# Topics in quantum algorithms - Adiabatic Algorithm, Quantum Money, and Bomb Query Complexity

by

Han-Hsuan Lin

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Ph.D. in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Signature redacted

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Physics
May 22, 2015

Signature redacted

Certified by . . . . . . . . . . . . . . . . .
Edward Farhi
Professor of Physics
Thesis Supervisor

Signature redacted

Accepted by . . . . . . . . . . . . . . .
Nergis Mavalvala
Associate Department Head for Education

# Topics in quantum algorithms - Adiabatic Algorithm, Quantum Money, and Bomb Query Complexity

by

Han-Hsuan Lin

Submitted to the Department of Physics
on May 22, 2015, in partial fulfillment of the
requirements for the degree of
Ph.D. in Physics

## Abstract

In this thesis, I present three results on quantum algorithms and their complexity. The first one is a numerical study on the quantum adiabatic algorithm(QAA) . We tested the performance of the QAA on random instances of MAX 2-SAT on 20 qubits and showed 3 strategies that improved QAA's performance, including a counter intuitive strategy of decreasing the overall evolution time. The second result is a security proof for the quantum money by knots proposed by Farhi et. al. We proved that quantum money by knots can not be cloned in a black box way unless graph isomorphism is efficiently solvable by a quantum computer. Lastly we defined a modified quantum query model, which we called bomb query complexity $B(f)$, inspired by the Elitzur-Vaidman bomb-testing problem. We completely characterized bomb query complexity be showing that $B(f) = \Theta(Q(f)^2)$. This result implies a new method to find upper bounds on quantum query complexity, which we applied on the maximum bipartite matching problem to get an algorithm with $O(n^{1.75})$ quantum query complexity, improving from the best known trivial $O(n^2)$ upper bound.

Thesis Supervisor: Edward Farhi
Title: Professor of Physics

# Acknowledgments

I want to thank my advisors Edward Farhi and Peter Shor. Thanks to my collaborators Edward Farhi, Peter Shor, Cedric Lin, and Elizabeth Crosson . Thanks to Aram Harrow and Scott Aaronson for useful discussions and mentoring. Thanks to Aram Harrow for LaTeX help. Thanks to Shelby Kimmel, Robin Kothari, Seth Lloyd, Issac Chuang, and everyone around the MIT quantum information community. Thanks to my parents.

# Contents

# Chapter 1

# Introduction

Quantum computing combines computer science and quantum mechanics. Physicists today believe the universe to be fundamentally quantum mechanical. On the other hand, computers, one of the most influential modern technology, are classical. Quantum computers bridge the difference: they are computers that store and manipulate data in the form of quantum states.

Are quantum computers better than classical computers? Many argue that the exponentially large Hilbert space and entanglement allows quantum computers to be more powerful than classical computers, and there is strong evidence for a positive answer, for example, Shor's factoring algorithm [2] and Grover search [3]. However, the power of quantum computers are still unknown in many areas. For example, one of the most sought-after question of quantum computing is whether quantum computers can efficiently solve NP-complete problems. In this thesis, we try to further our understanding in three different areas of quantum algorithms: quantum adiabatic algorithms, component quantum money, and bomb query complexity.

The quantum adiabatic algorithm was proposed by Farhi et. al. [4] to solve constraint satisfaction problems (CSP). It works by slowly changing an easily-solvable Hamiltonian to the complex problem Hamiltonian. By the adiabatic theorem, if the change in the Hamiltonian is slow enough (adiabatically), a system initially in the ground state will evolve as the instantaneous ground state of the changing Hamiltonian and end in the ground state of the final problem Hamiltonian, which encodes the answer of the CSP. We can solve any CSP with the adiabatic algorithm. However, the runtime required by the adiabatic algorithm is not known for most CSPs. Thus we cannot in general compare the efficiency of the quantum adiabatic algorithm with classical algorithms.

The first part of this thesis is a numerical study of quantum adiabatic algorithm and provides three strategies to improve its success probability. We numerically study the performance of the quantum adiabatic algorithm on randomly-generated 20-bit instances of MAX 2-SAT with a unique maximally satisfying assignment. The probability of getting this unique assignment at the end of the quantum evolution is the success probability. We pick out instances with low success probabilities and test three strategies on those hard instances. The strategies are: decreasing the overall evolution time, initializing the system in excited states, and adding a random local Hamiltonian to the middle of the evolution. All three strategies consistently improve the success probability on the hard instances.

The no-cloning theorem of quantum mechanics gives hints that quantum money cannot be counterfeited. Classical information is easy to copy: given a string on a computer, you can just copy and paste it. Therefore, if a bank hands out a secret string as a bill, a counterfeiter can easily clone it. This is why online banking requires a trusted third-party. On the other hand, the no-cloning theorem says that an arbitrary quantum state cannot be cloned. This opens up the possibility of quantum money that is physically impossible to counterfeit.

Formally, a quantum money scheme is defined by two procedures: 1. How the mint produces quantum money. 2. How everyone verifies quantum money. Both procedures need to be efficient for a practical quantum money scheme. Moreover, a secure quantum money scheme cannot be counterfeited, meaning that no one outside of the mint can forge additional valid quantum money state given one copy of a quantum money state. A quantum money scheme is private-key if only the mint can verify it; a quantum money scheme is public-key if everyone can verify a quantum money state without sending it back to the mint.

There are several proposed quantum money schemes. Wiesner's private-key quantum money [5] is the earliest proposal and boasts a complete security proof for one-time use. However, it is private-key and insecure after repeated uses [6] [7]. As a public key quantum money scheme, Aaronson and and Christiano constructed quantum money from hidden subspaces. In their scheme, every quantum money state is associated with a secret (the hidden subspace), and everyone who knows the secret can efficiently produce more copies of the corresponding quantum money state. The security of the subspace quantum money scheme is tied to the hardness of finding the zero-space of random multivariate polynomials. Last but not least, the focus of this thesis is quantum money from knots, proposed by Farhi et. al [1].

The second part of this thesis provides evidence on the security of quantum money from knots. Unlike quantum money from hidden subspaces, quantum money from knots does not require the mint to hold a secret. In the knot quantum money scheme, the mint starts with an equal superposition of oriented knots and measures its Alexander polynomial, a knot invariant. The post-measurement state, an equal superposition of knots with the same Alexander polynomial, is issued as a quantum money state, and the measured Alexander polynomial is published as a valid serial number. Because the state created is determined by quantum randomness, even the mint itself cannot easily reproduce a published quantum money state. The verification procedure checks whether a state has the correct Alexander polynomial and is invariant under permutations of knots. However, no security proof is known for knot quantum money except a vague insight that its security is closely related to the hardness of knot isomorphism. Lutomirski generalized quantum money from knot to component quantum money, replacing knots with an arbitrary set, measurement of the Alexander polynomial with a "labelling function", and permutation of knots with "mixers" [8]. In this thesis, we build from Lutomirski's generalization of knot quantum money and prove that knot quantum money is secure with black box calls to the labelling function and mixers, unless quantum computers can efficiently solve graph isomorphism, under reasonable assumptions.

In the last part of this thesis, we introduced a new query model, which we call bomb query complexity $B(f)$. Inspired by the Elitzur-Vaidman bomb testing problem, a bomb query is a controlled quantum query with the two extra restrictions: 1. The result of each query is immediately measured. 2. The algorithm fails whenever a 1 is measured. We characterize bomb query complexity by showing that $B(f) = \Theta(Q(f)^2)$. This result gives a new way to generate quantum upper bounds as follows: we give a method to modify a classical randomized algorithm into a bomb query algorithm, which then gives nonconstructive upper bounds on $Q(f) = \Theta(\sqrt{(B(f)})$. We subsequently find a method to construct explicit quantum algorithms matching the nonconstructive upper bounds. Applying this method to the maximum bipartite matching problem gives an algorithm using $O(n^{1.75})$ quantum queries, improving from the best known trivial $O(n^2)$ upper bound.

In conclusion, in this thesis we discuss three different topics about quantum algorithms. We numerically test the quantum adiabatic algorithm, which is a quantum algorithm for solving CSP. We give a security proof on quantum money from knots with certain assumptions, which shed some light on capabilities of quantum algorithms cloning quantum states and quantum cryptography. Finally we introduce bomb query complexity, which provides another angle to investigate quantum query complexity.

## 1.1 Quantum adiabatic algorithm

The quantum adiabatic algorithm (QAA) can be used on a quantum computer as an optimization method [4] for finding the global minimum of a classical cost function $f : \{0,1\}^n \to \mathbb{R}$. The cost function is encoded in a problem Hamiltonian $H_P$ which acts on the Hilbert space of $n$ spin-$\frac{1}{2}$ particles,

$$H_P = \sum_{z\in\{0,1\}^n} f(z)|z\rangle\langle z|. \tag{1.1.1}$$

The Hamiltonian $H_P$ is diagonal in the computational basis, and its ground state corresponds to the bit string that minimizes $f$. To reach the ground state of $H_P$ the system is first initialized to be in the ground state of a beginning Hamiltonian, which is traditionally taken to be

$$H_B = \sum_{i=1}^{n} \left( \frac{1 - \sigma_x^i}{2} \right) . \tag{1.1.2}$$

The ground state of $H_B$, which can be prepared efficiently, is the uniform superposition of computational basis states

$$|\psi_{init}\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle . \tag{1.1.3}$$

The system is then acted upon by the time-dependent Hamiltonian

$$H(t) = (1 - \frac{t}{T}) H_B + \frac{t}{T} H_P \tag{1.1.4}$$

from time $t = 0$ to $t = T$ according to the Schrodinger equation

$$i \frac{\mathrm{d}}{\mathrm{dt}} |\psi(t)\rangle \; = H(t)|\psi(t)\rangle . \tag{1.1.5}$$

For a problem instance with a unique string $w$ that minimizes $f$, the probability of obtaining $w$ at time $t = T$,

$$P(T) = |\langle w|\psi(t = T)\rangle|^2 , \tag{1.1.6}$$

is a metric for the success of the method on that particular instance.

By the adiabatic theorem, if we prepare the system initially in the ground state of $H_B$ and evolve for a sufficiently long time $T$, then the state of the system at the end of the evolution will have a large overlap with the ground state of $H_P$. Specifically, the adiabatic approximation requires $T > \mathcal{O}(g_{min}^{-2})$, where $g_{min}$ is the minimum difference between the ground state energy and the first excited state energy during the course of the evolution.

In this thesis we explore strategies that do not necessarily require a run time $T > \mathcal{O}(g_{min}^{-2})$. We sidestep the usual question of determining how the run time $T$ needed to achieve a certain success probability scales with the input size $n$. Instead we work at a fixed bit number, $n = 20$, and look at strategies for improving the success probability for hard instances at this number of bits. We observe three strategies that increase the success probability for all of the hard instances we generated: evolving the system more rapidly ("the hare beats the tortoise"), initializing the system to be in a superposition of the states in the first excited subspace of $H_B$ ("going lower by aiming higher"), and adding random local Hamiltonian terms to the middle of the evolution path ("the meandering path may be faster").

## 1.2 Quantum money

One of the well known properties of classical information is its inherent *copyability*; a piece of information can be duplicated an unlimited number of times. This often taken-for-granted fact has ensured that money counterfeiters and governments are engaged in a never-ending technological arms race. Moreover, digital commerce must thus involve a trusted third-party (such as a credit card company) for verification. In contrast, quantum information is inherently uncopyable, by virtue of the *No-Cloning Theorem.* We could therefore hope for a cryptographic protocol, which we shall call a *quantum money scheme*, that (informally) satisfies the following requirements:

1. The mint can produce a quantum state $|\$\rangle$ (which we call a *money state*).

2. Anyone with a quantum computer can verify whether the purported money state is valid or not.

3. No one, except for the mint, can forge a money state; in other words, given one money state it should be infeasible to produce another one.

How do we satisfy these requirements? For requirement 2, we could imagine issuing each money state $|\$_i\rangle$ with a *serial number* $i$; the serial number would tell you how to verify the money state. As for requirements 1 and 3, a natural approach is for the mint to hold a secret with which it generates money states and serial numbers. Hopefully, then, it would be difficult for anyone to forge a money state without the secret. This is the route taken by Aaronson and Christiano in constructing their quantum money from hidden subspaces [9].

In this thesis, we look at an alternative approach in which the mint does not keep any secret at all, but relies on the randomness of quantum measurements to generate serial numbers. This approach is realized in quantum money from knots by Farhi et. al. [1], first proposed in [10] and idealized to component money in [8]. In this thesis, we work on the idealized version of [8] where some operations are treated as oracles. Roughly speaking, a component money scheme is composed of a partition $\{S_i\}$ on a large set $S$, and "component mixer" and "labelling function" operations. The "component mixer" maps any element of a component $S_i$ into another element of the same component uniformly, and the labelling function $L$ identifies which component an element is in (if $s \in S_i$, $L(s) = i$). The valid money states are equal superpositions over the same component, $|S_i|^{-1/2} \sum\limits_{s \in S_i} |s\rangle$ for some serial number $i$. We can generate a money state by creating a uniform superposition over $S$, applying the labelling function $L$, and then measuring the component (see Section 2). Valid money states can be efficiently verified by applying the component mixer and labelling function. In the example of quantum money from knots, $S$ is the set of knots with certain size represented by link diagrams, each $\{S_i\}$ is the set of knots with certain Alexander polynomial, $i$ is the corresponding Alexander polynomial, and the mixer is approximated by a Markov chain of grid moves.

In this thesis, we aim to provide some evidence for the security of this scheme, by showing that if an attacker can forge a money state under this scheme with a black-box attack (that is, the attacker never uses the "inner workings" of the component mixer and the labelling function), then the attacker can solve the GRAPH ISOMORPHISM problem (given two graphs, tell whether the two graphs are isomorphic).

## 1.3 Bomb query complexity

Quantum query complexity is an important method of understanding the power of quantum computers. In this model we are given a black-box containing a boolean string $x = x_1 \cdots x_N$, and we would like to calculate some function $f(x)$ with as few quantum accesses to the black-box as possible. It is often easier to give bounds on the query complexity than to the time complexity of a problem, and insights from the former often prove useful in understanding the power and limitations of quantum computers. One famous example is Grover's algorithm for unstructured search [3]; by casting this problem into the query model it was shown that $\Theta(\sqrt{N})$ queries is required [11], proving that Grover's algorithm is optimal.

Several methods have been proposed to bound the quantum query complexity. Upper bounds are almost always proven by finding better query algorithms. Some general methods of constructing quantum algorithms have been proposed, such as quantum walks [12, 13, 14, 15] and learning graphs [16]. For lower bounds, the main methods are the polynomial method [17] and adversary method [18]. In particular, the general adversary lower bound [19] has been shown to tightly characterize quantum query complexity [20, 21, 22], but calculating such a tight bound seems difficult in general. Nevertheless, the general adversary lower bound is valuable as a theoretical tool, for example in proving composition theorems [21, 22, 23] or showing nonconstructive (!) upper bounds [23].

### 1.3.1 Our work

To improve our understanding of quantum query complexity, we introduce and study an alternative oracle model, which we call the *bomb oracle* (see Section 4.2 for the precise definition). Our model is inspired by the concept of *interaction free measurements*, illustrated vividly by the Elitzur-Vaidman bomb testing problem [24], in which a property of a system can be measured without disturbing the system significantly. Like the quantum oracle model, in the bomb oracle model we want to evaluate a function $f(x)$ on a hidden boolean string $x = x_1 \cdots x_N$ while querying the oracle as few times as possible. In this model, however, the bomb oracle is a controlled quantum oracle with the extra requirement that the algorithm fails if the controlled query returns a 1. This seemingly impossible task can be tackled using the quantum Zeno effect [25], in a fashion similar to the Elitzur-Vaidman bomb tester [26] (Section

4.1.1).

Our main result (Theorem 20) is that the bomb query complexity, $B(f)$, is characterized by the square of the quantum query complexity $Q(f)$:

**Theorem 20.**

$$B(f) = \Theta(Q(f)^2). \tag{1.3.1}$$

We prove the upper bound, $B(f) = O(Q(f)^2)$ (Theorem 22), by adapting Kwiat et al.'s solution of the Elitzur-Vaidman bomb testing problem (Section 4.1.1, [26]) to our model. We prove the lower bound, $B(f) = \Omega(Q(f)^2)$ (Theorem 23), by demonstrating that $B(f)$ is lower bounded by the square of the general adversary bound [19], $(\mathrm{Adv}^{\pm}(f))^2$. The aforementioned result that the general adversary bound tightly characterizes the quantum query complexity [20, 21, 22], $Q(f) = \Theta(\mathrm{Adv}^{\pm}(f))$, allows us to draw our conclusion.

