*}{c_S}\sqrt{\frac{M}{N}}.\end{cases} \tag{33}$$

We also require the conditions $M, N/M \to \infty$ *and* $C \to 0$*, where*

$$C \equiv \frac{c_S\sqrt{N}}{c_*\sqrt{\epsilon}2M\cos\left(\phi_{opt} + \sqrt{M/N}\right)}. \tag{34}$$

Theorem 6 matches the cost of our hybrid algorithm, Eq. (25).

The proof of Theorem 6 can be found in Appendix B; here we provide a very brief sketch. Just as a *Grover-like* algorithm for standard search can be thought of as acting on a two dimensional subspace of the full N-dimensional Hilbert space, a Grover-like algorithm for STO can be thought of as acting on a three-dimensional subspace. We create a progress function as a position of the state in this subspace such that G has no affect on the progress function, while $\mathcal{O}_*$ and $\mathcal{O}_S$ can cause the progress function to increase or decrease. We then show that the increase in the progress function due to one of the oracles, divided by the cost of that oracle, is bounded. In other words, for a given cost, we can only increase the progress function by a certain amount, no matter which oracle is used. We finally take the total change in the progress function necessary to achieve success, and divide by the change in progress per cost to put a lower bound the cost.

5 Classical Cost Complexity of STO

In this section, we give bounds on the classical randomized cost complexity (RCC) of STO. We will examine both the exact and bounded error cost complexity. For the exact cost complexity, we see that there are two classical algorithms that resemble Algorithm 1 and Algorithm 2, but whereas in the quantum case, it is possible to do better with the Hybrid Algorithm, we prove that there is no classical counterpart to the Hybrid Algorithm. In the case of exact and bounded error cost complexity, we see a polynomial increase in cost compared to the quantum case.

In the case of exact classical cost complexity, we have:

▶ **Lemma 7.** *The exact (0-error) classical cost complexity of STO is*

$$RCC_0(STO) = \min\{Nc_*, (N-1)c_S + Mc_*\}. \tag{35}$$

Proof. We consider an adversarial oracle that knows in advance the queries the algorithm will make.

Recall that for $i \in [N]$, f_S identifies whether $i \in S$ and f_* identifies whether $i = i_*$. We say an item has been *completely queried* if it has been queried with f_S, and is found to not be an element of S, or if it has been queried with f_*. Then the adversarial oracle acts in the following way:

- The first $M-1$ items that the algorithm queries using oracle f_S are all elements of S.
- If all elements except one have been queried (but not necessarily completely queried) using either function f_* or f_S, the final element to be queried will be an element of S (even if this element is not queried using f_S).
- The last element to be completely queried is the marked item, if it exists.

Any algorithm acting against this adversarial oracle that makes q queries using f_S, has worst-case cost at least

$$\begin{cases} Nc_S + Mc_* & \text{if } q = N, \\ qc_S + [(N-q) + (M-1)]c_* & \text{if } N-1 \geq q \geq M-1, \\ qc_S + Nc_* & \text{if } M-1 \geq q \geq 0. \end{cases} \tag{36}$$

These expressions are minimized at $q = N-1$ or $q = 0$, and we obtain

$$RCC_0(STO) \geq \min\{Nc_*, (N-1)c_S + Mc_*\}. \tag{37}$$

For the upper bound, consider the following two algorithms.

▶ **Algorithm 4.** *Query all items using f_*. This algorithm will find the marked item if it exists with certainty, and has cost Nc_*.*

▶ **Algorithm 5.** *Query all but the last item using f_S. Then:*

- *If M items of S have been found, query f_* on these M items.*
- *If $M-1$ items of S have been found, query f_* on these $M-1$ items, and also the last item (the item that was not queried using f_S).*
- *Otherwise $|S| \neq M$ and therefore no marked item exists.*

This algorithm will find the marked item if it exists with certainty, and has cost $(N-1)c_S + Mc_*$.

Thus we have

$$RCC_0(STO) \leq \min\{Nc_*, (N-1)c_S + Mc_*\}. \tag{38}$$

◀

Algorithm 1 can be thought of as the quantum version of Algorithm 4, while Algorithm 2 can be thought of as the quantum version of Algorithm 5. In the 0-error classical case, these two approaches tell the whole story. However, in the quantum case, you can do better with the Hybrid Algorithm. The Hybrid Algorithm works by doing something very quantum, which is to partially search for the elements of S. In the classical case, this doesn't work. Once you've found an element of S, you've found it; there is no way to partially find an element of S.

With Lemma 7, we've proven that in the 0-error case, we can obtain a polynomial reduction in cost by using a quantum algorithm for STO. Next, we show this polynomial reduction holds even in the case of bounded error algorithms. We do this by reducing STO to the problem of SEARCH. Recall that for SEARCH, we have:

▶ **Lemma 8.** *Any randomized classical algorithm that solves SEARCH with bounded probability must query f_* at least $\Omega(N)$ times.*

Now we can prove the reduction of STO to standard search:

▶ **Lemma 9.** *Any randomized classical algorithm that solves STO with bounded probability of error must use as least $\Omega(N)$ queries to either f_* or f_S, as long as $M/N \leq 1/9$.*

Proof. Suppose there is a randomized algorithm $\mathcal{A}$ that solves STO with probability 3/4 and makes q_* queries to f_* and q_S queries to f_S. Then we will use $\mathcal{A}$ to find i_* in the case when we are given f_* but not f_S. To do this, we will use f_* to create a function that behaves similarly to f_S. We choose a subset $T \in [N]$ with $|T| = M - 1$ at random, and create a function f_T that acts as

$$f_T(i) = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{if } i \notin T. \end{cases} \tag{39}$$

Then we create the function $\tilde{f}_S$ to simulate f_S, where

$$\tilde{f}_S(i) = f_T(i) \vee f_*(i). \tag{40}$$

Each time we want to query $\tilde{f}_S$, we must query $f_*(i)$. Notice that $\tilde{f}_S$ behaves like a valid f_S function unless i_* exists and $i_* \in T$ (because in this case $\tilde{f}_S$ marks $M-1$ items instead of M.) $i_* \in T$ with probability $\frac{M-1}{N.}$

We create $\tilde{f}_S$ as above, and we implement $\mathcal{A}$, but every time $\mathcal{A}$ asks us to apply f_S, we instead apply $\tilde{f}_S$. This new algorithm will succeed with probability $3/4(1-(M-1)/N) \geq 2/3$, because it succeeds with probability 3/4 as long as $i_* \notin \mathcal{F}$. This means we have created an algorithm for standard search which uses $q_* + q_S$ queries to f_* and which succeeds with probability 2/3. But by Lemma 8, we must have $q_* + q_S = \Omega(N)$. ◀