This characterization of Theorem 20 allows us to give *nonconstructive* upper bounds to the quantum query complexity for some problems. For some functions $f$ a bomb query algorithm is easily designed by adapting a classical algorithm: specifically, we show that (stated informally):

**Theorem 27** (informal). *Suppose there is a classical algorithm that computes $f(x)$ in $T$ queries, and the algorithm guesses the result of each query (0 or 1), making no more than an expected $G$ mistakes for all $x$. Then we can design a bomb query algorithm that uses $O(TG)$ queries, and hence $B(f) = O(TG)$. By our characterization of Theorem 20, $Q(f) = O(\sqrt{TG})$.*

This result inspired us to look for an explicit quantum algorithm that reproduces the query complexity $O(\sqrt{TG})$. We were able to do so:

**Theorem 28.** *Under the assumptions of Theorem 27, there is an explicit algorithm (Algorithm 30) for $f$ with query complexity $O(\sqrt{TG})$.*

Using Algorithm 30, we were able to give an $O(n^{3/2})$ algorithm for the single-source shortest paths (SSSP) problem in an unweighted graph with $n$ vertices, beating the best-known $O(n^{3/2}\sqrt{\log n})$ algorithm [27]. A more striking application is our $O(n^{7/4})$

algorithm for maximum bipartite matching; in this case the best-known upper bound was the trivial $O(n^2)$, although the time complexity of this problem had been studied in [28] (and similar problems in [29]).

Finally, in Section 4.6 we briefly discuss a related query complexity model, which we call the *projective query complexity* $P(f)$, in which each quantum query to $x$ is immediately followed by a classical measurement of the query result. This model seems interesting to us because its power lies between classical and quantum: we observe that $P(f) \leq B(f) = \Theta(Q(f)^2)$ and $Q(f) \leq P(f) \leq R(f)$, where $R(f)$ is the classical randomized query complexity. We note that Regev and Schiff [30] showed that $P(OR) = \Theta(N)$.

### 1.3.2 Past and related work

Mitchison and Jozsa have proposed a different computational model called *counterfactual computation* [31], also based on interaction-free measurement. In counterfactual computation the result of a computation may be learnt without ever running the computer. There has been some discussion on what constitutes counterfactual computation; see for example [32, 33, 34, 35, 36, 37, 38].

There have also been other applications of interaction-free measurement to quantum cryptography. For example, Noh has proposed counterfactual quantum cryptography [39], where a secret key is distributed between parties, even though a particle carrying secret information is not actually transmitted. More recently, Brodutch et al. proposed an adaptive attack [6] on Wiesner's quantum money scheme [5]; this attack is directly based off Kwiat et al.'s solution of the Elitzur-Vaidman bomb testing problem [26].

Our Algorithm 30 is very similar to Kothari's algorithm for the oracle identification problem [40], and we refer to his analysis of the query complexity in our work.

The projective query model we detail in Section 4.6 was, to our knowledge, first considered by Aaronson in unpublished work in 2002 [41].

# Chapter 2

# Different Strategies for Optimization Using the Quantum Adiabatic Algorithm

In this chapter we describe 3 strategies to improve the quantum adiabatic algorithm. This chapter is mostly excerpted from [42] which is joint work with Elizabeth Crosson, Edward Farhi, Cedric Yen-Yu Lin, and Peter Shor.

## 2.1 Instance Selection

We sought to accumulate an ensemble of instances of MAX 2-SAT on $n = 20$ bits that are hard for the QAA as described above. Our instances are constructed by randomly generating 60 distinct clauses, each involving two distinct bits, and retaining the instance only if there is a unique assignment $w$ that minimizes the number of violated clauses. We keep only those instances that have a unique minimal assignment because degenerate ground states of $H_P$ make the energy gap zero, and because we wish to avoid the complication of having success probabilities depend on the number of optimal solutions.

We generated $202,078$ instances and selected all those having a low success probability at $T = 100$, using $P(100) < 10^{-4}$ as our cutoff, resulting in a collection of

137 hard instances. To speed up the search for these instances, we used a mean-field algorithm to approximate the QAA in equations 1.1.1 through 1.1.4 with $T = 100$, and then we discarded the instances that had a final mean-field energy of 0.5 or less above the energy of the optimal assignment. We checked that instances that are easy for the mean-field algorithm would also have a high success probability under the full Schrodinger evolution by sampling a separate population of $15,000$ instances, and found that whenever the mean-field algorithm produced a final energy less than 0.5 above the ground state energy the instance had success probability $P(100) > 0.2$ according to the Schrodinger evolution. The use of this filter allowed us to discard $3/4$ of the initial $202,078$ instances, and for the remainder we numerically integrated the Schrodinger evolution with $T = 100$.

The success probabilities at $T = 100$ for the test population of $15,000$ instances are given in figure 2.1.1. Most of the instances we generate have high success probability at $T = 100$ (in fact, over half of this population had $P(100) > 0.95$), and hard instances at this time scale and number of bits are rare. This is why we needed to generate roughly $200,000$ total instances, and search through them using over 20,000 hours of CPU time, to obtain our ensemble of 137 instances which have $P(100) < 10^{-4}$.

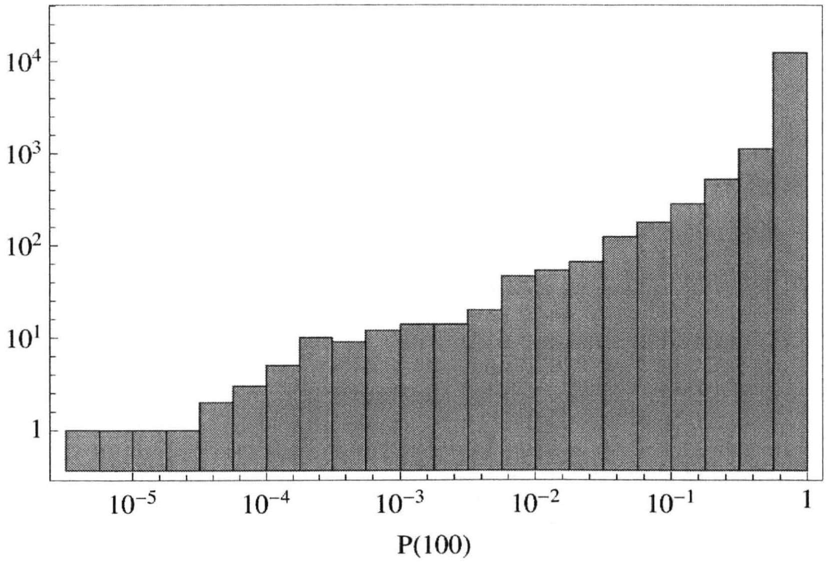


Figure 2.1.1: The distribution of success probabilities for 15000 instances.

## 2.2 The Hare Beats the Tortoise

Considering the success probability $P(T)$ as a function of the total evolution time $T$, we find that all of our instances with low success probability at $T = 100$ exhibit higher success probability at lower values of $T$. Figure 2.2.1 depicts this phenomenon for a single hard instance, which happened to be the first instance we carefully examined. We will refer to this instance as instance #1. We see a distinct peak of success probability at $T_{\max} = 12$ with $P(T_{max}) = 0.05$, which is to be compared with $P(100) = 5 \times 10^{-5}$ and $P(200) = 5 \times 10^{-6}$.

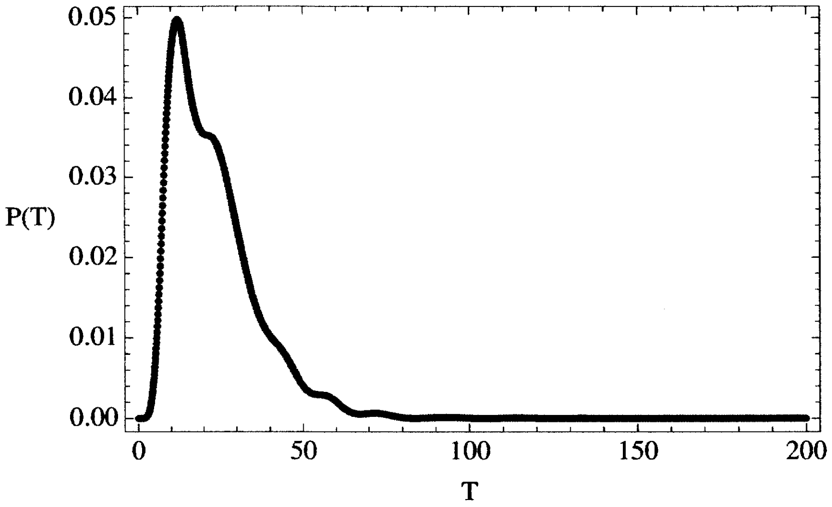


Figure 2.2.1: The success probability as a function of total evolution time $T$ for instance #1.

In figure 2.2.2 we plot the three lowest energy levels of instance #1, and we see a small energy gap which corresponds to an avoided crossing near $s = 0.66$. To see why changing the Hamiltonian more rapidly increases the success probability, figure 2.2.3 gives the instantaneous expectation of the energy, $\langle\psi(t)|H(t)|\psi(t)\rangle$, as a function of $t$ for $T = 10$ and $T = 100$, together with the three lowest energy eigenvalues. We see that when the Hamiltonian is changed slowly, the $T = 100$ case, the system remains close to the ground state for all time $t < 0.66T$, but then switches to closely following the first excited state after the avoided crossing, and arrives with most of its amplitude in the first excited state subspace of $H_P$ with virtually no overlap with $|w\rangle$.

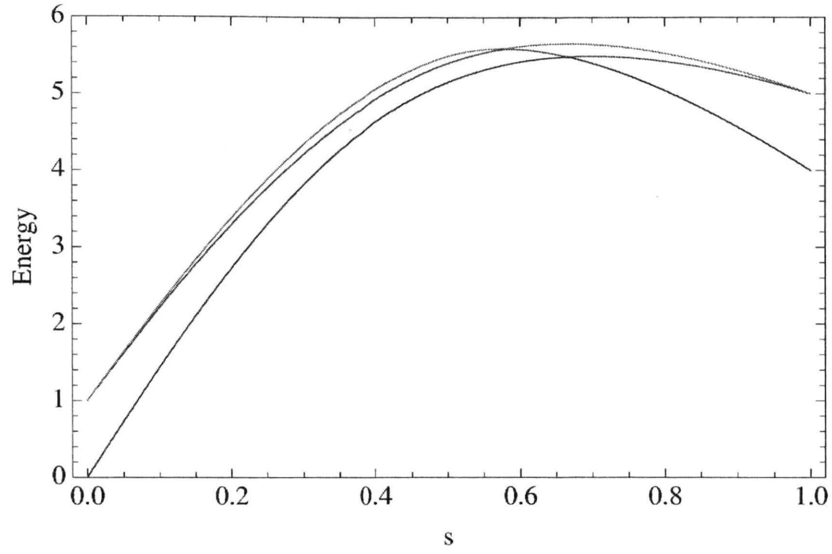


Figure 2.2.2: The lowest three energy levels for instance #1.

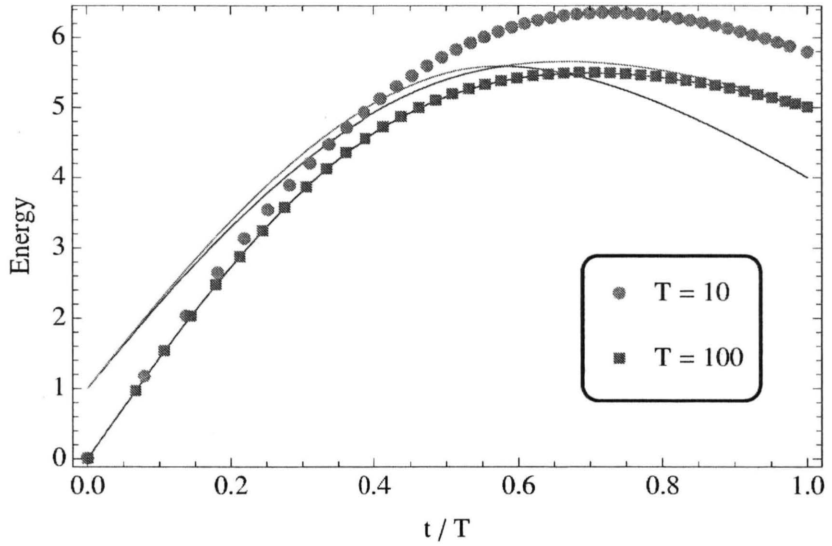


Figure 2.2.3: The lowest three energy levels for instance #1, superimposed with the instantaneous expectation of the energy as a function of $t$ for $T = 10$ and $T = 100$.

In figure 2.2.4 we track the overlap of the rapidly evolved system ($T = 10$) with the lowest two energy eigenstates of $H(t)$, and see that the overlap of the system with the ground state immediately after the crossing corresponds to the overlap with the first excited state immediately before it. When evolving more rapidly, leaking substantial amplitude into the first excited state prior to the crossing is responsible for the increased probability of finding the system in the ground state at the end of the evolution.

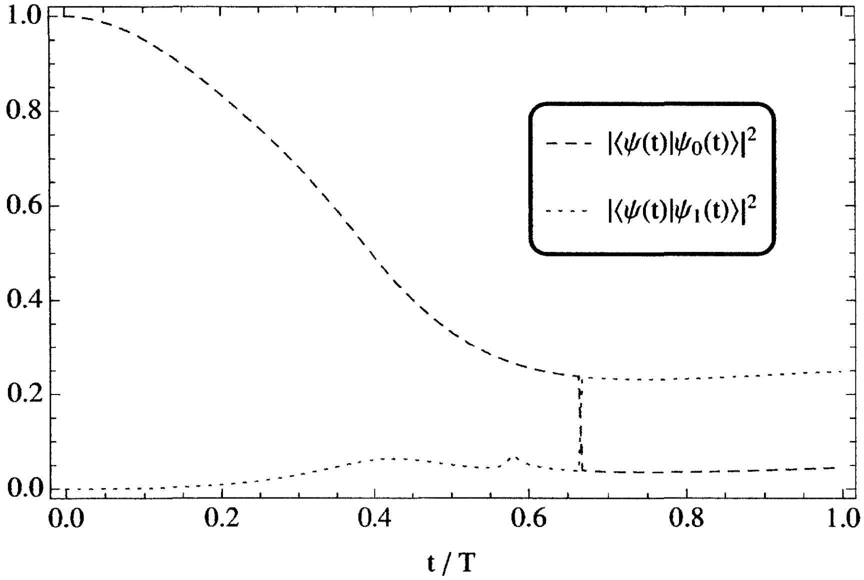


Figure 2.2.4: The overlap of the rapidly evolved system ($T = 10$) with the lowest two instantaneous energy eigenstates of $H(t)$, labeled here as $|\psi_0(t)\rangle$ and $|\psi_1(t)\rangle$. The bump in the overlap with the first excited state near $s = 0.58$ coincides with the avoided crossing between levels 2 and 3, as seen in figure 2.2.2.

Having described this phenomenon for a single instance, we now present evidence that it generalizes to many other hard instances. For each of our 137 hard instances we determined the location $T_{\text{max}}$ where the success probability is maximized in the interval $[0, 40]$, and in figure 2.2.5 we compare the success probability at $T_{\text{max}}$ with the success probability at $T = 100$. It is notable that every data point appears to the right of the 45 degree line, indicating that every one of our instances was improved by evolving the Hamiltonian more rapidly. The minimum improvement $P(T_{\text{max}})/P(100)$ for this batch of instances is 108, and the median improvement is 809.

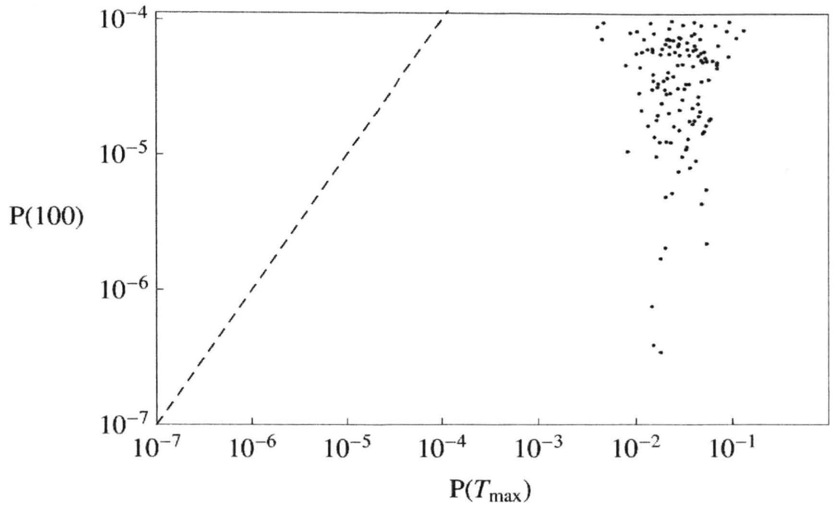


Figure 2.2.5: A log-log scatter plot comparing $P(T_{\max})$ with $P(100)$, where the value of $T_{\max}$ depends on the instance.