Finally, we note that there is an additional restriction on the number of queries to f_*:

▶ **Lemma 10.** *Any randomized classical algorithm that solves STO with bounded probability must use at least $\Omega(M)$ queries to f_*.*

Proof. Suppose the elements of the subset S were known. Then in the worst case, that would still only narrow down the search to M items. (This is the worst case because if $|S| \neq M$, then one immediately knows there is no marked item.) One must then perform a search for one marked item out of M, which requires $\Omega(M)$ queries via Lemma 8. ◀

Now we can state our lower bound on the query cost of STO:

▶ **Theorem 11.** *The bounded error classical randomized cost complexity of STO is*

$$RCC(STO) = \min\left\{\Omega(c_S N + c_* M), \Omega(c_* N)\right\}. \tag{41}$$

Proof. When $M/N \leq 1/9$, we solve the following linear program:

$$\begin{aligned} \text{minimize: } & q_* c_* + q_{\mathcal{S}} c_{\mathcal{S}} \\ \text{subject to: } & q_* \geq f_1(M, \epsilon) \\ & q_* + q_{\mathcal{S}} \geq f_2(N, M, \epsilon). \end{aligned} \tag{42}$$

When $M/N > 1/9$, from Lemma 10, we have have $q_* = \Omega(M) = \Omega(N)$, so the cost is as least $\Omega(c_* M) = \Omega(c_* N)$. ◀

Comparing Eq. (41) with Eq. (20), we see that there is always a separation between the quantum and classical costs of STO. In particular, to get the quantum scaling from the classical scaling, simply replace all M's and N's by $\sqrt{M}$ and $\sqrt{N}$.

6 Conclusions and Open Questions

While query complexity is a well understood and powerful tool for quantifying the power of quantum computers, there are still problems that are not easily characterized by query complexity. Cost complexity is one way of extending the standard query model, and we've argued that this approach has potential applications in constraint satisfaction problems.

While we motivated STO with problems like k-SAT, graph isomorphism, and the traveling salesman problem, it is not obvious how much of a speed-up an STO inspired algorithm for these problems would be. The speed-up in STO depends critically on N, M, c_*, and c_s. It would be interesting to calculate approximately what this relationship is, for example, in a random k-SAT instance. Once this relationship is better understood, we could determine the amount of speed-up an STO algorithm would give for such a problem. However, even with a better understanding of this relationship, it is unlikely that M would be known exactly. In that case, a method such as fixed point search [16] might be helpful.

STO is a very simple extension of a search problem, and thus the methods described here all have a Grover-ish flavor to them. It would be interesting to find well motivated problems for the cost complexity model where other quantum algorithms could be employed.

We have also left open the question of the exact cost of STO. We believe our algorithm is optimal, but it seems new techniques are needed to prove it.

Acknowledgments. The authors would like to thank Edward Farhi, Andrew Childs, and Aram Harrow for illuminating discussions. Funding for SK provided by the Department of Defense. HSL and CYL are supported by the ARO grant Contract Number W911NF-12-0486. CYL gratefully acknowledges support from the Natural Sciences and Engineering Research Council of Canada.

References

1 Andris Ambainis. Quantum lower bounds by quantum arguments. In *Proc. 32nd ACM STOC*, pages 636–643. ACM, 2000.

2 Andris Ambainis. Quantum search with variable times. *Theory of Computing Systems*, 47(3):786–807, 2010.

3 Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems – the hardness of quantum rewinding. *arXiv preprint arXiv:1404.6898*, 2014.

4 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.

5 Aleksandrs Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *Proc. IEEE 53rd FOCS*, pages 207–216. IEEE Computer Society, 2012.

6 Aleksandrs Belovs and Ansis Rosmanis. On the power of non-adaptive learning graphs. *Computational Complexity*, 23(2):323–354, 2014.

7 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

8 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *arXiv preprint quant-ph/0005055*, 2000.

9 Nicolas J Cerf, Lov K Grover, and Colin P Williams. Nested quantum search and structured problems. *Physical Review A*, 61(3):032303, 2000.

10 Cătălin Dohotaru and Peter Høyer. Exact quantum lower bound for grover's problem. *Quantum Information and Computation*, 9(5-6):533–540, 2009.

11 Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004.

12 Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proc. 39th ACM STOC*, pages 526–535, 2007.

13 Troy Lee, Rajat Mittal, Ben W Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proc. 52nd IEEE FOCS*, pages 344–353. IEEE, 2011.

14 Ashley Montanaro. Quantum search with advice. In *Theory of Quantum Computation, Communication, and Cryptography*, pages 77–93. Springer, 2011.

15 Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *arXiv preprint arXiv:1408.3193*, 2014.

16 Theodore J Yoder, Guang Hao Low, and Isaac L Chuang. Fixed-point quantum search with an optimal number of queries. *Physical review letters*, 113(21):210501, 2014.

A Analysis of the Hybrid Algorithm

Throughout this section, when we are calculating something "to zeroth order", we drop terms whose sizes are $O(M^{-1/2})$ or $O((M/N)^{1/4})$ multiplied by the size of the largest term.

In Section 3, Eq. (24), we showed that the cost of the Hybrid Algorithm is

$$\begin{aligned}\mathrm{Cost(Hybrid)} = &\left\lceil \frac{\arcsin\sqrt{1-\epsilon} - \arcsin\frac{\sqrt{1-\alpha}}{\sqrt{M}}}{2\arcsin\frac{\sqrt{1-\alpha}}{\sqrt{M}}} \right\rceil \\ &\times \left(c_* + 2c_S \left\lceil \frac{\left(\arcsin\sqrt{1-\alpha} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil \right). \end{aligned} \tag{43}$$

In this appendix, we prove that in the limit of $M \to \infty$ and $N/M \to \infty$, there is a choice of α such that the cost is

$$\mathrm{Cost(Hybrid)} = \frac{c_S\sqrt{N}\arcsin\sqrt{1-\epsilon}}{2} \sec\left(\phi_{opt} + \sqrt{\frac{M}{N}}\right), \tag{44}$$

where ϕ_{opt} is given by

$$\phi_{opt} = \max\left\{ \begin{array}{l} 0 \\ \phi : \tan\left(\phi + \sqrt{\frac{M}{N}}\right) = \phi + \frac{c_*}{c_S}\sqrt{\frac{M}{N}}. \end{array} \right. \tag{45}$$