From an algorithmic perspective, it may not be possible to efficiently estimate the value of $T_{\max}$ for each instance in advance. The distribution of $T_{\max}$ for our 137 instances is shown in figure 2.2.6.

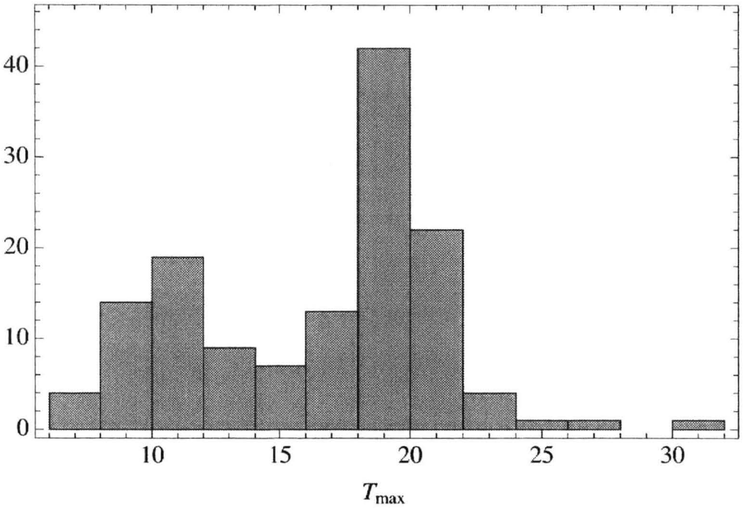


Figure 2.2.6: The distribution of the times $T_{max}$ at which the success probabilities of our hard instances are maximized in the interval $[0, 40]$.

The phenomenon we are describing is sufficiently robust that we can choose a fixed short time such as $T = 10$ and still gain a substantial improvement for every instance.

In figure 2.2.7 we compare the success probabilities at $T = 10$ and $T = 100$. Here the minimum improvement $P(10)/P(100)$ is 15, and the median improvement is 574.

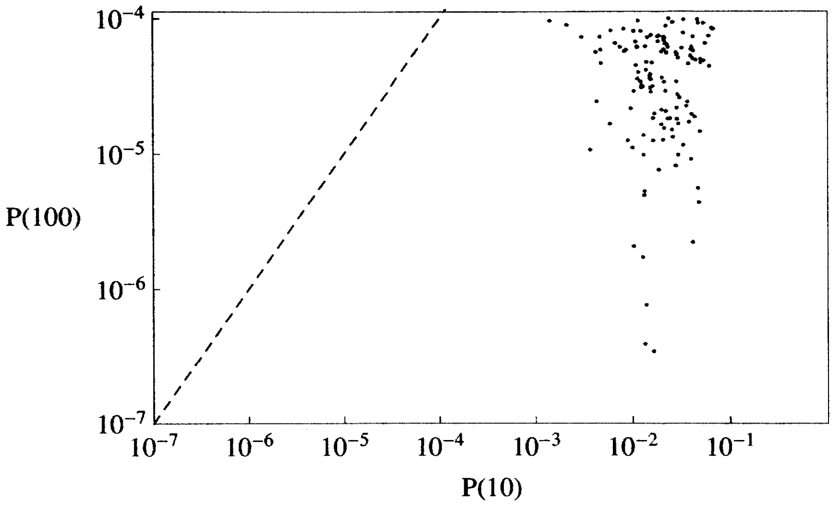


Figure 2.2.7: A log-log scatter plot comparing $P(10)$ with $P(100)$.

## 2.3 Going Lower by Aiming Higher

In the previous section we saw that having a substantial overlap with the first excited state before the avoided crossing increases the overlap with the ground state at the end of the evolution. In this section we attempt to directly exploit this effect by preparing the system at $t = 0$ to be in one of the 20 first excited states of $H_B$, obtained by taking the ground state (in equation 1.1.3) and flipping one of its qubits from $(|0\rangle+|1\rangle)/\sqrt{2}$ to $(|0\rangle-|1\rangle)/\sqrt{2}$. We did this for each of the 20 first excited states for each of our 137 hard instances. For each instance the average success probability over the 20 excited states of $H_B$ is given in figure 2.3.1, and the maximum success probability for every instance is given in figure 2.3.2.

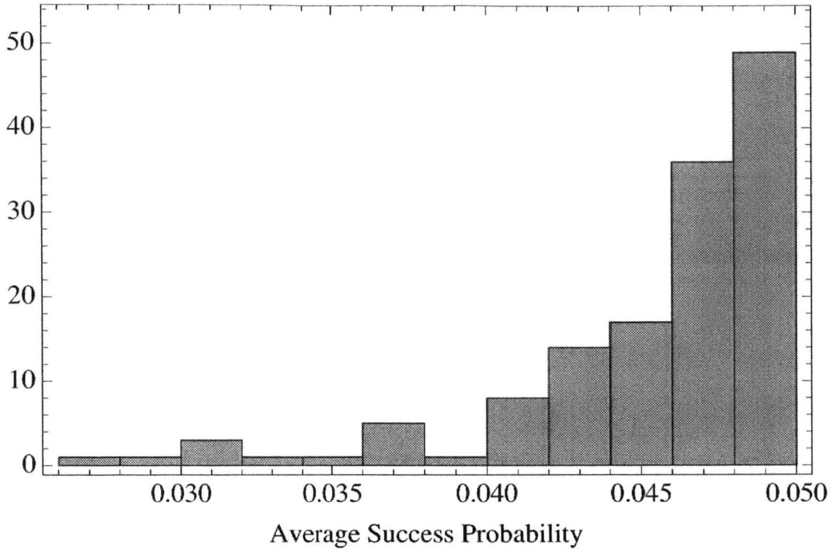


Figure 2.3.1: The average success probability at $T = 100$ for 137 instances obtained by initializing the system in each of the 20 first excited states of $H_B$.

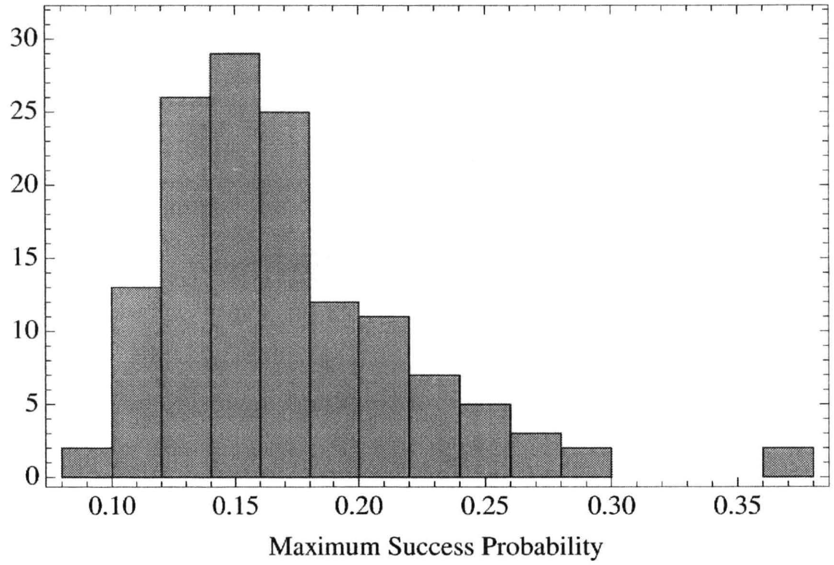


Figure 2.3.2: The maximum success probability at $T = 100$ for 137 instances obtained by initializing the system in each of the 20 first excited states of $H_B$.

As shown in figure 2.3.1, this strategy produces an average success probability near 1/20 for most of our 137 instances. This saturates the upper bound given by probability conservation, since the sum of the success probabilities associated with the 20 orthonormal initial states cannot exceed 1.

A similar strategy to ours was used in [43] to overcome an exponentially small gap in a particular Hamiltonian construction by initializing the system in a random low energy state. The authors argue that this technique is useful whenever there are a small number of low lying excited states that are separated from the remaining space by a large energy gap. The possibility of using non-adiabatic effects to drive a system from its ground state on one side of a phase transition to its ground state on the other side was considered in [44] as a problem in quantum control theory. Here we quantify the viability of this strategy for particularly hard instances of MAX 2-SAT at 20 bits.

## 2.4 The Meandering Path May Be Faster

The traditional time-dependent Hamiltonian in equation 1.1.4 represents a path in Hamiltonian space which is a straight line between $H_B$ and $H_P$. Here, as was previously considered in [45], we modify this path by adding an extra randomly chosen Hamiltonian $H_E$,

$$H(t) = \left(1 - \frac{t}{T}\right) H_B + \frac{t}{T}\left(1 - \frac{t}{T}\right) H_E + \frac{t}{T} H_P. \tag{2.4.1}$$

A reasonable constraint on $H_E$ is that it be a sum of local terms with the same interaction graph as the problem Hamiltonian $H_P$, but should not use any other information specific to the particular instance. We consider three categories of $H_E$:

1. **Stoquastic** with zeroes on the diagonal. Stoquastic matrices are defined by having non positive off-diagonal terms. We realize stoquastic $H_E$ in the following way: Each 2-local term of $H_E$ is a linear combination of 1 and 2-qubit Pauli operators from the set $\{I\sigma_x, \sigma_x I, \sigma_z\sigma_x, \sigma_x\sigma_z, \sigma_x\sigma_x, \sigma_y\sigma_y\}$. For each 2-local term, the 6 real coefficients are sampled from a Gaussian distribution with mean zero, and are then normalized so that their squares sum to 1. Moreover, the coef-

ficients are kept only if the local Hamiltonian term constructed in this way is stoquastic (i.e. all of the off-diagonal matrix elements are real and non-positive).

2. **Complex** with zeroes on the diagonal. Each 2-local term of $H_E$ is a linear combination of 1 and 2-qubit Pauli operators chosen from the set

   $$\{I\sigma_x, \sigma_x I, I\sigma_y, \sigma_y I, \sigma_z\sigma_x, \sigma_x\sigma_z, \sigma_x\sigma_x, \sigma_y\sigma_y, \sigma_z\sigma_y, \sigma_y\sigma_z, \sigma_y\sigma_x, \sigma_x\sigma_y\}$$

   . For each 2-local term, the 12 real coefficients are sampled from a Gaussian distribution with mean zero, and are then normalized so that their squares sum to 1.

3. **Diagonal**. Each 2-local term of $H_E$ is a linear combination of 1 and 2-qubit Pauli operators chosen from the set $\{I\sigma_z, \sigma_z I, \sigma_z\sigma_z\}$. For each 2-local term, the 3 real coefficients are sampled from a Gaussian distribution with mean zero, and are normalized so that their squares sum to 1.

The reason that we work with zero diagonal $H_E$ in the first two categories is to be sure that we are exploring purely quantum strategies for increasing the success probability, since the diagonal elements of $H_E$ could be seen as time-dependent classical modifications to the energy landscape of $H_P$. The reason that we separate stoquastic path change from general complex path change is that ground states of stoquastic Hamiltonians have various special properties which may limit their computational power. Ground state local Hamiltonian problems are known to have lower computational complexity when the Hamiltonians are restricted to be stoquastic [46][47]. Moreover, ground state properties of stoquastic Hamiltonians can be determined using Quantum Monte Carlo (a collection of classical methods for finding properties of quantum systems) at system sizes of up to a few hundred qubits (for a general review see [48], for an application to the QAA see [49]). Non-stoquastic Hamiltonians have a "sign problem" that prevents this, and we know of no efficient simulation techniques for non-stoquastic Hamiltonians at system sizes of more than roughly 20 qubits. The traditional QAA Hamiltonian defined by equations 1.1.1, 1.1.2, and 1.1.4 is stoquastic, and we are interested in seeing whether non-stoquastic path change can increase the computational power of this algorithm.

As a first demonstration of the potential for path change to increase success probabilities, we return to instance #1 which had $P(100) = 5 \times 10^{-5}$. In figure 2.4.1 we plot the spectrum for this instance with a particularly successful choice of complex $H_E$, and see that the avoided crossings in figure 2.2.2 have been eliminated.

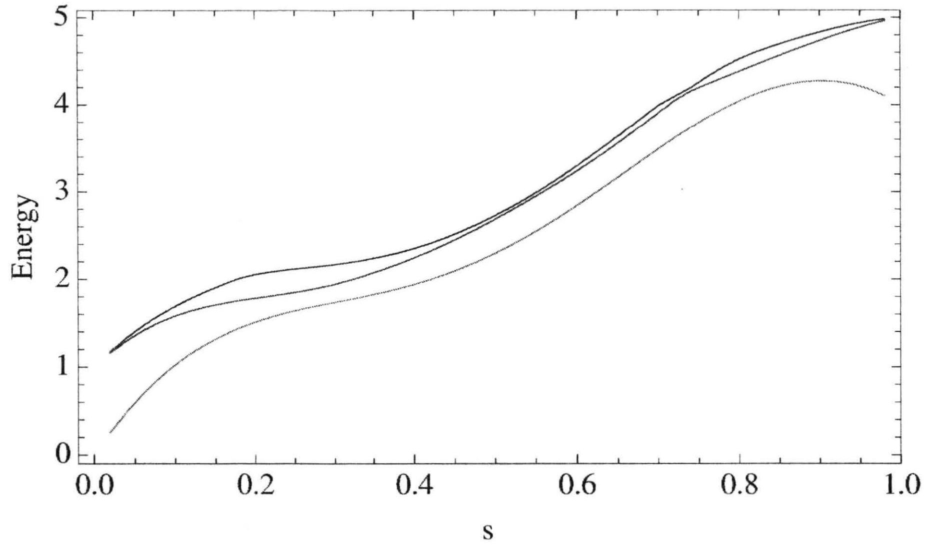


Figure 2.4.1: The energy spectrum of instance #1 with a particular choice of complex $H_E$ which gives $P(100) = 0.91$.

We tested the performance of this strategy by simulating 25 trials of stoquastic, complex, and diagonal path change for each of our 137 hard instances (which all have $P(100) < 10^{-4}$ when $H_E = 0$). The path changes are all chosen independently so that there are no correlations between the instances. Simulating these path change trials for all of our instances required over 25,000 hours of CPU time. The full distribution of success probabilities we obtained at $T = 100$ is given in figure 2.4.2.

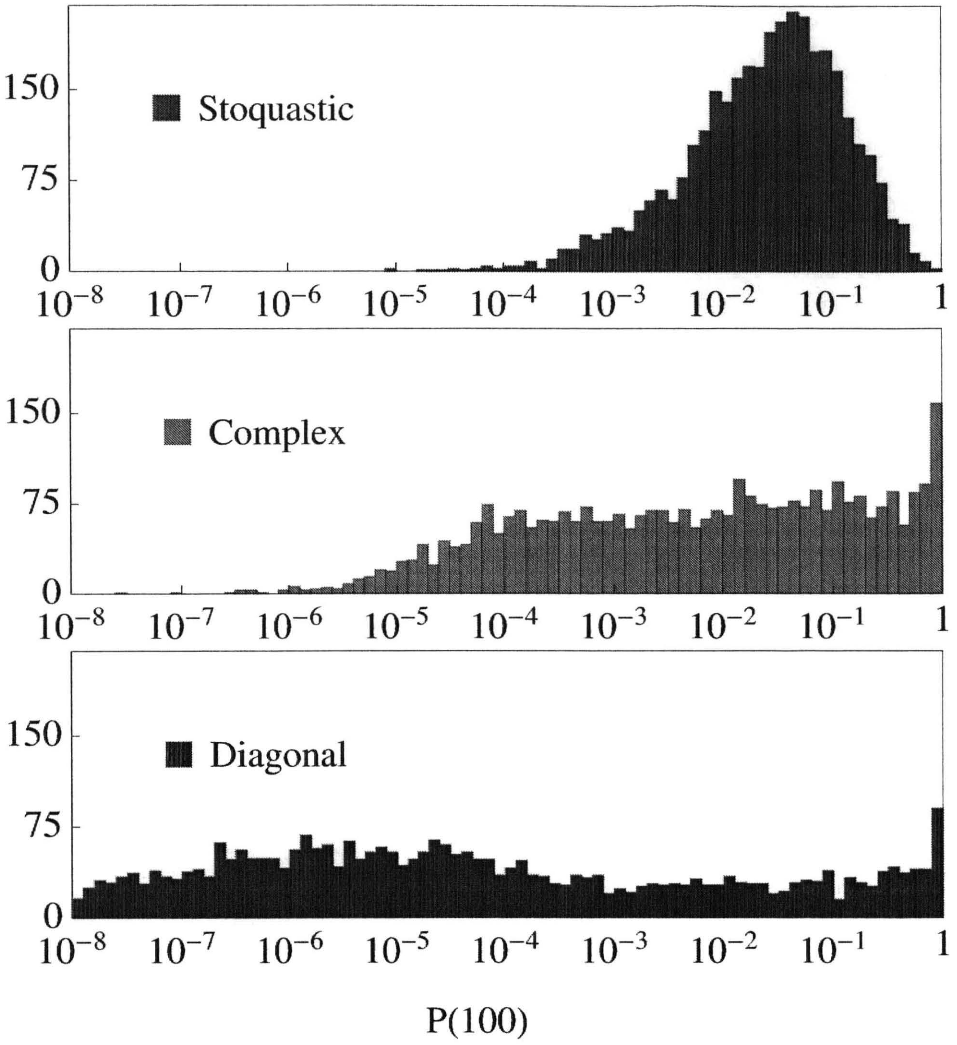


Figure 2.4.2: The distribution of success probabilities for 137 hard instances, when each is run with 25 randomly sampled path changes.

While stoquastic path changes almost always increase the success probability above $10^{-4}$, we see that they rarely produce success probabilities near 1. This is shown in the distribution of the maximum success probability we obtained for each instance with 25 trials of path change, shown in figure 2.4.3.

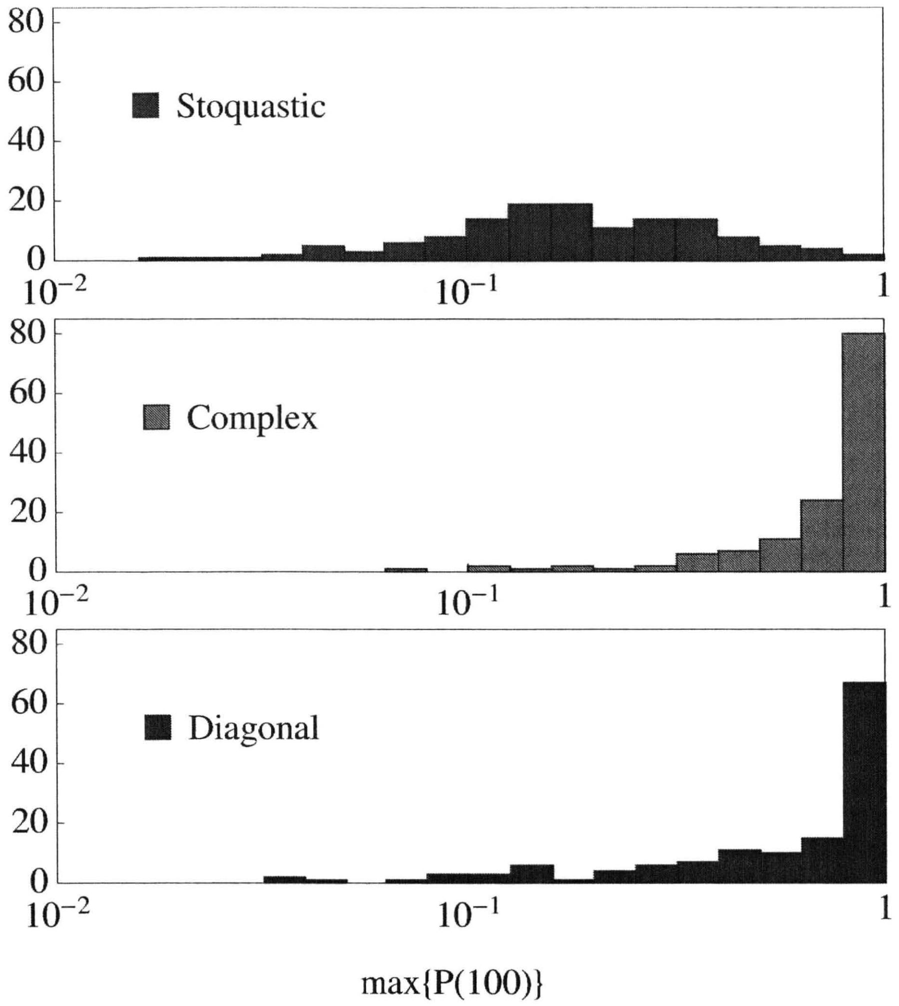


Figure 2.4.3: The maximum success probabilities for each of the 137 hard instances, when each is run with 25 randomly sampled path changes.

To take into account the spread in the distribution we estimate the geometric mean of the failure probabilities obtained by many trials of path change. For each instance we use the 25 trials to compute

$$\chi = \left( \prod_{i=1}^{25} \text{failure probability of the } i\text{-th trial} \right)^{1/25} . \qquad (2.4.2)$$

We take $1-\chi$ to be the effective success probability of a single trial of the path change strategy. In figure 2.4.4 we give the distributions of $1 - \chi$ for our ensemble of 137 hard instances.

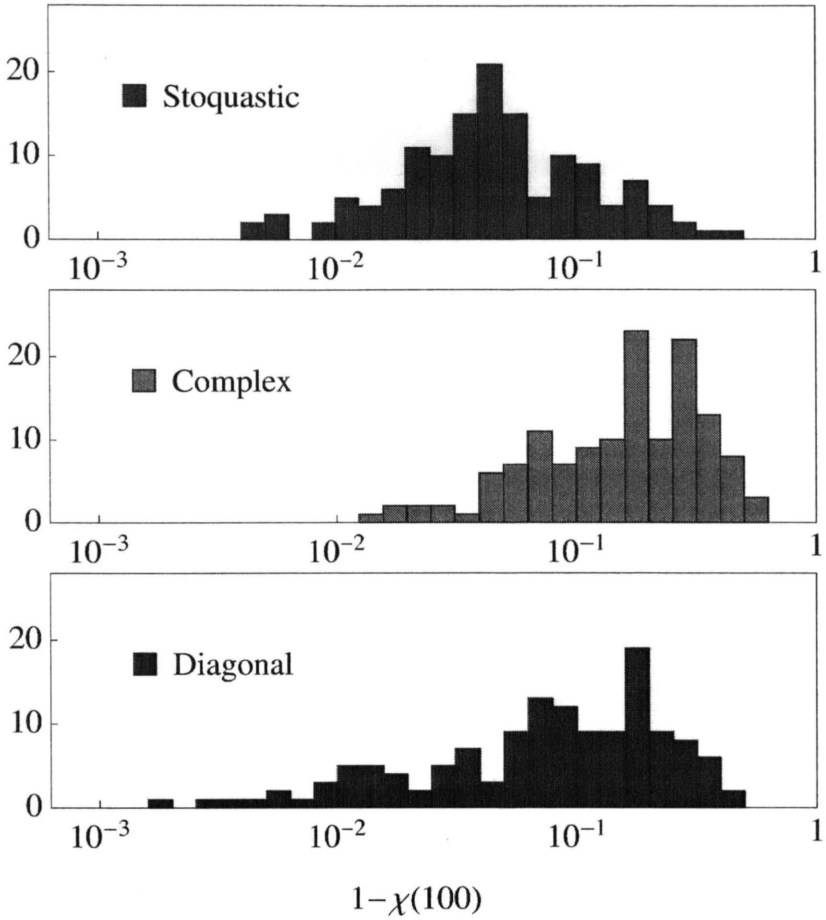


Figure 2.4.4: The effective success probabilities (given by 1 minus the geometric mean of the failure probabilities) obtained by running each of the 137 hard instances with 25 randomly sampled path changes.

We find that all three types of path change increase the effective success probability for all 137 of our hard instances, with complex path change typically producing the largest increase in the effective success probability.

To check whether the widening of the spectral gap seen in figure 2.4.1 also occurs for other successful trials of path change, we computed the minimum spectral gap $g_{min}$ between the ground state and the first excited state for a subset of our path change trials. For each of our 137 hard instances we computed $g_{min}$ for the most successful path change trial of each of the three types, and also for a randomly selected trial of each of the three types. Figure 2.4.5 compares these minimum gaps to the corresponding success probabilities.

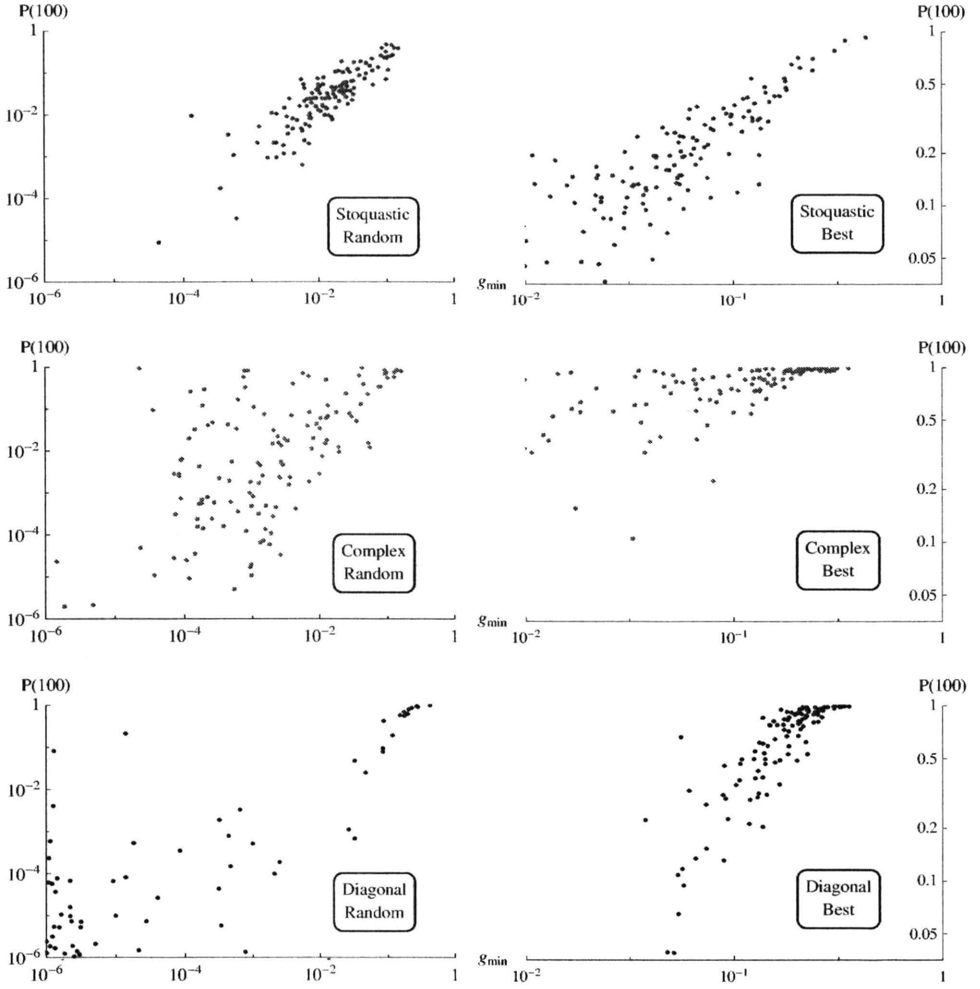


Figure 2.4.5: A comparison of success probabilities with the minimum spectral gap for several trials of path change. The plots in the left column contain one random path change trial for each instance. In the right column we plot the most successful path change trial for each instance. Note that the scales for the probabilities and the minimum gaps are different between the left column (random) and the right column (best).

We see a correlation between high success probability and large gaps, and almost no data with large gaps and low success probabilities.

## 2.5 Conclusion

We generated over 200,000 instances of MAX 2-SAT on 20 bits with a unique optimal satisfying assignment, and selected the subset for which the numerically exact time-simulation of QAA governed by equations 1.1.1 through 1.1.5 finds success probabilities of less than $10^{-4}$ at $T = 100$. We gave three strategies which increase the success probability for all of these instances. First we ran the adiabatic algorithm more rapidly, and observed an increased success probability at shorter times for all 137 instances. Second, we initialized the system in a random first excited state of $H_B$ and saw that the average success probability for this strategy is close to the upper bound 0.05 for the majority of hard instances. Finally, we observe that adding a random local Hamiltonian to the middle of the adiabatic path often increased the success probability, and that different types of path changes produced different distributions, with the stoquastic case most often increasing the success probability, the complex case being the most likely to give success probabilities close to 1, and the diagonal case having the most spread and highest likelihood of reducing the success probability.

To guard against the possibility that what we observe are low bit number phenomena, we also tested these strategies for the QAA version of the Grover search algorithm. Here the problem Hamiltonian $H_P$ assigns energy 1 to all of the computational basis states aside from one of them which is assigned energy 0. The Grover problem requires exponential time for any quantum algorithm, so we expect that our strategies should not improve the success probability. Indeed, at $n = 20$ qubits they do not.

One striking thing about these strategies is that they increase the success probability for all of the very low success probability instances we generated. This may be a consequence of testing our strategies on particularly hard instances, which have the most room for improvement. Figure 2.1.1 shows that the overwhelming majority of instances we generated at 20 bits are far easier than the ones we selected. At higher bit number it may be that most instances have very low success probability when the traditional QAA is run for a time that scales polynomially in the number of bits. If the only effect of these strategies is to bring the most difficult instances in line with the typical instances, which may in fact have very low success probabilities, then the algorithmic value of these strategies is limited. We hope that one day these strategies will be tested on a quantum computer running the Quantum Adiabatic Algorithm at high bit number where classical simulations are not available.

# Chapter 3

# Cloning Component Quantum Money Solves Graph Isomorphism

In this chapter we give a security proof to quantum money from knots. We will introduce and focus on the idealized version, component quantum money, where mixers and labelling functions are given as black boxes. We prove that component quantum money is secure unless there is a efficient quantum algorithm for GRAPH ISOMORPHISM. Therefore to counterfeit quantum money from knots, the counterfeiter must look into the inner workings of the mixer and labelling function or provide a fast algorithm for GRAPH ISOMORPHISM. This is joint work with Edward Farhi, Cedric Yen-Yu Lin, and Peter Shor.

## 3.1 Preliminaries

We call a function $f(x)$ *negligible* if $f(x) = o(1/x^n)$, $\forall n$. We call an algorithm *efficient* if the number of gates (and oracle calls, if any) it uses is polynomial in the size of the input. We denote the symmetric group of $\ell$ elements by $\mathrm{Sym}_\ell$. Recall that the fidelity between two quantum states is defined by $F(\rho,\sigma) = \mathrm{Tr}(\sqrt{\rho^{1/2}\sigma\rho^{1/2}})$. We say that two $m$-qubit quantum states $\rho$ and $\sigma$ are *negligibly different* if $F(\rho,\sigma) = 1 - f(m)$ for some negligible function $f$.

Throughout this chapter, within a ket the "+" and "−" signs always mean addition and subtraction modulo the dimension of the register. For example, we take $|a+b\rangle$ to mean $|(a+b) \pmod{D}\rangle$ if the register has dimension $D$. Similarly, $|a-b\rangle$ is shorthand for $|(a-b) \pmod{D}\rangle$.

### 3.1.1 Quantum Money

In this section, we formally define the meaning of a quantum money scheme.

**Definition 1** (Quantum Money Scheme). A quantum money scheme is defined by two efficient procedures:

1. How the mint generates quantum money states

2. How the public verifies quantum money states

Moreover, we say a quantum money scheme is *secure* if a counterfeiter cannot efficiently copy a quantum money state; given a state that passes the verification procedure, the counterfeiter cannot generate two states that both passes the verification procedure with high probability.

Scott Aaronson and Paul Christiano gave more detailed definitions in [9], including the definition of a mini-scheme and how to handle serial numbers. For the purpose of our work, the simple definition above suffices.

### 3.1.2 Component Mixers and Labelling functions

In this section, we introduce component mixers and labelling functions, two operations central to construction of component quantum money. The component mixer and labelling function of a component quantum money scheme are based on a partition $\{S_i\}$ of a large set $S \subseteq \{0,1\}^m$. We call the subpartitions $S_i$ *components* and index them by some set LAB. In other words, $S_i \cap S_j = \emptyset$ for $i \neq j$ and $S = \bigcup_{i \in \text{LAB}} S_i$. We now proceed to define the component mixer and labelling function corresponding to $\{S_i\}$:

**Definition 2** (Component Mixers). Let $\{M_a\}_{a \in \text{Ind}_M}$ be a family of one-to-one maps, indexed by a set $\text{Ind}_M \subseteq \{0,1\}^{\text{poly}(m)}$. We say $\{M_a\}$ is a *component mixer* on the partition $\{S_i\}_{i \in \text{LAB}}$ if all of the following hold:

- All of the $M_a$'s do not mix between components: for all $s \in S_i$ and $a \in \text{Ind}_M$, $M_a(s) \in S_i$.
- $\{M_a\}$ instantly mixes within each component: for all $s \in S_i$, if $a$ is taken uniformly random from $\text{Ind}_M$, the total variation distance between $M_a(s)$ and a uniform sample from $S_i$ is no more than $2^{-m-2}$.