We first define

$$t = \left\lceil \frac{\left(\arcsin\sqrt{1-\alpha} - \arcsin\sqrt{\frac{M}{N}}\right)}{2\arcsin\sqrt{\frac{M}{N}}} \right\rceil, \tag{46}$$

so t is a non-negative integer. Substituting t for α in Eq. (43), we obtain

$$\begin{aligned}\mathrm{Cost(Hybrid)} = &(2tc_S + c_*) \\ &\times \left\lceil \arcsin\sqrt{1-\epsilon} \left\lceil 2\arcsin\left(\frac{\sin\left((2t+1)\arcsin\sqrt{\frac{M}{N}}\right)}{\sqrt{M}} \right) \right\rceil^{-1} - 1/2 \right\rceil. \end{aligned} \tag{47}$$

To zeroth order, this becomes

$$\text{Cost(Hybrid)} = \frac{(2tc_S + c_*)\sqrt{M}\arcsin\sqrt{1-\epsilon}}{2\sin\left((2t+1)\sqrt{\frac{M}{N}}\right)}. \tag{48}$$

Finally, we denote $\phi = 2t\sqrt{M/N}$ to obtain

$$\text{Cost(Hybrid)} = \frac{\left(\phi c_S + \sqrt{\frac{M}{N}}c_*\right)\sqrt{N}\arcsin\sqrt{1-\epsilon}}{2\sin\left(\phi + \sqrt{\frac{M}{N}}\right)}. \tag{49}$$

We take the partial derivative of the cost with respect to ϕ, and set it to zero to find the value of ϕ that gives the smallest cost. We find the cost is minimized when $\phi = \phi_{opt}$, where ϕ_{opt} satisfies

$$\tan\left(\phi_{opt} + \sqrt{M/N}\right) = \phi_{opt} + \frac{c_*}{c}\sqrt{M/N}. \tag{50}$$

Notice that there is always a solution with $\phi_{opt} \in [-\sqrt{M/N}, \pi/2]$. However t is non-negative, so if $\phi_{opt} < 0$ we set $\phi_{opt} = 0$. This condition, along with Eq. (49) and Eq. (50), immediately gives the cost claimed in Eq. (44).

We might not be able to exactly attain this cost, because t must be an integer, so we might only be able to set ϕ close to ϕ_{opt}. We show that even if we can't set ϕ exactly to ϕ_{opt}, we can still attain the cost of Eq. (44), to zeroth order.

There are two cases to consider. In the first case, we assume $(M/N)^{1/4} \leq \phi_{opt} \leq \pi/2$. We require that t be a non-negative integer, so we choose $t = \left\lceil (\phi_{opt}\sqrt{N})/(2\sqrt{M}) \right\rceil$, and hence we set

$$\phi = \left\lceil \frac{\phi_{opt}}{2}\sqrt{\frac{N}{M}} \right\rceil 2\sqrt{\frac{M}{N}}. \tag{51}$$

For that choice, notice that

$$\phi - \phi_{opt} = O\left((M/N)^{1/2}\right). \tag{52}$$

This allows us to relate terms involving ϕ to those involving ϕ_0:

$$\begin{aligned}
\sin\left(\phi + \sqrt{M/N}\right) &= \sin\left(\phi_{opt} + \sqrt{M/N}\right) \pm O((M/N)^{1/2}) \\
&= \sin\left(\phi_{opt} + \sqrt{M/N}\right)\left(1 \pm O((M/N)^{1/4})\right) \\
&= \left(\phi_{opt} + \frac{c_*}{c_S}\sqrt{M/N}\right)\cos\left(\phi_{opt} + \sqrt{M/N}\right)\left(1 \pm O\left((M/N)^{1/4}\right)\right) \\
&= \left(\phi + \frac{c_*}{c_S}\sqrt{M/N}\right)\cos\left(\phi_{opt} + \sqrt{M/N}\right)\left(1 \pm O\left((M/N)^{1/4}\right)\right),
\end{aligned} \tag{53}$$

where in the first line, we use the angle addition formula and Eq. (52); in the second, we use the assumption that $\phi_{opt} \geq (M/N)^{1/4}$; in the third line we applied Eq. (50); and in the last we have used Eq. (52) and the assumption on the size of ϕ_{opt}. Plugging Eq. (53) into our expression for the cost in Eq. (49), we have that to zeroth order, we obtain Eq. (44), as desired.

We now consider the second case, when $0 \leq \phi_{opt} < (M/N)^{1/4}$. In this case, we simply set $t = 0$, and hence $\phi = 0$. Plugging $\phi = 0$ the cost of Eq. (49), we have, to zeroth order,

$$\text{Cost(Hybrid)} = \frac{\arcsin\sqrt{1-\epsilon}\sqrt{N}}{2} c_*. \tag{54}$$

We will show that Eq. (54) and Eq. (44) are equivalent for $0 \leq \phi_{opt} < (M/N)^{1/4}$. We have

$$\sec\left(\phi_{opt} + \sqrt{M/N}\right) = 1 + O\left((M/N)^{1/4}\right). \tag{55}$$

We can expand Eq. (50) to get

$$c_S = c_*\left(1 - O\left((M/N)^{1/4}\right)\right). \tag{56}$$

Plugging Eqs. (55) and (56) into Eq. (44) and keeping only zeroth order terms, we recover Eq. (54).

B Proof of Theorem 6

In this section, we prove the following theorem:

▶ **Theorem 6** (Exact Lower Bound). *The cost of every Grover-like algorithm for STO that succeeds with probability at least $1-\epsilon$ for a constant ϵ is at least*

$$\frac{c_S\sqrt{N}\arcsin\sqrt{1-\epsilon}}{2}\sec\left(\phi_{opt} + \sqrt{M/N}\right), \tag{32}$$

where ϕ_{opt} satisfies

$$\phi_{opt} = \max\begin{cases} 0, \\ \phi : \tan\left(\phi + \sqrt{\frac{M}{N}}\right) = \phi + \frac{c_*}{c_S}\sqrt{\frac{M}{N}}. \end{cases} \tag{33}$$

We also require the conditions $M, N/M \to \infty$ and $C \to 0$, where

$$C \equiv \frac{c_S\sqrt{N}}{c_*\sqrt{\epsilon}2M\cos\left(\phi_{opt} + \sqrt{M/N}\right)}. \tag{34}$$

Proof. Throughout this section, when we say to zeroth order, we mean dropping terms of size at most $O(M^{-1/2})$ or $O\left((M/N)^{1/2}\right)$ or $O(C)$ of the zeroth order terms.