We will often simply write the component mixer $\{M_a\}$ as $M$, i.e. we drop the subscript $a$ and the brackets.

For a concrete example, suppose $S$ is the set of all undirected graphs with $n$ vertices encoded by the adjacency matrices, with each component containing graphs that are isomorphic to each other. Then one component mixer for this partition is the set of all permutations of $n$ vertices: permutations only map between isomorphic graphs, and a random permutation maps a graph to a uniform sample from the same component. In this case, $\text{Ind}_M = \text{Sym}_n$, the symmetric group of $n$ elements.

In the case of quantum money from knots, $S$ is the set of oriented knots of a certain size encoded as grid diagrams. Each component $S_i$ is the set of knots with the same Alexander polynomial, a knot invariant. The component mixer are maps between isomorphic knots. However, there are several caveats. Firstly, since the size of a knot is an ambiguous idea, $S$ is actually a Gaussian distribution over different grid sizes with tails cut off. Secondly, the component mixer are actually approximated by a Markov chain of grid operations. Finally, we only mix between isomorphic knots instead of all knots with the same Alexander polynomial, so the mixer does not instantly mix. The first two issues are approximations of bounded error, and in the end of this chapter we show that our security proof can work around the last issue.

An algorithm having quantum query access to a component mixer basically means that it can coherently apply component mixer operations $M_a$ described in Definition 2. We formally define quantum query access to a component mixer as follows:

**Definition 3** (Quantum Query Access to a Component Mixer). An algorithm has *quantum query access to a component mixer* $\{M_a\}$ if the algorithm can do all of the following coherently:

- Test a string for membership in $\text{Ind}_M$.
- Create the uniform superposition over all elements of $\text{Ind}_M$,

$$|\text{Ind}_M|^{-1/2} \sum_{a \in \text{Ind}_M} |a\rangle.$$

- Perform the "controlled-$M$" operator $CM$, defined by

$$CM|\alpha, a, s\rangle = |\alpha, a, M_a^{\alpha}(s)\rangle$$

. Here $\alpha \in \{1, 0, -1\}$, $a \in \text{Ind}_M$, and $s \in S$.

In our security proof of general component money, we think $CM$ as a black box, and when we talk about "querying the component mixer", we mean applying $CM$.

Let us also define a *labelling function*: a function that, given an element $s \in S$, tells us which component $s$ is in:

**Definition 4** (Labelling Function). The labelling function $L : S \rightarrow \text{LAB}$ on the partition $\{S_i\}_{i \in \text{LAB}}$ is defined as follows: if $s \in S_i$, then $L(s) = i$.

Continuing from our previous example, suppose $S = \bigcup\limits_i S_i$ is the set of all $n$-vertex undirected graphs, where each $S_i$ contains graphs that are isomorphic to each other. Then an example of a labelling function for this partition would be $f(g) = g^{canon}$, where $g^{canon}$ is the canonical graph, a graph isomorphic to $g$ and with vertices ordered in some canonical fashion. Of course, computing a labelling function for isomorphic graphs is at least as difficult as solving the GRAPH ISOMORPHISM problem (see section 2.3), since to determine whether two graphs are isomorphic we could simply check whether they have the same canonical graph. There is therefore no known quantum or classical algorithm to efficiently compute such a labelling function.

For quantum money from knots, the labelling function is the Alexander polynomial, which can be computed efficiently on a classical computer. Therefore anyone with a quantum computer has quantum query access to the labelling function of quantum money from knots. Note that it is tempting to partition $S$ into isomorphic classes of knots and use the corresponding labelling function. But similar to the previous paragraph, we do not know how to efficiently calculate the canonical knots, which solves knot isomorphism. Tracing this line of logic back, this is why we partition $S$ according to Alexander polynomials instead and leads to the inability to instantly mix.

We also define quantum query access to a labelling function as the ability to perform the operation $|s, i\rangle \to |s, L(s) + i\rangle$ coherently.

### 3.1.3 Component Quantum Money

We now introduce *component quantum money,* as proposed in [8]. We assume the public (mint, users, and counterfeiters) have quantum query assess to the component mixer and the labelling function. Recall that a quantum money scheme consist of a procedure to generate money states and a procedure to verify money states. The procedure to generate component money state is:

**Algorithm 5** (Generate Component Money)**.** The mint uses the following steps to generate component quantum money states:

1. The mint creates the uniform superposition

$$|S|^{-1/2} \sum_{s \in S} |s\rangle.$$

   (We require that this can be done efficiently. For quantum money from knots, this requirement is trivial.)

2. The mint applies the labelling function with an ancilla register, giving

$$|S|^{-1/2} \sum_{i \in \mathrm{LAB}} \sum_{s \in S_i} |i\rangle |s\rangle.$$

3. The ancilla register is measured, collapsing the second register to the equal superposition of the measured random label $k$

$$|S_k\rangle \equiv |S_k|^{-1/2} \sum_{s \in S_k} |s\rangle.$$

The mint takes the state $|S_k\rangle$ to be the money state and appends $k$ as its serial number.

The notation $|S_k\rangle$ defined in the last step will be used throughout this chapter.

The mint repeats this procedure polynomially many times, each time generating a different money state $|S_k\rangle$; now the mint publishes a list of all valid serial numbers. For the money scheme to be secure, we require the number of labels $|LAB|$ to be superpolynomial in $m$, the number of qubits used to store the money state. In this case, a naive counterfeiter repeating the mint's procedure will fail to forge a money state because the probability of getting a serial number in the published list is negligible.

To show that a merchant can verify a valid money state with quantum query access to the mixer and labelling function, we use the following lemma proved in [8]:

**Lemma 6** (Projection using Component Mixer). *Given query access to a component mixer $\{M_a\}$ of a partition $\{S_i\}_{i \in LAB}$, a quantum computer can, with negligible error, measure the projector*

$$P = \sum_{i \in LAB} |S_i\rangle\langle S_i| = \sum_{i \in LAB} \left( \frac{1}{\sqrt{|S_i|}} \sum_{s \in S_i} |s\rangle \right) \left( \frac{1}{\sqrt{|S_i|}} \sum_{s \in S_i} \langle s| \right) \tag{3.1.1}$$

*with two queries to the component mixer. The labelling function is not required for this measurement.*

In other words, using the component mixer, we can efficiently measure the subspace spanned by the component money states of all labels. To verify that a quantum state is a valid money state, i.e. $|S_k\rangle$ with serial number $k$, a verifier checks the serial number using the labelling function and measures the projector $P = \sum_{i \in \mathrm{LAB}} |S_i\rangle\langle S_i|$.

Quantum money from knots is very similar to quantum money with component mixers. As mentioned previously, quantum money from knots partitions the set of knots of certain size by the Alexander polynomial; in other words, each component consists of knot diagrams of the same Alexander polynomial. A labelling function on this partition is provided; however the mixer only mixes between isomorphic knots, and therefore each component is further divided into subcomponents of isomorphic knots which the mixers actually mixes. This mismatch between the labelling function and the component mixer results in the verification procedure accepting many states that are not the bank note issued. Nevertheless, the recipes used to create and verify knot quantum money is very similar to that of a component mixer quantum money. In Section 3.3, we will describe the situation in detail and introduce *generalized component money*, which quantum money from knots is an example of. We will then show that our security proof holds for generalized component money, but the construction is unnecessarily complicated , so most of this chapter just discusses component quantum money. A careful reader might recall that quantum money from knots also suffers from the ambiguity of knot size and the component mixer being inexact. We argue that errors to those approximations are well bounded and can be neglected, following [1].

In this thesis, we give a security proof for component quantum money. Specifically, we are concerned about a powerful counterfeiter that can break *all* component quantum money schemes. Such a counterfeiter can be abstracted as a quantum algorithm that takes one copy of valid money as input, queries the corresponding component mixer and labelling function as black box quantum operations, and outputs two copies of valid quantum money in polynomial time. This algorithm can be treated as a family of circuits, with one circuit for each size of the money state. We will denote the circuit that takes as input an $m$-qubit money state, and queries mixer $M$ and labelling function $L$, as $Clone_m(M, L)$. The main result of this chapter shows that one can leverage *Clone* to efficiently solve GRAPH ISOMORPHISM.

### 3.1.4 Graph Isomorphism

The problem GRAPH ISOMORPHISM is defined as the following:

**Problem 7.** GRAPH ISOMORPHISM

Input: Two $n$-vertex graphs $g_1$ and $g_2$.

Output: 1 if $g_1$ is isomorphic to $g_2$, and 0 otherwise.

GRAPH ISOMORPHISM is in the complexity class NP, since one could give the isomorphism as a certificate. On the other hand, no known algorithm, classical or quantum, solves GRAPH ISOMORPHISM in polynomial time.

Considerable effort has been put into the search of an efficient quantum algorithm for GRAPH ISOMORPHISM. Although none of the attempts have yet to succeed, our work builds on a popular line of thought. The idea is to create the equal superposition of all isomorphic graphs $|X_g\rangle$ for a graph $g$ and then use a swap test to compare two such states. Seemingly feasible, all attempts to efficiently create the coherent superposition have failed. In the main result of this chapter, we leverage the powerful cloning algorithm *Clone* to efficiently create $|X_g\rangle$. To create $|X_g\rangle$, we introduce a ladder of partially mixed states $\left|X_g^\ell\right\rangle$, where $\left|X_g^1\right\rangle = |g\rangle, \left|X_g^n\right\rangle = |X_g\rangle$. We create the states $\left|X_g^\ell\right\rangle$ one by on, climbing up the ladder from $|g\rangle$ to $|X_g\rangle$ with the help of *Clone*. Now let us give the formal definitions:

**Definition 8** (Graph States and Subgraph States)**.** Given a graph $g$ with $n$ vertices, with some specified ordering of the vertices, let $|g\rangle$ be the quantum state of $g$ represented by its adjacency matrix, and let permutations $\pi \in \mathrm{Sym}_\ell$ act on $g$ by permuting the first $\ell$ vertices. We define the following states, omitting normalization factors:

- The *graph state* of $g$ is $|X_g\rangle \propto \sum_{\pi\in\mathrm{Sym}_n} |\pi g\rangle$
- The $n$ *subgraph states* are $\left|X_g^\ell\right\rangle \propto \sum_{\pi\in\mathrm{Sym}_\ell} |\pi g\rangle\ ,\ \ell = 1, \ldots, n$

When we write $\left|X_g^\ell\right\rangle$, we mean the normalized state.

*Remark* 9. $\left|X_g^1\right\rangle = |g\rangle, \left|X_g^n\right\rangle = |X_g\rangle$.

*Remark* 10 (Normalization of Subgraph States). We note the fact that overlap between consecutive subgraph states is polynomial sized. In fact,

$$\langle X_g^{\ell-1} | X_g^{\ell} \rangle \geq 1/n. \tag{3.1.2}$$

If the graph has no automorphism, $\langle X_g^{\ell-1} | X_g^{\ell} \rangle = 1/\ell$, the relation trivially holds. If the graph is invariant under some permutation, we need to be careful about the normalization. Let $C_{\ell,g}$ be the number of permutations of the first $\ell$ vertices that leave $g$ invariant. Then the sum $\sum_{\pi \in \mathrm{Sym}_\ell} |\pi g\rangle$ counts each distinct graph $C_{\ell,g}$ times, and so the correct normalization is

$$\left| X_g^{\ell} \right\rangle = \frac{1}{\sqrt{\ell!/C_{\ell,g}}} \left( \frac{1}{C_{\ell,g}} \sum_{\pi \in \mathrm{Sym}_\ell} |\pi g\rangle \right) = \frac{1}{\sqrt{C_{\ell,g}\ell!}} \sum_{\pi \in \mathrm{Sym}_\ell} |\pi g\rangle \, . \tag{3.1.3}$$

From the definition of $C_{\ell,g}$ it is clear that

$$C_{\ell,g} \geq C_{\ell-1,g} \tag{3.1.4}$$

and Eq. (3.1.2) follows.

As mentioned previously, if one could efficiently generate $|X_g\rangle$ for any graph $g$, one can solve graph isomorphism by doing a SWAP test on $|X_{g1}\rangle$ and $|X_{g2}\rangle$. If $g_1$ and $g_2$ are isomorphic, $|X_{g1}\rangle = |X_{g2}\rangle$. The graph states are orthogonal otherwise.

We will consider the subgraph states $\left| X_g^{\ell} \right\rangle$ as component money states: $S$ is the set of all $n$-vertex graphs and the component $S_i$ are graphs isomorphic under the permutation on the first $\ell$ vertices. The mixer is the set of permutation on the first $\ell$ vertices, which can be implemented efficiently even classically. The labelling function needs to map $g$ to some canonical representative of the set $\{\pi g | \pi \in \mathrm{Sym}_\ell\}$. However, implementing such a labelling function solves GRAPH ISOMORPHISM, so no one has quantum query access to the labeling function of this component quantum money scheme.

We can now provide a more detailed sketch of our security proof. First, inspired by Remark 10, we give a simple procedure that produces $|X_g^\ell\rangle$ from $|X_g^{\ell-1}\rangle$ with probability at least $1/n$. We can efficiently produce one copy of $|X_g^\ell\rangle$ from one copy of $|X_g^{\ell-1}\rangle$ by this procedure if we have the ability to copy $|X_g^{\ell-1}\rangle$. Secondly, we prove that we can trick the cloning algorithm *Clone* to copy a component money state even without access to the labelling function, i.e. having access to the mixer is enough. Therefore, we can copy $|X_g^\ell\rangle$ with *Clone*. Combining this result with the procedure of the first part, we are able to efficiently produce $|X_g\rangle$ starting from $|g\rangle$, and thus solve GRAPH ISOMORPHISM.

### 3.1.5 Random Functions and Quantum-Secure Pseudorandom Functions

A random function is defined as follows:

**Definition 11** (Random Function)**.** Let $[P] = \{0, \cdots, P-1\}$ and $[Q] = \{0, \cdots, Q-1\}$ be two discrete sets. Then we say a function $f : [P] \to [Q]$ is (uniformly) *random* if $f$ is chosen from the $Q^P$-sized set of functions from $[P]$ to $[Q]$ uniformly at random, i.e. each possible function is picked with probability $Q^{-P}$. Equivalently, for all $x \in [P]$, each $f(x)$ is independently chosen uniformly at random from $[Q]$.

We will use random functions in our work. Despite it often being said that randomness comes "for free" in quantum computing, it is unknown whether a quantum computer gains computational power by getting access to random functions of exponentially sized $P$ and $Q$. Even though a quantum computer can get random bits by measuring qubits, to construct a random function the quantum computer needs to guarantee that the same output is always returned on the same input, so it is not obvious how to do this.

Quantum-secure pseudorandom functions are deterministic functions that mimic random functions:

**Definition 12** (Quantum-Secure Pseudorandom Function (QPRF))**.** A *quantum-secure pseudorandom function* (QPRF) is a function $QPRF : [K] \times [P] \to [Q]$, $K = O(\text{poly}(P))$, which:

- is efficiently implementable on a quantum computer, and
- no quantum computer can distinguish between the following two cases with non-negligible probability:
    - oracle access to a random function from $[P]$ to $[Q]$
    - oracle access to the function $QPRF(k)$ treated as a function from $[P]$ to $[Q]$, where $k$ is uniformly chosen from the keyspace $[K] = \{0, \cdots, K-1\}$ and hidden.

Zhandry proved that if quantum secure one-way functions exist, then QPRFs exist. [50].

In the technical part of our security proof, Section (3.2.2), we use random functions to obscure the labelling function and fool *Clone* to work without a correct labelling function. When we use *Clone* to create the graph state $|X_g\rangle$, we replace the random function with a quantum-secure pseudorandom function, which we assume exists.

## 3.2 Main Theorem

Our main theorem is a security proof for component quantum money, which can be generalized to include quantum money from knots. [1]

**Theorem 13** (Main Theorem). *Suppose there is a quantum algorithm, a uniformly generated family of circuits $\{Clone_m(M, L)\}$, that given an $m$-qubit component money state $|\psi\rangle$ and black box access to the corresponding component mixer $M$ and labelling function $L$, outputs a state negligibly different from $|\psi\rangle^{\otimes 2}$ in polynomial time and polynomial number of queries. Then assuming quantum-secure pseudorandom functions exist, there exists an efficient quantum algorithm that solves GRAPH ISOMORPHISM.*

Informally, this theorem says that if there is a quantum algorithm *Clone* that can break all component quantum money schemes, without looking inside the black boxes of $M$ and $L$, then modulo some cryptographic assumptions, we can leverage *Clone* to solve GRAPH ISOMORPHISM.