Since we only consider the operations $\mathcal{O}_S$, $\mathcal{O}_*$, and G, the state of the system never leaves the three-dimensional space spanned by the orthonormal states

$$\left\{ |i_*\rangle, \quad |S^-\rangle = \tfrac{1}{\sqrt{M-1}}\textstyle\sum_{i\in S-\{i_*\}}|i\rangle, \quad |S^\perp\rangle = \tfrac{1}{\sqrt{N-M}}\textstyle\sum_{i\notin S}|i\rangle \right\}. \tag{57}$$

It turns out that it is more convenient to work in a slightly shifted basis from that of Eq. (57). We instead use the orthonormal basis states:

$$\begin{aligned} |x\rangle &= \cos\theta_0|i_*\rangle - \sin\theta_0|S^-\rangle, \\ |y\rangle &= \cos\phi_0\sin\theta_0|i_*\rangle + \cos\phi_0\cos\theta_0|S^-\rangle - \sin\phi_0|S^\perp\rangle, \\ |z\rangle &= \sin\phi_0\sin\theta_0|i_*\rangle + \cos\theta_0\sin\phi_0|S^-\rangle + \cos\phi_0|S^\perp\rangle \\ &= |N\rangle. \end{aligned} \tag{58}$$

We can think of these states as forming the axes of a 3-dimensional space, where a state

$$|\chi\rangle = x|x\rangle + y|y\rangle + z|z\rangle \tag{59}$$

is identified with the point (x, y, z). Then if the algorithm is initialized in the equal superposition state $|z\rangle$, the goal of the algorithm is to move from the $|z\rangle$-axis towards the $|x\rangle$-axis.

Since any normalized state of the system corresponds to a point on the unit sphere in this space, let us now introduce polar coordinates, with the $|x\rangle$-axis as the polar axis. Specifically, we associate the state $|\chi\rangle$ with the polar coordinates (θ, ϕ), where

$$x = \sin\theta, \quad y = \cos\theta\sin\phi, \quad z = \cos\theta\cos\phi \tag{60}$$

for $\theta \in [-\pi/2, \pi/2]$. (The variable ϕ in this section plays a nearly identical role to ϕ in Appendix A, so we use the same variable name.)

If we multiply a state by -1, this transforms the coordinates from (θ, ϕ) to $(-\theta, \phi + \pi)$. Because overall phases do not affect the state, we can apply this transformation for free. In particular, we use it to "pick a gauge" and choose the coordinates that satisfy $\theta \geq 0$.

For a Grover-like algorithm which finds the marked state with high success probability, the algorithm starts at the point $(\theta = 0, \phi = 0)$, and must end near $\theta = \pi/2$. We define a progress function $H(\theta, \phi)$, for $\theta > 0$, as

$$H(\theta, \phi) = \theta - k \min_{\ell \in \mathbb{Z}} |\phi + 2\ell\pi - \pi/2|, \tag{61}$$

where

$$k = \theta_0 \cos(\phi_{opt} + \phi_0), \tag{62}$$

$$\phi_0 = \arcsin\sqrt{M/N}, \tag{63}$$

$$\theta_0 = \arcsin\sqrt{1/M}, \tag{64}$$

$$\phi_{opt} = \max \begin{cases} 0 \\ \phi : \tan(\phi + \phi_0) = \phi + \frac{c_*}{c}\phi_0. \end{cases} \tag{65}$$

The second term of $H(\theta, \phi)$ is proportional to the angular distance of ϕ to $\pi/2$ (taken so the distance is $< \pi$).

Before we analyze how each unitary changes the progress function, we will look at the total progress that must occur for the algorithm to succeed. The total progress gained by the algorithm must be larger than the difference between the value of the progress function at the starting point and the end point. We pick the starting point as the *last* time the algorithm increases θ from less than $2\theta_0$ to more than $2\theta_0$, and $\phi \geq 0$. (We require $\phi \geq 0$ for Lemma 12, and we require $\theta \geq 2\theta_0$ in order to calculate the progress due to $\mathcal{O}_*$.) We will show later that such a point will always exist for any successful algorithm, and also that at such a point $\theta < 6\theta_0$. Thus the value of the progress function at the starting point is at most $6\theta_0$.

For the end point of the algorithm, note that the probability of success is

$$\sin^2(\theta) > 1 - \epsilon, \tag{66}$$

to zeroth order. Thus the total change in progress function is at least

$$\arcsin\sqrt{1-\epsilon} - k\pi - 6\theta_0 > \arcsin\sqrt{1-\epsilon} - (6+\pi)\theta_0, \tag{67}$$

where we bound k using Eq. (62), and the $k\pi$ term comes from the worst possible value of ϕ when θ gets sufficiently large.

We note the following: from Eq. (25) and Eq. (62) we see that the cost of the optimal algorithm is at most

$$\frac{c_S \arcsin\sqrt{1-\epsilon}}{\phi_0 k}, \tag{68}$$

and from Eq. (67) the change in the progress function is at least $\arcsin\sqrt{1-\epsilon} - (6+\pi)\theta_0$; therefore the progress per unit cost must be at least $\phi_0 k/c_S$, to zeroth order. It therefore follows that when calculating the change in progress function, we only need to keep track of terms up to order $O(\phi_0 k/c_S)$ per cost. For example, for $\mathcal{O}_*$, we need only keep track of the change in *progress* (not progress per cost) up to order $O(\phi_0 k c_*/c_S)$.

The change in the progress function $H(\theta,\phi)$ due to the unitaries G, $\mathcal{O}_S$, and $\mathcal{O}_*$ can be calculated by how they change the coordinates (θ,ϕ) of a state. After some algebra and using our gauge choice, we obtain

- G: The unitary G is a reflection about the z-axis, and in polar coordinates is the map

$$G:\ (\theta,\phi) \to (\theta, \pi-\phi). \tag{69}$$

  Comparing with Eq. (61), we see G has no effect on the progress function.

- $\mathcal{O}_S$: The oracle $\mathcal{O}_S$ is a reflection about the state which has polar coordinates $(\theta=0, \phi=-\phi_0)$.

$$\mathcal{O}_S:\ (\theta,\phi) \to (\theta, \pi-\phi-2\phi_0) \tag{70}$$

  We see that $\mathcal{O}_S$ can change the progress function by at most $2\phi_0 k$. Thus the increase in the progress function per cost due to $\mathcal{O}_S$ is at most

$$\frac{2\phi_0 k}{c_S} = \frac{2\phi_0\theta_0\cos(\phi_{opt}+\phi_0)}{c_S}. \tag{71}$$

- $\mathcal{O}_*$: The oracle $\mathcal{O}_*$ is a reflection about the state $|i_*\rangle$, which is close to $|x\rangle$. We find $\mathcal{O}_*$ transforms coordinates as

$$\theta \to \theta + 2\theta_0\sin(\phi+\phi_0) + O(\theta_0^2) \tag{72}$$

$$\phi \to \pi + \phi + O\left(\frac{\theta_0}{\cos\theta}\right). \tag{73}$$

Now we consider how $\mathcal{O}_*$ affects the progress function; unlike the previous cases, which we calculated exactly, we will only analyze this case to zeroth order. We will first show that we can assume $|\phi| \le \pi/2$. Suppose that $|\phi| > \pi/2$ just before we would like to apply $\mathcal{O}_*$. Then instead of applying $\mathcal{O}_*$, we apply $G\mathcal{O}_*G$. One can check that with this replacement, when $\mathcal{O}_*$ is applied, $|\phi| \le \pi/2$. Furthermore one can verify that this replacement causes θ to increase (which can only be good for the progress function), while on the other hand, the value of ϕ changes by at most $O\left(\theta_0/\cos\theta\right)$ due to this replacement, resulting in a change in the progress function of size $O\left(k\theta_0/\sqrt{\epsilon}\right)$ (using Eq. (66) to bound $\cos\theta$). Using our assumption that that $C = o(1)$, this change has order less than $O(\phi_0 k c_*/c_S)$, and so can be discarded using the argument following Eq. (68). We can therefore assume that $\mathcal{O}_*$ is always applied at $|\phi| \le \pi/2$.

Now we can examine the change in the progress function due to the action of $\mathcal{O}_*$. The increase in the progress function is

$$\begin{aligned} &2\theta_0 \sin(\phi + \phi_0) + O\left(\theta_0^2\right) \\ &- k\left(\min_{\ell\in\mathbb{Z}} |-\phi + 2\ell\pi - \pi/2| - \min_{\ell\in\mathbb{Z}} |\phi + 2\ell\pi - \pi/2|\right) + O\left(\frac{k\theta_0}{\cos\theta}\right). \end{aligned} \tag{74}$$

Since $|\phi| \leq \pi/2$, the increase in the progress function due to $\mathcal{O}_*$ is less than

$$2\theta_0 \sin(\phi + \phi_0) - 2\phi\theta_0 \cos(\phi_{opt} + \phi_0) + O\left(\frac{\theta_0^2}{\sqrt{\epsilon}}\right), \tag{75}$$

where we have used the value of k from Eq. (62) and bounded $\cos\theta$ with Eq. (66).

Taking the first and second derivatives of Eq. (75) with respect to ϕ, we see that when $\phi \geq 0$, the increase in the progress function is maximized when $\phi = \phi_{opt}$. It turns out that if one applies $\mathcal{O}_*$ at $\phi < 0$, it is sometimes possible to achieve a larger increase in progress per cost than when $\phi \geq 0$. However, we show at the end of this section, (Lemma 12), that applying $\mathcal{O}_*$ when $\phi < 0$ will always be less efficient (up to higher order terms) in terms of the increase in progress function per cost, than applying $\mathcal{O}_*$ at $\phi = \phi_{opt}$, when viewed in the context of the larger algorithm. Applying the definition of ϕ_{opt} from Eq. (65) to Eq. (75), and using the definition of C from Eq. (34), the increase in the progress function due to $\mathcal{O}_*$ is less than

$$\frac{c_* 2\phi_0\theta_0 \cos(\phi_{opt} + \phi_0)}{c_S}\left(1 + O(\phi_0^2) + O\left(C\right)\right), \tag{76}$$

where the $O(\phi_0^2)$ term accounts for the case that $\phi_{opt} = 0$.

From Eq. (71) and Eq. (76) we see that (to zeroth order) the maximum increase in the progress function per cost is the same whether $\mathcal{O}_*$ is applied or $\mathcal{O}_S$ is applied. Dividing the total necessary change in progress (Eq. (67)) by the maximum change in progress per cost (Eq. (76)) gives us the minimum cost:

$$\arcsin\sqrt{1-\epsilon}\frac{c_S}{2\phi_0\theta_0\cos(\phi_{opt} + \phi_0)}\left(1 - O(C) - O(M^{-1/2}) - O\left((M/N)^{-1/2}\right)\right). \tag{77}$$

In the limit of $N, M \to \infty$ and $C \to 0$, (to zeroth order) we have that the cost is at least

$$\arcsin\sqrt{1-\epsilon}\frac{c_S\sqrt{M}}{2\phi_0\cos(\phi_{opt} + \phi_0)}, \tag{78}$$

which matches the cost of Eq. (25).

We now justify why the value of the progress function must be less than $6\theta_0$ when we start tracking it. Immediately before we start tracking the progress function, we have $\theta < 2\theta_0$, so the bound on the increase in progress given by Eq. (75) does not necessarily apply. However, it is simple to show that the increase in the progress function due to $\mathcal{O}_*$ is always bounded by $2\theta_0$, where we have dropped terms of $O(\theta_0^2/\sqrt{\epsilon})$ as before. Thus if $\theta < 2\theta_0$, and then $\mathcal{O}_*$ is applied, θ can increase by at most $2\theta_0$, and so the new value of θ satisfies $\theta < 4\theta_0$. At this point, $\theta > 2\theta_0$, but ϕ might be negative. Notice that θ can not increase unless $\mathcal{O}_*$ is applied, (and θ must increase in order to obtain a high probability of success) but $\mathcal{O}_*$ flips the sign of ϕ, so after applying $\mathcal{O}_*$ at most one more time, we will have both the conditions $\theta > 2\theta_0$ and $\phi \geq 0$ satisfied, at which point we start tracking the progress function. This tells us that the value of θ will be at most $6\theta_0$ when we start tracking the progress function. ◀

▶ **Lemma 12.** *Suppose there is an algorithm than applies $\mathcal{O}_*$ when $\phi < 0$. Then there is always an alternative algorithm that achieves the same or greater increase in progress for the same or less cost (up to zeroth order), but applies $\mathcal{O}_*$ only when $\phi \geq 0$.*

Proof. We begin by classifying the the possible sequences of $\mathcal{O}_*$, $\mathcal{O}_S$, and G the algorithm can take. We will use notation such that unitaries act from right to left, so $G\mathcal{O}_*$ signifies $\mathcal{O}_*$ acts first, and then G acts.