We will show this in two steps. In the first step, we present an algorithm that solves GRAPH ISOMORPHISM by efficiently generating the graph state $|X_g\rangle$, defined in Definition 8, given a subroutine that clones the subgraph states $\left|X_g^\ell\right\rangle$ for a graph $g$. In the second step, we demonstrate how to clone the subgraph states $\left|X_g^\ell\right\rangle$ with *Clone*. We observe that the subgraph states $\left|X_g^\ell\right\rangle$ can be treated as component money states: each $\ell$ corresponds to a different component money scheme, $S$ is the set of graphs with the same number of vertices, and the components $S_i$ are the set of graphs isomorphic under permutation of first $\ell$ vertices, and the mixers are permutations of the first $\ell$ vertices. However, we do not have access to the labelling function of the subgraph states, since such a labelling function would already solve graph isomorphism. Therefore, if we apply *Clone* to the subgraph states, we cannot provide the correct labelling functions, so *Clone* is free to send back whatever state it likes, since *Clone* is only contracted to clone a state if you are able to provide both the mixer and the labelling function. This is fixed by generalizing a theorem of Lutomirski [8]. We show a way to modify a component money scheme by appending $p = \theta(m)$ bits, exponentially enlarge the state space $S$, and randomly permute the new labelling function. Because the state space is too large, *Clone* cannot check in polynomial time whether we provide the correct labelling function, so it is tricked to work even without the correct labelling function being provided.

The result does not directly apply to quantum money from knots, because quantum money from knots is not a component money scheme. However, they are really similar, so in Section 3.3 , we define *generalized component money*, of which quantum money from knots is an example, and show how our result could be applied to generalized component money.

### 3.2.1 Generating graph state $|X_g\rangle$ by cloning subgraph states

Here we give the first part of the proof of the Main theorem 13.

**Theorem 14** (Generating Graph State). *Define graph states $|X_g\rangle$ and subgraph states $|X_g^\ell\rangle$ as in Definition 8. Suppose there is an efficient quantum algorithm that can clone the subgraph states $|X_g^\ell\rangle$ for any graph $g$ and any $\ell \in \{1, \cdots, n\}$, that is, given a description of $g$, the integer $\ell$, and the state $|X_g^\ell\rangle$, the algorithm returns $|X_g^\ell\rangle|X_g^\ell\rangle$ with negligible error. Then there is another efficient quantum algorithm that, starting from scratch, generates the graph state $|X_g\rangle$, and thus solve GRAPH ISOMORPHISM.*

*Proof Sketch.* We show a procedure that, given a copy of $|X_g^{\ell-1}\rangle$, generates the state $|X_g^\ell\rangle$ with probability at least $1/\ell$. We can then generate the whole sequence of subgraph states, starting from $|X_g^1\rangle = |g\rangle$ and ending in $|X_g^n\rangle = |X_g\rangle$. The starting state $|g\rangle$ is easily generated. To obtain the state $|X_g^2\rangle$, we make a large number of copies of $|X_g^1\rangle$ (either by running the cloning algorithm, or by simply creating $|g\rangle$ multiple times) and run the procedure multiple times. Then we can make a large number of copies of $|X_g^2\rangle$ by using the given cloning algorithm and run the procedure to arrive at $|X_g^3\rangle$. We keeps making a large number of copies and trying for the next subgraph state until we reach $|X_g\rangle = |X_g^n\rangle$.

*Proof.* We demonstrate a quantum algorithm that, starting from scratch, generates the subgraph state $|X_g^\ell\rangle$, for $\ell$ from 1 to $n$. We show this by induction on $\ell$. The case $\ell = 1$ is immediate, since $|X_g^1\rangle = |g\rangle$. Now assuming we can generate the subgraph state $|X_g^{\ell-1}\rangle$ efficiently, we give a procedure to generate $|X_g^\ell\rangle$ given the state $|X_g^{\ell-1}\rangle$:

1. Clone one copy of $|X_g^{\ell-1}\rangle$. This can be done efficiently by the assumption of the theorem.

2. Prepare $\frac{1}{\sqrt{\ell}}\sum_{k=1}^{\ell} |k\rangle$ on ancilla qubits to get

$$\frac{1}{\sqrt{\ell}}\sum_{k=1}^{\ell} |k\rangle\,|X_g^{\ell-1}\rangle = \frac{1}{\sqrt{\ell}}\frac{1}{\sqrt{C_{\ell-1,g}(\ell-1)!}}\sum_{k=1}^{\ell}\sum_{\pi\in\mathrm{Sym}_{\ell-1}} |k\rangle\,|\pi g\rangle \tag{3.2.1}$$

where $C_{\ell,g}$ is defined in Remark 10.

3. Perform a controlled transposition between vertex $\ell$ and vertex $k$ to get

$$\frac{1}{\sqrt{\ell}}\frac{1}{\sqrt{C_{\ell-1,g}(\ell-1)!}}\sum_{k=1}^{\ell}\sum_{\pi\in\mathrm{Sym}_{\ell-1}}|k\rangle\,|(\ell k)\,\pi g\rangle \tag{3.2.2}$$

   where $(\ell k)$ denotes the permutation that is the transposition of vertices $\ell$ and $k$, and $(\ell k)\pi g$ is the graph arrived at by applying the permutations $\pi$ followed by $(\ell k)$ to the graph $g$.

4. Perform a quantum Fourier transform on the ancilla qubits to get

$$\frac{1}{\sqrt{\ell}}\frac{1}{\sqrt{C_{\ell-1,g}(\ell)!}}\sum_{k=1}^{\ell}\sum_{\pi\in\mathrm{Sym}_{\ell-1}}\sum_{j=1}^{\ell}e^{2\pi ijk/\ell}\,|j\rangle\,|(\ell k)\,\pi g\rangle \tag{3.2.3}$$

5. Measure the ancilla qubits. With probability $\frac{1}{\ell}\frac{C_{\ell,g}(\ell)!}{C_{\ell-1,g}(\ell)!}\geq\frac{1}{\ell}$ , we measure $j=0$ obtain the state

$$\begin{aligned}&|0\rangle\left(\frac{1}{\sqrt{C_{\ell,g}(\ell)!}}\sum_{k=1}^{\ell}\sum_{\pi\in\mathrm{Sym}_{\ell-1}}|(\ell k)\,\pi g\rangle\right)\\ &=|0\rangle\left(\frac{1}{\sqrt{C_{\ell,g}(\ell)!}}\sum_{\pi\in\mathrm{Sym}_{\ell}}|\pi g\rangle\right)\\ &=|0\rangle\,|X_g^{\ell}\rangle\,.\end{aligned} \tag{3.2.4}$$

   Otherwise we start over from step 1. The procedure is expected to succeed in $O(\ell)$ iterations.

If the cloning algorithm has negligible error, the above algorithm to generate the graph state $|X_g\rangle$ also has negligible error, since the algorithm only takes polynomially many steps. □

### 3.2.2 Cloning without querying labelling functions

In this section, we give the second part of the proof of the Main theorem 13, which is a generalization of the main theorem in [8], modified mainly to make it time efficient.

Colloquially, we show that if there is a cloning algorithm *Clone* which guarantees to clone any component money state wherever we supply it with the corresponding mixer and labelling function, we can trick it to clone any component money state, supplying the mixer but not the labelling function. This is done by extending the space storing the money state and hide the new labelling function with a random permutation. Assuming the money state we want to clone is $m$-qubit, we append $p = \theta(m)$ ancilla qubits to it, making the dimension of the state space $2^p$ times larger. We define a new partition on the enlarged state space which trivially extends the original partition, so that the original money state appended with zeros is still a money state in the new partition, and the new mixer can be easily simulated with a few queries to the original mixer. *Clone* also works on the new $(m+p)$-qubit component money scheme because it works on all component money schemes. Now we apply a random permutation to the appended money state along with the new partition, hiding the position of the original partition. Now since *Clone* runs in polynomial time, it cannot find the original partition in the exponentially large space by the Grover search lower bound [11]. Therefore even if we lie on the labels of parts corresponding to the original partition, *Clone* must still work with negligible error, and that is what we do. We ask *Clone* to copy our money state in this randomly permuted large space, giving it the correct mixer and a carefully crafted labelling function that is correct on all the trivial extension part but lies on the original partition. Because *Clone* cannot caught us lying, it must copy the state as its guarantee. Finally we revert the random permutation on *Clone*'s output, getting two copies of the original component money state.

Here is a more detailed sketch of the proof. Suppose we would like to clone a component money state with support $S = \{0,1\}^m$, partition $\{S_i\}$, and component mixer $\{M_a\}$. Without loss of generality, we assume that the component money state we would like to clone is $|S_1\rangle \equiv \sum\limits_{s \in S_1} |s\rangle$.

We proceed by extending the support space to $S' = \{0,1\}^p \times S$, and defining a new money scheme on this space. We treat this new support as having two indices, the first index being a number from 0 to $2^p - 1$, and the second index an element of

$S$. We will sometimes visualize $S'$ as a rectangle, with $2^p$ rows and $|S| = 2^m$ columns.

Now we want to apply the circuit $Clone_{m+p}$, which clones any $(m+p)$-qubit money state, on $S'$. To this end we partition $S'$ into the following components :$\{0\} \times S_i$, for all $i$, and $\{(r,s)\}$, for all $r > 0$ and all $s \in S$. In other words, the components of the original money scheme are placed in the first row, while every other element outside of the first row is its own component. We can easily implement the corresponding component mixer

$$M_a^{incon}(r,s) = \begin{cases} (0, M_a(s)) & r = 0 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.2.5}$$

However, implementing the corresponding labeling function requires us to label the original partition of $S$, which we cannot do. But we can implement the following labelling function which is *inconsistent* with $M^{incon}$:

$$L^{incon}(r,s) = \begin{cases} \star & r = 0 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.2.6}$$

The set of labels is contained in $(\{0,1\}^p \times S) \cup \{\star\}$, where $\star$ is a special symbol. $L^{incon}$ correctly labels most of the components, but gives a blank answer on original partition, which is a exponentially small fraction. $M^{incon}$ and $L^{incon}$ are gates that we can and will implement in our algorithm to clone the component money state, but since $L^{incon}$ is *inconsistent* with $M_a^{incon}$, there is no guarantee that $Clone_{m+p}$ works with $M^{incon}$ and $L^{incon}$. More precisely, $L^{incon}$ fails to distinguish $\{0\} \times S_i$ between different $i$, and therefore $L^{incon}$ violates the promise that two elements have the same label if and only if they are from the same component.

We want to argue that giving $Clone_{m+p}$ $M^{incon}$ and $L^{incon}$ is good enough. To this end, we define the the ideal mixer

$$M_a^{ideal}(r,s) = \begin{cases} (0, M_a(s)) & r = 0 \,\text{and}\, s \in S_1 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.2.7}$$

and the ideal labelling function

$$L^{ideal}(r,s) = \begin{cases} \star & r = 0 \,\mathrm{and}\, s \in S_1 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.2.8}$$

Contrary to $M^{incon}$ and $L^{incon}$ , $M^{ideal}$ and $L^{ideal}$ are "ideal" in the sense that they are a valid mixer and labelling function for $|0\rangle|S_1\rangle$, and hence would clone this money state when used with $Clone_{m+p}$. Specifically, $M^{ideal}$ and $L^{ideal}$ are the component mixer and labelling function for the following partition of $S'$ : $\{0\} \times S_1$, and $\{(r,s)\}$ for all other elements of $S'$. In other words, we keep the component corresponding to the money state at the first row, and trivially extend every other element to its own component. Since the partition depends on the label of the money state we want to clone, without access to the labelling function we cannot and will not implement either $M^{ideal}$ or $L^{ideal}$. They are just auxiliary constructions for the proof.

Since the differences between $\{M^{incon}, L^{incon}\}$ and $\{M^{ideal}, L^{ideal}\}$ occur only on an exponentially small ($2^{-p}$) fraction of the support space (namely $\{0\} \times S$), it is quite possible that these differences will not be detected by the cloner at all. However, an algorithm checking the first row will quickly found their difference. Therefore we hide the difference by applying a random cyclic permutation on each column.

We formally define "cyclic permutation on each column" as follows: define $\mathcal{F}$ as the set of all functions from $S$ to $\{0, \cdots, 2^p - 1\}$. For each $\lambda \in \mathcal{F}$, define a classical function $K_\lambda : S' \to S'$ and corresponding unitary operator $\tilde{K}_\lambda$ as follows:

$$\begin{aligned} K_\lambda(r,s) &= (r + \lambda(s), s) \\ \tilde{K}_\lambda|r,s\rangle &= |r + \lambda(s), s\rangle, \end{aligned} \tag{3.2.9}$$

where addition is taken modulo $2^p$. In other words, $\tilde{K}_\lambda$ cyclically shifts the contents of the $s$-th column by $\lambda(s)$ (see Figure 3.2.1). We now formally define the meaning of applying a permutation to the algorithm:

**Definition 15** (Apply $\lambda$). When we apply a function $\lambda$, we modify the cloning algorithm $Clone_{m+p}$ into the following steps:

1. Apply $\tilde{K}_\lambda$ to the starting state $|0\rangle|S_1\rangle$.
2. Change the mixer $M$ to $K_\lambda M K_\lambda^{-1}$.
3. Change the labelling function $L$ to $LK_\lambda^{-1}$. Also relabel the singleton components so their labels do not give away any possible incriminating information to the cloner. The relabelling is defined in details in Algorithm 16.
4. Apply the cloning algorithm $Clone_{m+p}$ to the current state, with the modified component mixer and labelling function.
5. At the end of the algorithm, apply $\tilde{K}_\lambda^{-1} \otimes \tilde{K}_\lambda^{-1}$ to the final (cloned) state.

We will "apply $\lambda$" to either the *inconsistent* or *ideal* algorithm. The above mixer and labelling function $\{M, L\}$ are either "inconsistent", $\{M^{incon}, L^{incon}\}$ or "ideal", $\{M^{ideal}, L^{ideal}\}$. We claim that if we apply a random $\lambda \in \mathcal{F}$, the action of $M^{incon}$ and $L^{incon}$ is "hidden" on the components $\{0\} \times S_i$ for any $i$ except $i = 1$. Therefore, we will argue that the "inconsistent" mixer and labelling function are indistinguishable from the "ideal" versions. Intuitively, for any column $s \notin S_1$, $K_\lambda M^{incon} K_\lambda^{-1}$ and $K_\lambda M^{ideal} K_\lambda^{-1}$ differ only at the random location $r = \lambda(s)$.[1] We will show, in the manner of the BBBV lower bound for Grover's problem [11], that we need $\Omega(\sqrt{2^p})$ queries to distinguish them, while $Clone_{m+p}$ only queries them poly$(m+p)$ times. Therefore with a reasonably large $p$ (such as $p = \Theta(m)$), we cannot distinguish between $M^{incon}$ and $M^{ideal}$. A similar reasoning applies for the relabelled versions of $L^{incon}K_\lambda^{-1}$ and $L^{ideal}K_\lambda^{-1}$.

Since $M^{ideal}$ and $L^{ideal}$ are the component mixer and labelling function for the partition $S'$ : $\{0\} \times S_1$, and $\{(r, s)\}$ for all other elements of $S'$, $Clone_{m+p}(M^{ideal}, L^{ideal})$ will clone the starting state $|0\rangle|S_1\rangle$. Note that $Clone_{m+p}(M^{ideal}, L^{ideal})$ clones $|0\rangle|S_1\rangle$ even with the application of any $\lambda$ by construction: $K_\lambda M^{ideal} K_\lambda^{-1}$ and the relabelled $L^{ideal}K_\lambda^{-1}$ are a valid component mixer and labelling function pair for $\tilde{K}_\lambda|0\rangle|S_1\rangle$, and

[1] The $S_1$ component is different because the algorithm gets a clue of its location from the permuted money state $\tilde{K}_\lambda|0\rangle|S_1\rangle$.