First look at $\mathcal{O}_*$. We can always assume $\mathcal{O}_*$ is followed by a G; if it is not, insert a GG pair after the $\mathcal{O}_*$. Note in the discussion following Eq. (73), we proved that we can assume $|\phi| < \pi/2$ before applying $\mathcal{O}_*$. With Eqs. (69) and (73) we have

$$G\mathcal{O}_* : \phi \to -\phi + O\left(\frac{\theta_0}{\cos\theta}\right). \tag{79}$$

Since $|\phi| < \frac{\pi}{2}$ before $G\mathcal{O}_*$ acts, we also have $|\phi| < \frac{\pi}{2}$ after $G\mathcal{O}_*$ acts, up to an additive factor of $O\left(\frac{\theta_0}{\cos\theta}\right)$, which we can ignore thanks to the discussion following Eq. (68). Therefore $G\mathcal{O}_*$ maps ϕ inside the $|\phi| < \frac{\pi}{2}$ region.

In between applications of $G\mathcal{O}_*$, there is always a sequence of one of the following forms:

$$(G\mathcal{O}_S)^m, \quad G(G\mathcal{O}_S)^m, \quad (\mathcal{O}_S G)^m, \quad \text{or} \quad G(\mathcal{O}_S G)^m, \tag{80}$$

where m is a non-negative integer that indicates multiple applications of the unitary sequence inside the parenthesis. These are the only possible sequences because $\mathcal{O}_S\mathcal{O}_S = I$ and $GG = I$. Combining the action of G and $\mathcal{O}_S$ in Eqs. (69) and (70) we get

$$(\mathcal{O}_S G)^m : (\theta, \phi) \to (\theta, \phi - 2m\phi_0) \tag{81}$$

$$(G\mathcal{O}_S)^m : (\theta, \phi) \to (\theta, \phi + 2m\phi_0). \tag{82}$$

Thus the 4 sequences of Eq. (80) rotate ϕ by some amount $\pm 2m\phi_0$, possibly followed by the transformation $\phi \to \pi - \phi$.

Now we focus on the algorithm's action on ϕ. Since the $G\mathcal{O}_*$'s are mapping ϕ between points inside the $|\phi| < \frac{\pi}{2}$ region, the four possible sequences of alternating G and $\mathcal{O}_S$ in Eq (80) just connect the value of ϕ after applying $G\mathcal{O}_*$ to the value of ϕ before the next application of $G\mathcal{O}_*$. Generalizing Figure 3, one can see that the shortest path uses either $(G\mathcal{O}_S)^m$ or $(\mathcal{O}_S G)^m$ to connect points inside the $|\phi| < \frac{\pi}{2}$ region. Therefore we do not need to consider the sequences $G(\mathcal{O}_S G)^m$ or $G(G\mathcal{O}_S)^m$.

Next, we show that if one initially has $\phi > 0$, it is never advantageous to again apply $G\mathcal{O}_*$ when $\phi < 0$. Since the algorithm must consist of applications of $G\mathcal{O}_*$ separated by sequences of either $(\mathcal{O}_S G)^m$ or $(G\mathcal{O}_S)^m$, we can enumerate and address the three possible cases that lead us to apply $\mathcal{O}_*$ at some $\phi = \phi_{neg} < 0$ after initially having $\phi \geq 0$. The three possible cases are laid out graphically in Figure 4. In order to prove that none of the cases are optimal, we define the function

$$p_*(\phi) = 2(\theta_0 \sin(\phi + \phi_0) - k\phi) \tag{83}$$

as the change in progress function due to an application of $\mathcal{O}_*$, dropping higher order terms. Note for $\phi \geq 0$, ϕ_{opt} optimizes Eq. (83) as discussed after Eq. (75). We proceed to treat the three cases.

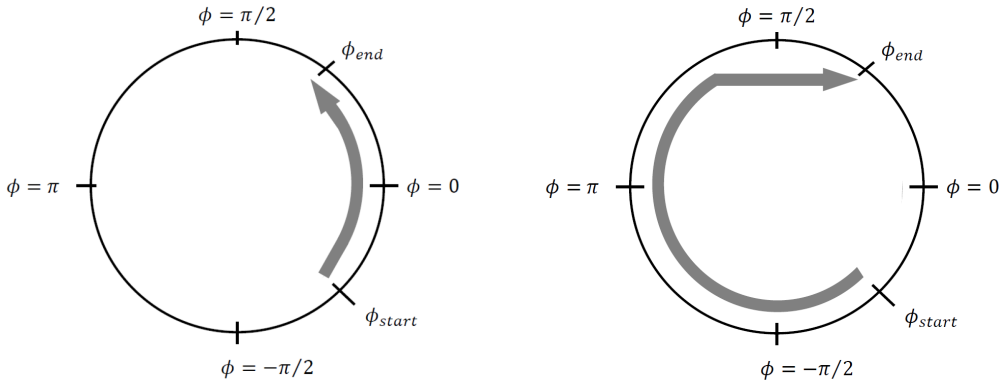

Figure 3 The path in the figure at left uses a sequence $(G\mathcal{O}_S)^m$ to move from ϕ_{start} to ϕ_{end}, whereas the path in figure at right uses a sequence $G(\mathcal{O}_S G)^m$. The path using $(G\mathcal{O}_S)^m$ is shorter, signifying that fewer uses of $\mathcal{O}_S$ are required to move from ϕ_{start} to ϕ_{end}, and thus this is the more efficient path.

Sequence I. We consider the following sequence of operations (see Figure 4):

(i) Start with $\phi_i > 0$. Then apply $G\mathcal{O}_*$ to get to $-\phi_i$.

(ii) Apply $(G\mathcal{O}_S)$ some number of times to increase ϕ to $\phi_{neg} > -\phi_i$.

(iii) Apply $G\mathcal{O}_*$ to get to $-\phi_{neg} < \phi_i$.

The change in progress due only to $\mathcal{O}_*$ in this sequence is

$$\begin{aligned} p_*(\phi_i) + p_*(\phi_{neg}) &= 2(\theta_0 \sin(\phi_i + \phi_0) - k\phi_i) \\ &\quad + 2(\theta_0 \sin(\phi_{neg} + \phi_0) - k\phi_{neg}) \\ &\leq 4[\theta_0 \sin(\frac{\phi_i + \phi_{neg}}{2} + \phi_0) - k\frac{\phi_{neg} + \phi_i}{2}] \\ &= 2p_*(\phi_i + \phi_{neg}) \\ &\leq 2p_*(\phi_{opt}), \end{aligned} \tag{84}$$

Since $\phi_{neg} + \phi_i \geq 0$, the average progress due to the two applications of $\mathcal{O}_*$ is worse than if we had applied $\mathcal{O}_*$ at ϕ_{opt} both times. Thus this sequence cannot be optimal.

Sequence II. We consider the following sequence of operations (see Figure 4):

(i) Start with $\phi_i > 0$. Then apply $G\mathcal{O}_*$ to get to $-\phi_i$.

(ii) Apply $(\mathcal{O}_S G)$ some number of times to decrease ϕ to $\phi_{neg} < -\phi_i$.

(iii) Apply $G\mathcal{O}_*$ to get to $-\phi_{neg} > \phi_i$.

Compare Sequence **II** to the following Sequence **2**:

(a) Start with $\phi_i > 0$. Then apply $(G\mathcal{O}_S)$ some number of times to increase ϕ to $-\phi_{neg} > \phi_i$.

The difference in progress between Sequence **II** and Sequence **2** is

$$\begin{aligned} &(2\theta_0 \sin(\phi_i + \phi_0) + 2\theta_0 \sin(\phi_{neg} + \phi_0)) \\ =&4\theta_0 \sin(\frac{\phi_i + \phi_{neg}}{2} + \phi_0) \cos(\frac{\phi_i - \phi_{neg}}{2}) \\ <&4\theta_0 \sin\phi_0, \end{aligned} \tag{85}$$

since $-\frac{\pi}{4} < \frac{\phi_i + \phi_{neg}}{2} < 0$ and $0 < \frac{\phi_i - \phi_{neg}}{2} < \frac{\pi}{2}$. Sequence **II** and Sequence **2** both use the same number of applications of $\mathcal{O}_S$ (in steps (ii) and (a) respectively). Therefore, the

Sequence **II** has an additional cost $2c_*$ while it only has an added increase in progress of

$$\begin{aligned} 4\theta_0 \sin\phi_0 =& 2p_*(0) \\ \leq& 2p_*(\phi_{opt}). \end{aligned} \tag{86}$$

Therefore Sequence **II** does not attain the increase in progress per cost that one could attain by only applying $\mathcal{O}_*$ at ϕ_{opt}.

Sequence III. We consider the following sequence of operations (see Figure 4):

(i) Start with $\phi_i \geq 0$, then apply $(O_S G)$ some number of times to decrease ϕ to $\phi_{neg} < 0$.

(ii) Apply $G\mathcal{O}_*$ to get to $-\phi_{neg}$.

Compare Sequence **III** to the following Sequence **3**:

(a) Start with $\phi_i \geq 0$, and then apply $(O_S G)$ some number of times to decrease ϕ to ϕ_w such that $2\phi_0 > \phi_w \geq 0$.

(b) Apply $G\mathcal{O}_*$ to get to $-\phi_w$.

(c) Apply (GO_S) some number of times to increase ϕ to $-\phi_{neg} > 0$.

Note that we can always create a sequence with such a ϕ_w because $(O_S G)$ changes ϕ by at most $2\phi_0$ each time. The cost of Sequence **III** is the same as the cost of Sequence **3**. The difference in progress between Sequence **III** and Sequence **3** is

$$\begin{aligned} & 2\theta_0 \sin(\phi_{neg} + \phi_0) - 2\theta_0 \sin(\phi_w + \phi_0) \\ \leq& 4\theta_0 \cos\left(\frac{\phi_{neg} + \phi_w}{2} + \phi_0\right) \sin\left(\frac{\phi_{neg} - \phi_w}{2}\right) \\ <& 0 \end{aligned} \tag{87}$$

since $|\frac{\phi_{neg}+\phi_w}{2} + \phi_0| < \frac{\pi}{2}$ and $\frac{\pi}{2} < \frac{\phi_{neg}-\phi_w}{2} < 0$. Therefore Sequence III is not optimal either.

Hence we conclude that applying $\mathcal{O}_*$ at negative ϕ never achieves as much increase in progress per cost as applying $\mathcal{O}_*$ at ϕ_{opt}, and therefore we only need to consider applying $\mathcal{O}_*$ at positive ϕ, at ϕ_{opt}. ◀

C An Adversary Lower Bound

In this section, we will show how to apply the adversary method to the problem of cost complexity of STO.

Suppose we are given access to an oracle $\mathcal{O}_*$, which implements the function f_*, and an oracle $\mathcal{O}_S$, which implements the function f_S. Then any algorithm which solves STO using these oracles, after t steps, produces a state

$$|\psi^t_{f_*,f_S}\rangle = U^t \mathcal{O}_{c_t} \cdots U^2 \mathcal{O}_{c_2} U^1 \mathcal{O}_{c_1} |\psi^0\rangle, \tag{88}$$

where $c_j \in \{*, S\}$, and U^j are fixed unitaries independent of f_* and f_S.

We create an adversary matrix Γ, a matrix whose rows and columns are indexed by pairs of functions $(f_*, f_S) \in D_{\text{STO}}$, where D_{STO} is the set of valid inputs to STO. Furthermore, we have the condition that that $\Gamma[(f_*, f_S), (g_*, g_S)] = 0$ if $\text{STO}(f_*, f_S) = \text{STO}(g_*, g_S)$. With this notation, we define the progress function:

$$W^t = \sum_{(f_*,f_S),(g_*,g_S) \in D_{\text{STO}} \times D_{\text{STO}}} \Gamma_{(f_*,f_S),(g_*,g_S)} v_{f_*,f_S} v^*_{g_*,g_S} \langle \psi^t_{f_*,f_S} | \psi^t_{g_*,g_S} \rangle \tag{89}$$