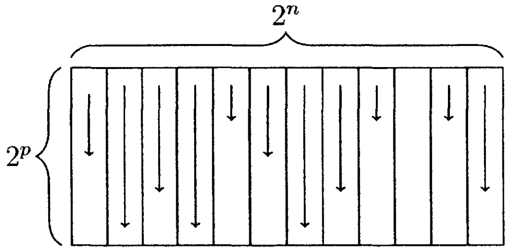


Figure 3.2.1: Schematic drawing of an example of $K_\lambda$. The effect of $K_\lambda$ is to shift the contents of each column independently.

hence the $Clone_{m+p}$ algorithm always works as expected at step 4, giving the state $(\tilde{K}_\lambda|0\rangle|S_1\rangle)^{\otimes 2}$. Applying $(\tilde{K}_\lambda^{-1})^{\otimes 2}$ in step 5 will thus give the state $(|0\rangle|S_1\rangle)^{\otimes 2}$. Because $Clone_{m+p}(M^{ideal}, L^{ideal})$ is indistinguishable to $Clone_{m+p}(M^{incon}, L^{incon})$ with a random $\lambda$, $Clone_{m+p}(M^{incon}, L^{incon})$ must clone the state, too.

We now formally define our algorithm that clones component money states with no calls to the corresponding $L$:

**Algorithm 16** (Label-Free Cloning Algorithm)**.** The following algorithm clones component money states without access to the corresponding labelling function, given the access to a random oracle

*Input.*

- An $m$-qubit money state $|S_1\rangle = \frac{1}{\sqrt{|S_1|}} \sum_{s \in S_1} |s\rangle$.
- Quantum query access to the component mixer $\{M_a\}$, but not an associated labelling function.
- $Clone_{m+p}$, a circuit that efficiently clones any $(m+p)$-qubit quantum money by querying its component mixer and labeling function. We will assume that the output of the circuit is a pure state, for which the first $2(m+p)$ qubits contain the two copies of the money state.

*Output.* A state negligibly different from $(|0\rangle|S_1\rangle)^{\otimes 2}$.

*Procedure.*

1. Append a $p$-qubit register initialized to 0, so we have $|0\rangle|S_1\rangle$. Append an $(m+p)$-qubit register initialized to 0. This register is where the cloned copy of the money goes to. Append ancilla qubits required by $Clone_{m+p}$.

2. Choose $\lambda \in \mathcal{F}$ uniformly randomly, where $\mathcal{F}$ is the set of all functions from $S$ to $\{0, \cdots, 2^p - 1\}$. Apply $\tilde{K}_\lambda$ to $|0\rangle|S_1\rangle$, getting the state $\tilde{K}_\lambda|0\rangle|S_1\rangle = \sum_{s \in S_1} |\lambda(s)\rangle|s\rangle$.

3. Apply $Clone_{m+p}(K_\lambda M^{incon} K_\lambda^{-1}, L_\lambda^{incon})$ to the current state, where $L_\lambda^{incon}$ is defined as
$$L_\lambda^{incon}(r, s) = \begin{cases} \star & r = \lambda(s) \\ (r, s) & \text{otherwise,} \end{cases} \tag{3.2.10}$$
and $M^{incon}$ is defined as
$$M_a^{incon}(r, s) = \begin{cases} (0, M_a(s)) & r = 0 \\ (r, s) & \text{otherwise.} \end{cases} \tag{3.2.11}$$
Quantum queries of $L_\lambda^{incon}$ can be implemented efficiently from $\lambda$. Note that $L_\lambda^{incon}$ and $L^{incon} K_\lambda^{-1}$ label the same set of components, but give different labels to the singleton components. We need to relabel $L^{incon} K_\lambda^{-1}$ into $L_\lambda^{incon}$ so the singleton components do not give away information of $\lambda$. See the following Remark 17 for details. Quantum queries of $M_a^{incon}$ can be implemented efficiently from $M_a$ and $\lambda$.

4. Apply $\tilde{K}_\lambda^{-1} \otimes \tilde{K}_\lambda^{-1}$ to the first $2(m+p)$ qubits of the current state; these qubits will contain a state negligibly different from $(|0\rangle|S_1\rangle)^{\otimes 2}$.

*Remark* 17 (Relabelling $L^{incon} K_\lambda^{-1}$). Compare $L_\lambda^{incon}$, the labelling function used in our algorithm, with $L^{incon} K_\lambda^{-1}$. The latter is
$$L^{incon} K_\lambda^{-1}(r, s) = \begin{cases} \star & r = \lambda(s) \\ (r - \lambda(s), s) & \text{otherwise.} \end{cases} \tag{3.2.12}$$

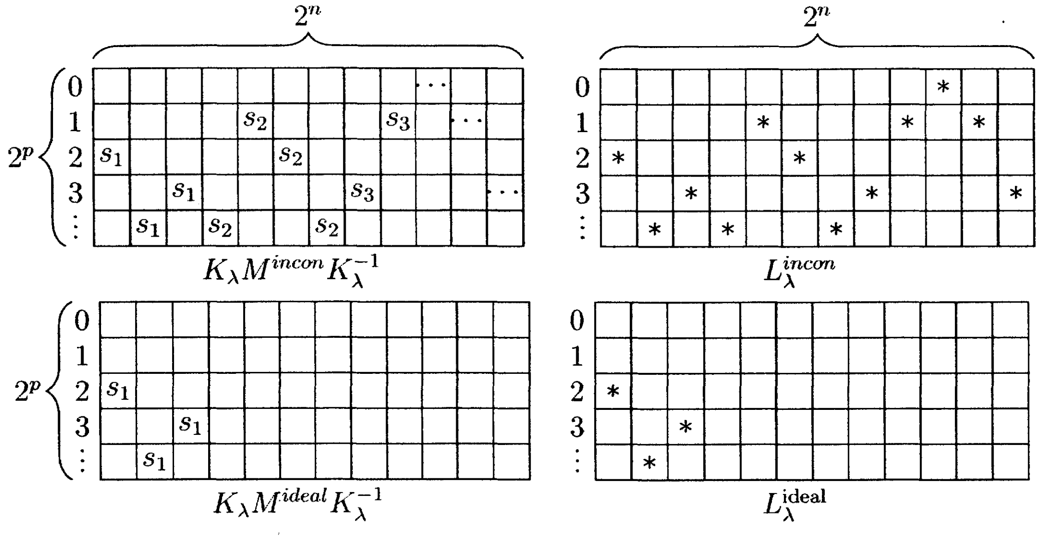

$2^n$
$2^n$
$2^p$
0
1
2
3
$s_1$
$s_2$
$s_3$
$\cdots$
$*$
$K_\lambda M^{incon} K_\lambda^{-1}$
$L_\lambda^{incon}$
$2^p$
$K_\lambda M^{ideal} K_\lambda^{-1}$
$L_\lambda^{\text{ideal}}$

Figure 3.2.2: Schematic drawings of the components represented by $K_\lambda M^{incon} K_\lambda^{-1}$, $L_\lambda^{incon}$, $K_\lambda M^{ideal} K_\lambda^{-1}$, and $L_\lambda^{ideal}$. $s_i$ represents an element of the component $S_i$, while blank squares are singleton components. Here the shift function $\lambda$ is the one represented in Figure 3.1. Note that the components of $K_\lambda M^{ideal} K_\lambda^{-1}$ and $L_\lambda^{ideal}$ match.

If we were to use $L^{incon} K_\lambda^{-1}$ in our algorithm, then the original cloner $Clone_{m+p}$ could easily gain information about $\lambda$, since $L^{incon} K_\lambda^{-1}$ maps $(r, s)$ to $(r - \lambda(s), s)$ for almost all inputs. We hide this information from the cloner by *relabelling* the label of all singleton components $\{(r, s)\}$ from $(r - \lambda(s), s)$ to $(r, s)$, i.e. by using $L_\lambda^{incon}$ in place of $L^{incon} K_\lambda^{-1}$. The relabelling can be easily done.

*Proof of validity of Algorithm 16.* Denote $\psi_\lambda^{incon}$ as the full $2(m+p)$-qubit-plus-ancilla state we obtain from Algorithm 16, for a particular choice of $\lambda$ in step 2. We will write $|\psi\rangle$ as $\psi$ to reduce cluttering. We will define an *ideal* mixer and labelling function momentarily; if we use them instead in step 3, denote similarly the output state as $\psi_\lambda^{ideal}$. The ideal mixer and labelling function are defined as follows:

$$L_\lambda^{ideal}(r, s) = \begin{cases} \star & r = \lambda(s), s \in S_1 \\ (r, s) & \text{otherwise,} \end{cases} \tag{3.2.13}$$

$$M_a^{ideal}(r,s) = \begin{cases} (0, M_a(s)) & r = 0, s \in S_1 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.2.14}$$

The idealized version of the algorithm is a construction for this proof and is not meant to be implemented since we are assuming that we do not have access to a labelling function that can identify elements of $S_1$. See Figure 3.2.2 for a sketch of the difference between the ideal and inconsistent labelling functions.

We will need the following facts:

1. $E_\lambda \left\| \psi_\lambda^{incon} - \psi_\lambda^{ideal} \right\|^2 \approx 0$, where $E_\lambda$ stands for the expectation over uniformly random $\lambda$.

2. For all functions $\lambda$, we have that $\psi_\lambda^{ideal} = (|0\rangle|S_1\rangle)^{\otimes 2} \otimes |work_\lambda\rangle$ for some final state of the work bits $|work_\lambda\rangle$ (the state $(|0\rangle|S_1\rangle)^{\otimes 2}$ is our desired output).

Fact 2 is immediate by construction: note that $K_\lambda M^{ideal} K_\lambda^{-1}$ and $L_\lambda^{ideal}$ are a valid component mixer and labelling function pair for $\tilde{K}_\lambda|0\rangle|S_1\rangle$, and hence the ideal algorithm always clones as expected at step 3, giving the state $(\tilde{K}_\lambda|0\rangle|S_1\rangle)^{\otimes 2} \otimes |work_\lambda\rangle$. Applying $(\tilde{K}_\lambda^{-1})^{\otimes 2}$ to the first $2(m+p)$ qubits in step 4 will thus give the state $\psi_\lambda^{ideal} = (|0\rangle|S_1\rangle)^{\otimes 2} \otimes |work_\lambda\rangle$, as expected.

In the following, we will prove the first fact. We will then conclude from these two facts that after discarding the ancilla qubits, the final state of Algorithm 16, $\mathrm{Tr}_{work} E_\lambda |\psi_\lambda^{incon}\rangle\langle\psi_\lambda^{incon}|$, is negligibly different from $(|0\rangle|S_1\rangle)^{\otimes 2}$.

To prove that $E_\lambda \left\| \psi_\lambda^{incon} - \psi_\lambda^{ideal} \right\|^2 \approx 0$, we use an argument similar to the BBBV lower bound for the Grover problem[11]. The presentation here follows that of Nielsen and Chuang [51]. We define the permuted oracle $O_\lambda$ as a quantum query to either $K_\lambda M K_\lambda^{-1}$ or $L_\lambda$ as called for by the algorithm. Let the input state to the original cloner, i.e. the state after step 2 of our algorithm, be $|\psi_\lambda^{start}\rangle = \tilde{K}_\lambda|0\rangle|S_1\rangle \otimes |0\rangle|0\rangle \otimes |0_{ancilla}\rangle$. Define the intermediate states after $k$ oracle calls:

$$\begin{aligned} \psi_{\lambda,k}^{incon} &\equiv U_k O_\lambda^{incon} U_{k-1} O_\lambda^{incon} \ldots O_\lambda^{incon} U_0 |\psi_\lambda^{start}\rangle \\ \psi_{\lambda,k}^{ideal} &\equiv U_k O_\lambda^{ideal} U_{k-1} O_\lambda^{ideal} \ldots O_\lambda^{ideal} U_0 |\psi_\lambda^{start}\rangle, \end{aligned} \tag{3.2.15}$$

where $U_0, U_1, \dots, U_k$ are unitary operations implemented by algorithm 16 between $O_\lambda$. We also define the progress function

$$D_k \equiv E_\lambda \left\|\psi_{\lambda,k}^{incon} - \psi_{\lambda,k}^{ideal}\right\|^2. \tag{3.2.16}$$

Intuitively, $D_k$ is the average difference between the idea and the inconsistent cases. Assume the algorithm makes $T = \text{poly}(m+p)$ queries in total, and note that $\psi_\lambda^{incon} = (\tilde{K}_\lambda^{-1} \otimes \tilde{K}_\lambda^{-1})\psi_{\lambda,T}^{incon}$ and $\psi_\lambda^{ideal} = (\tilde{K}_\lambda^{-1} \otimes \tilde{K}_\lambda^{-1})\psi_{\lambda,T}^{ideal}$. We prove that the progress function in upper bounded by a quadratic function:

**Lemma 18** (Upper bound on progress function).

$$D_k \le \frac{4k^2}{2^p}. \tag{3.2.17}$$

*$D_k$ is the progress function defined in Eq. (3.2.16)*

*Proof.* We give an inductive proof. The base clase of the induction is clearly true; when $k = 0$, $D_k = 0$. For the inductive step, note that

$$\begin{aligned} D_{k+1} &= E_\lambda \left\|(O_\lambda^{ideal})^\dagger O_\lambda^{incon}\psi_{\lambda,k}^{incon} - \psi_{\lambda,k}^{ideal}\right\|^2 \\ &= E_\lambda \left\|O_\lambda^{if}(\psi_{\lambda,k}^{incon} - \psi_{\lambda,k}^{ideal}) + (O_\lambda^{if} - I)\psi_{\lambda,k}^{ideal}\right\|^2 \end{aligned} \tag{3.2.18}$$

where

$$O_\lambda^{if} \equiv (O_\lambda^{ideal})^\dagger O_\lambda^{incon} \tag{3.2.19}$$

is an oracle recoding the difference between the ideal oracle and the inconsistent oracle. Moreover we define the unnormalized state

$$b_{\lambda,k} \equiv (O_\lambda^{if} - I)\psi_{\lambda,k}^{ideal}. \tag{3.2.20}$$

$b_{\lambda,k}$ characterize the difference between $D_k$ and $D_{k+1}$, as showed in the following. Applying $\|a+b\|^2 \le \|a\|^2 + 2\|a\|\|b\| + \|b\|^2$ with $a \equiv O_\lambda^{if}(\psi_{\lambda,k}^{incon} - \psi_{\lambda,k}^{ideal})$ and $b_{\lambda,k}$, gives

$$D_{k+1} \leq E_\lambda \left( \|\psi^{incon}_{\lambda,k} - \psi^{ideal}_{\lambda,k}\|^2 + 2\|\psi^{incon}_{\lambda,k} - \psi^{ideal}_{\lambda,k}\| \|b_{\lambda,k}\| + \|b_{\lambda,k}\|^2 \right) \tag{3.2.21}$$

Applying the Cauchy-Schwarz inequality to the second term gives

$$\begin{aligned} D_{k+1} &\leq D_k + 2\sqrt{E_\lambda \|\psi^{incon}_{\lambda,k} - \psi^{ideal}_{\lambda,k}\|^2} \sqrt{E_\lambda \|b_{\lambda,k}\|^2} + E_\lambda \|b_{\lambda,k}\|^2 \\ &= D_k + 2\sqrt{D_k}\sqrt{E_\lambda \|b_{\lambda,k}\|^2} + E_\lambda \|b_{\lambda,k}\|^2 \end{aligned} \tag{3.2.22}$$

We will show in Lemma 19 that $E_\lambda \|b_{\lambda,k}\|^2 \leq 4/2^p$; it will then follow from the inductive hypothesis $D_k \leq 4k^2/2^p$ that

$$D_{k+1} \leq 4k^2/2^p + 8k/2^p + 4/2^p = 4(k+1)^2/2^p \tag{3.2.23}$$

and we complete the induction. □

**Lemma 19** (Upper bound of $b_{\lambda,k}$).

$$E_\lambda \|b_{\lambda,k}\|^2 \leq 4/2^p, \tag{3.2.24}$$

*where $b_{\lambda,k}$ is defined in Eq. (3.2.20).*

The proof is straightforward but quite lengthy. See Section (3.4) for the proof of Lemma 19.

Now that we have proved the progress $D_k$ is trivial, the only remaining work is to carefully sum the difference over different $\lambda$ with a suitable distance measure while keeping track of the work qubits.