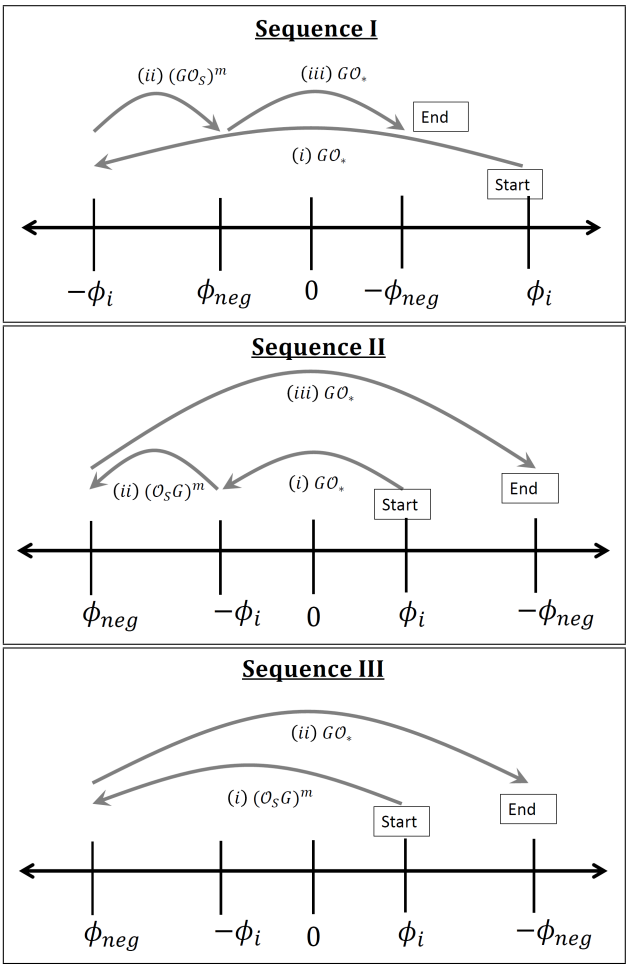

Figure 4 Possible paths that could lead to applying $G\mathcal{O}_*$ at a negative value of ϕ, when initially, ϕ has positive value.

for a vector v indexed by the elements of D_{STO}, such that $\|v\| = 1$ and v is an eigenvector of Γ with eigenvalue $\pm\|\Gamma\|$, (where $\|\cdot\|$ signifies the l-2 norm for vectors or the induced l-2 norm for matrices).

Then following [12][3], we have

1. $W^0 = \|\Gamma\|$.
2. $W^T \leq \left(2\sqrt{\epsilon(1-\epsilon)} + 2\epsilon\right)\|\Gamma\|$, for any algorithm with probability of error at most ϵ.
3. $W^{t-1} - W^t \leq 2\max_i \|\Gamma \circ D_i^{c_t}\|$ where $D_i^{c_t}$ are $|D_{\mathrm{STO}}| \times |D_{\mathrm{STO}}|$ matrices satisfying

$$D_i^*[(f_*, f_S), (g_*, g_S)] = \begin{cases} 0 \text{ if } f_*(i) = g_*(i), \\ 1 \text{ otherwise,} \end{cases} \qquad D_i^S[(f_*, f_S), (g_*, g_S)] = \begin{cases} 0 \text{ if } f_S(i) = f_S(i), \\ 1 \text{ otherwise.} \end{cases}$$

Thus if q_* queries are made to $\mathcal{O}_*$ and q_S queries are made to $\mathcal{O}_S$, we have

$$\|\Gamma\| g(\epsilon) \leq q_* \max_i \|\Gamma \circ D_i^*\| + q_S \max_i \|\Gamma \circ D_i^S\| \tag{90}$$

[3] The proofs are identical, so we omit them.

where

$$g(\epsilon) = \frac{1 - \left(2\sqrt{\epsilon(1-\epsilon)} + 2\epsilon\right)}{2}. \tag{91}$$

We construct the following adversary matrix for STO: $\Gamma[(f_*, f_S), (g_*, g_S)] = 1$ if one of the following conditions holds:

- $\mathrm{STO}(f_*, f_S) = 1$, $\mathrm{STO}(g_*, g_S) = 0$, and $f_S(i) = g_S(i)$ except if $f_*(i^*) = 1$, then $g_S(i^*) = 0$,
- $\mathrm{STO}(g_*, g_S) = 1$, $\mathrm{STO}(f_*, f_S) = 0$, and $g_S(i) = f_S(i)$ except if $g_*(i^*) = 1$, then $f_S(i^*) = 0$.

Otherwise, $\Gamma = 0$.

One can calculate (or it is easy to see by analogy to a standard Grover search over $N - M + 1$ items) that

$$\begin{aligned} \|\Gamma\| &= \sqrt{N - M + 1}, \\ \max_i \|\Gamma \circ D_i^{c_t}\| &= 1, \\ \max_i \|\Gamma \circ D_i^{S}\| &= 1. \end{aligned} \tag{92}$$

Plugging into Eq. (90) we have

$$g(\epsilon)\sqrt{N - M + 1} \leq q_* + q_S, \tag{93}$$

so for $N > M/2$, we have

$$QCC(\mathrm{STO}) = \Omega(c_S\sqrt{N}). \tag{94}$$

We also consider a second adversary matrix for STO. Let $\Gamma[(f_*, f_S), (g_*, g_S)] = 1$ if one of the following conditions holds:

- $\mathrm{STO}(f_*, f_S) = 1$, $\mathrm{STO}(g_*, g_S) = 0$, and $f_S(i) = g_S(i)$,
- $\mathrm{STO}(g_*, g_S) = 1$, $\mathrm{STO}(f_*, f_S) = 0$, and $g_S(i) = f_S(i)$.

Otherwise, $\Gamma = 0$.

In this case, the adversary matrix only pairs instances such that $\mathcal{O}_S$ is the same in both pairs. Thus it is as if the set S is known ahead of time. In this case, one can calculate (or it is easy to see by analogy to a standard Grover search over M items), that

$$\begin{aligned} \|\Gamma\| &= \sqrt{M} \\ \max_i \|\Gamma \circ D_i^{j_t}\| &= 1 \\ \max_i \|\Gamma \circ D_i^{S}\| &= 0. \end{aligned} \tag{95}$$

Plugging into Eq. (90), we have

$$g(\epsilon)\sqrt{M} \leq q_*, \tag{96}$$

so

$$QCC(\mathrm{STO}) = \Omega(c_*\sqrt{M}) \tag{97}$$

Combining Eq. (94) and Eq. (97), we obtain a bound that matches Eq. (20):

$$QCC(\mathrm{STO}) = \Omega\left(\max\{c_*\sqrt{M}, c_S\sqrt{N}\}\right). \tag{98}$$