Since $Clone_{m+p}$ is an efficient algorithm, it cannot make more than $T = \text{poly}(m+p)$ queries to $O_\lambda$. Recalling from Lemma 18 that

$$D_T = E_\lambda \left\|\psi^{incon}_{\lambda,T} - \psi^{ideal}_{\lambda,T}\right\|^2 \leq 4T^2/2^p = O(\text{poly}(m+p)/2^p),$$

we have

$$\begin{aligned} & E_\lambda \left\|\psi_\lambda^{incon} - \psi_\lambda^{ideal}\right\|^2 = E_\lambda \left\|(\tilde{K}_\lambda^{-1} \otimes \tilde{K}_\lambda^{-1})(\psi_{\lambda,T}^{incon} - \psi_{\lambda,T}^{ideal})\right\|^2 \\ & \qquad = E_\lambda \left\|\psi_{\lambda,T}^{incon} - \psi_{\lambda,T}^{ideal}\right\|^2 = D_T \\ & \qquad = O(\mathrm{poly}(m+p)/2^p) \\ \Rightarrow \quad & E_\lambda \mathrm{Re}\langle\psi_\lambda^{incon}|\psi_\lambda^{ideal}\rangle = 1 - O(\mathrm{poly}(m+p)/2^p). \end{aligned} \tag{3.2.25}$$

Finally we change the distance measure to fidelity and wrap up the proof. For the inconsistent case, define the density matrix we obtain by tracing out the ancilla (work) qubits in the final state to be

$$\rho_\lambda^{incon} \equiv \mathrm{Tr}_{work}\left[|\psi_\lambda^{incon}\rangle\langle\psi_\lambda^{incon}|\right]. \tag{3.2.26}$$

For the ideal case, the cloner behaves perfectly and therefore

$$\mathrm{Tr}_{work}\left[|\psi_\lambda^{ideal}\rangle\langle\psi_\lambda^{ideal}|\right] = |\phi^{target}\rangle\langle\phi^{target}| \tag{3.2.27}$$

where the target state $|\phi^{target}\rangle \equiv (|0\rangle|S_1\rangle)^{\otimes 2}$. Then we obtain

$$\begin{aligned} F(E_\lambda\rho_\lambda^{incon}, |\phi^{target}\rangle\langle\phi^{target}|) & \geq E_\lambda F(\rho_\lambda^{incon}, |\phi^{target}\rangle\langle\phi^{target}|) \quad \text{(concavity of fidelity)} \\ & \geq E_\lambda \left|\langle\psi_\lambda^{incon}|\psi_\lambda^{ideal}\rangle\right| \\ & \geq E_\lambda \mathrm{Re}\langle\psi_\lambda^{incon}|\psi_\lambda^{ideal}\rangle \\ & = 1 - O(\mathrm{poly}(m+p)/2^p). \end{aligned} \tag{3.2.28}$$

where the second inequality holds since the fidelity can only increase when a subsystem (in this case, the work qubits) is traced out. Let us pick $p$ to be the same order as $m$; then it follows that the final state of Algorithm 16, $E_\lambda\rho_\lambda^{incon}$, is negligibly different from the target state, completing the proof.

□

### 3.2.3 Proof of Main Theorem

We now observe that Theorem 14 and Algorithm 16 together imply Theorem 13:

*Proof of Theorem 13.* Assume, as in the hypothesis, that there exists an algorithm to clone all component money states. Also assume that quantum-secure pseudorandom functions exist. By Algorithm 16, any component money state with an efficiently-implementable component mixer can be cloned without access to the labelling function but with access to a random function. We can then replace the random function with a QPRF and clone the subgraph states of a graph $g$

$$\left|X_g^\ell\right\rangle = \sum_{\pi \in \mathrm{Sym}_\ell} \left|\pi g\right\rangle, \; \ell = 1, \ldots, n \tag{3.2.29}$$

, since they are component money states under the component mixer $\mathrm{Sym}_\ell$. By Theorem 14, this implies that GRAPH ISOMORPHISM can be efficiently solved. □

## 3.3 Quantum money from knots [1] and generalized component money

Recall that in Section 3.2.2 we assume the existence of a black-box cloner *Clone* that clones component quantum money states using only black-box accesses to the mixer $M$ and the labelling function $L$. We then provide evidence against the existence of such a cloner, by showing that (Theorem 13) one could use the cloner to efficiently solve GRAPH ISOMORPHISM. The related scheme of quantum money from knots [1] is not component quantum money; nevertheless, we discuss in this section why Theorem 13 is relevant in that setting.

We give a simple review of quantum money from knots.[2] Given some Alexander polynomial $p$ (a knot invariant), the corresponding knot quantum money state is the

[2] We will ignore finite size corrections in this review; refer to [1] for full details.

equal superposition of all knots with Alexander polynomial $p$, i.e.

$$|\$_p\rangle = \sum_{G:f(G)=p} |G\rangle, \tag{3.3.1}$$

where $f(G)$ evaluates the Alexander polynomial of a knot $G$. Knot quantum money is very similar to a component mixer money state: we have a labelling function ($f$), and we can also implement a component mixer that instantly mixes within sets of isomorphic knots. However, if we treat knots with the same Alexander polynomial as components, the mixer only instantly mixes in some subset of the components because non-isomorphic knots might have the same Alexander polynomial.

To verify the money state $|\$_p\rangle$, one first checks that the Alexander polynomial of the constituent knots is $p$. One then uses Lemma 6 to verify that the money state is in the span of uniform superposition states over isomorphic knots. In other words, the verifier can test that the money state is of the following form:

$$\sum_{G\in C_p} a_G|S_G\rangle \tag{3.3.2}$$

where the $a_G$'s are complex complex amplitudes, $C_p$ is a set of canonical knots of the isomorphic classes with same Alexander polynomial $p$, i.e.

$$\begin{aligned} &\text{if } G \in C_p, \text{ then } f(G) = p \\ &\text{if } G_1 \in C_p \text{ and } G_2 \in C_p, \text{ then } G_1 = G_2 \text{ or } G_1 \nsim G_2, \end{aligned} \tag{3.3.3}$$

where $\sim$ indicates isomorphism. $|C_p|$ is the number of different isomorphic classes with the Alexander polynomial $p$. $|S_G\rangle$ is the equal superposition of isomorphic knots

$$|S_G\rangle \equiv \sum_{G':G'\sim G} |G'\rangle. \tag{3.3.4}$$

Since a whole subspace of states has the form (3.3.5), there are many states, aside from the genuine money state, that pass the verification. [1] gives a discussion on why (hopefully) we might expect such other states to be difficult to generate.

We now argue what it means for a cloner to clone quantum money from knots in a "black-box" manner. We can define a *generalized* component money scheme which also has a component mixer and labelling function, but each component is further partitioned into subcomponents, and the mixer only instantly inside the sub components. Quantum money from knots is an example of a generalized component money. Given a label $p$, states which pass the verifier are of the form

$$\sum_{G \in C_p} a_G |S_G\rangle, \tag{3.3.5}$$

where $C_p$ is the set of labels subcomponents of the component $p$, and $|S_G\rangle$ is the equal superposition of the subcomponent $G$.

The banknotes issued by the mint $|\$_p\rangle$ is the equal superposition of states in the component $p$, so in terms of $a_G$ and $|S_G\rangle$, it can be written in the form of

$$|\$_p\rangle = \sum_{G \in C_p} \frac{|G|}{\sqrt{\sum |G|^2}} |S_G\rangle, \tag{3.3.6}$$

where $|G|$ is the size of the subcomponent $G$.

We argue that a black-box cloner for generalized component money states, with quantum query access to a mixer $M$ and a labelling function $L$, can take the bank note as input, and output two possibly entangled states that both pass the verification. On states that pass the verifier, the cloner would act as follows:

$$|\$_p\rangle \to \sum_{G_1 \in C_p, G_2 \in C_p} A_{G_1,G_2,G} |S_{G_1}\rangle |S_{G_2}\rangle |junk_{G_1,G_2,G}\rangle, \text{ where } G \in C_p. \tag{3.3.7}$$

The cloner that we assume exists in Section 3.2.2, *Clone*, is weaker than this generalized black-box cloner, since the generalized cloner functions as *Clone* when $L$ labels the components of $M$ uniquely. Therefore if we could clone all generalized component money states, including quantum money from knots, in a black-box fashion, we can solve GRAPH ISOMORPHISM as well, by Theorem 13.

## 3.4 Proof of the upper bound on $b_{\lambda,k}$, Lemma 19

**Lemma 19.**

$$E_\lambda \|b_{\lambda,k}\|^2 \leq 4/2^p, \tag{3.4.1}$$

*where $b_{\lambda,k}$ is defined in Eq. (3.2.20)*

*Proof.* Recall that $\psi^{ideal}_{\lambda,k}$ depends on $\lambda$ in the following places in Algorithm 16: the application of $\tilde{K_\lambda}$ in step 2, and the use of the labelling function $L^{ideal}_\lambda$ and mixer $K_\lambda M^{ideal} K_\lambda^{-1}$. Using 3.2.13 and 3.2.14 we can write these functions explicitly:

$$L^{ideal}_\lambda(r,s) = \begin{cases} \star & r = \lambda(s), s \in S_1 \\ (r,s) & \text{otherwise,} \end{cases} \tag{3.4.2}$$

$$K_\lambda M^{ideal} K_\lambda^{-1}(r,s) = \begin{cases} (\lambda(s), M_a(s)) & r = \lambda(s), s \in S_1 \\ (r,s) & \text{otherwise.} \end{cases} \tag{3.4.3}$$

We see that $L^{ideal}_\lambda$ and $K_\lambda M^{ideal} K_\lambda^{-1}$ only depend on $\lambda$'s values on $S_1$ (these functions act as the identity when $s \notin S_1$). Similarly, at step 2 $\tilde{K_\lambda}$ is applied to $|0\rangle|S_1\rangle$, and therefore the values of $\lambda$ on $S - S_1$ are never used. It follows that $\psi^{ideal}_{\lambda,k}$ only depends of the values of $\lambda$ on $S_1$.

We therefore split up the average over $\lambda$ into two parts: the average over $\lambda$ when restricted to $S_1$, and the average over $\lambda$ when restricted to $S_1^c = S - S_1$. (Notationally, we will write $E_\lambda = E_{\lambda(S_1)} E_{\lambda(S_1^c)}$). This is possible because $\mathcal{F}$ is a random function, and therefore $\lambda(S_1)$ and $\lambda(S_1^C)$ are independently randomly chosen. We then have

$$\begin{aligned} E_\lambda \|b_{\lambda,k}\|^2 &= E_{\lambda(S_1)} E_{\lambda(S_1^c)} \langle \psi^{ideal}_{\lambda,k} | (O^{if\dagger}_\lambda - I)(O^{if}_\lambda - I) | \psi^{ideal}_{\lambda,k} \rangle \\ &= E_{\lambda(S_1)} \langle \psi^{ideal}_{\lambda,k} | E_{\lambda(S_1^c)} (O^{if\dagger}_\lambda - I)(O^{if}_\lambda - I) | \psi^{ideal}_{\lambda,k} \rangle \\ &\leq \| E_{\lambda(S_1^c)} (O^{if\dagger}_\lambda - I)(O^{if}_\lambda - I) \| \end{aligned} \tag{3.4.4}$$

where the matrix norm in the last line is the spectral norm.

We analyze the two cases of $O_\lambda$:

- If $O_\lambda$ represents a quantum query to $L_\lambda$, $O_\lambda^{if} = (O_\lambda^{ideal})^\dagger O_\lambda^{incon}$ is

$$O_\lambda^{if}|r,s\rangle|l\rangle = \begin{cases} |r,s\rangle|l+\star-(r,s)\rangle & r=\lambda(s), s\notin S_1 \\ |r,s\rangle|l\rangle & \text{otherwise} \end{cases} \tag{3.4.5}$$

for some encoding of $\{\star\}\cup\{(r,s)\}$ into integers. We see that $O_\lambda^{if}$ acts as the identity on most inputs, and gives a $\{\star-(r,s)\}$ label on the set $\{r=\lambda(s), s\notin S_1\}$. Therefore by direct calculation,

$$(O_\lambda^{if\dagger}-I)(O_\lambda^{if}-I)|r,s\rangle|l\rangle = \begin{cases} |r,s\rangle(2|l\rangle - |l+\star-(r,s)\rangle \\ \quad -|l-\star+(r,s)\rangle) & r=\lambda(s), s\notin S_1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.6}$$

Now taking the average over $\lambda(S_1^C)$, since each $r$ has a $1/2^p$ chance to equal $\lambda(s)$ we have that

$$E_{\lambda(S_1^C)}(O_\lambda^{if\dagger}-I)(O_\lambda^{if}-I)|r,s\rangle|l\rangle = \begin{cases} \frac{1}{2^p}|r,s\rangle(2|l\rangle - |l+\star-(r,s)\rangle \\ \quad -|l-\star+(r,s)\rangle) & s\notin S_1 \\ 0 & s\in S_1. \end{cases} \tag{3.4.7}$$

Reading off the matrix elements,

$$\begin{aligned} E_{\lambda(S_1^C)}(O_\lambda^{if\dagger}-I)(O_\lambda^{if}-I) = \frac{1}{2^p}\Bigg[\sum_r\sum_{s\notin S_1}|r,s\rangle\langle r,s| \\ \otimes\left(2I-\sum_l(|l+\star-(r,s)\rangle - |l-\star+(r,s)\rangle)\langle l|\right)\Bigg] \end{aligned} \tag{3.4.8}$$

Note that the matrix $E_{\lambda(S_1^C)}(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I)$ is block-diagonal, with each $(r, s)$ forming an independent block with spectral norm no more than $4/2^p$. Thus $\|E_{\lambda(S_1^C)}(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I)\| \leq 4/2^p$, and $E_\lambda \|b_{\lambda,k}\|^2 \leq 4/2^p$, as claimed.

- If $O_\lambda$ represents a quantum query to $K_\lambda M_a K_\lambda^{-1}$, $O_\lambda^{if} = (O_\lambda^{ideal})^\dagger O_\lambda^{incon}$ is instead

$$O_\lambda^{if}|r, s\rangle = \begin{cases} |\lambda(M_a(s)), M_a(s)\rangle & r = \lambda(s), s \notin S_1 \\ |r, s\rangle & \text{otherwise} \end{cases} \tag{3.4.9}$$

So we have

$$(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I)|r, s\rangle = \begin{cases} 2|r, s\rangle - |\lambda(M_a(s)), M_a(s)\rangle & \\ \quad -|\lambda(M_a^{-1}(s)), M_a^{-1}(s)\rangle & r = \lambda(s), s \notin S_1 \\ 0 & \text{otherwise} \end{cases}$$

$$E_{\lambda(S_1^C)}(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I)|r, s\rangle = \begin{cases} 2/2^p|r, s\rangle - \frac{1}{2^{2p}}\sum_{r'} & \\ \quad [|r', M_a(s)\rangle + |r', M_a^{-1}(s)\rangle] & s \notin S_1 \\ 0 & s \in S_1 \end{cases}$$

$$\begin{aligned} E_{\lambda(S_1^C)}(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I) &= \frac{2}{2^p}P_{S_1^c} - \frac{1}{2^{2p}}\left(\sum_{r'}\sum_{r} |r'\rangle\langle r|\right) \\ &\quad \otimes \left(\sum_{s \notin S_1} |M_a(s)\rangle\langle s| + \sum_{s \notin S_1} |M_a^{-1}(s)\rangle\langle s|\right) \\ E_\lambda \|b_{\lambda,k}\|^2 &\leq \frac{2}{2^p} + \frac{1}{2^{2p}}2^p(1+1) \\ &= \frac{4}{2^p}. \end{aligned} \tag{3.4.10}$$

When going from the first to the second equation, the first term picks up a $1/2^p$ factor from the probability of $r = \lambda(s)$ and the second term picks up a $1/2^{2p}$ factor from the probabilities of $r = \lambda(s)$ and $r' = \lambda(M_a(s))$. In the third equation we simply read off the matrix elements of $E_{\lambda(S_1^C)}(O_\lambda^{if\dagger} - I)(O_\lambda^{if} - I)$, and $P_{S_1^C}$ is the projector onto $S_1^C$.

This finishes the proof that $E_\lambda \|b_{\lambda,k}\|^2 \leq 4/2^p$ for all cases. $\square$

# Chapter 4

# Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester

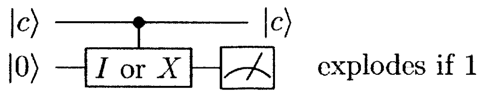
In this chapter we introduce bomb query complexity. This chapter is mostly excerpted from [52], which is joint work with Cedric Yen-Yu Lin.

## 4.1 Preliminaries

### 4.1.1 The Elitzur-Vaidman bomb testing problem

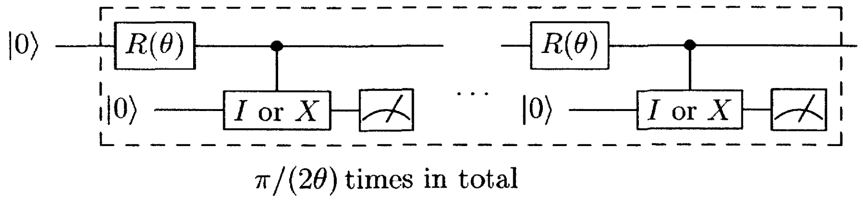
The Elitzur-Vaidman bomb testing problem [24] is a well-known thought experiment to demonstrate how quantum mechanics differs drastically from our classical perceptions. This problem demonstrates dramatically the possibility of *interaction free measurements*, the possibility of a measurement on a property of a system without disturbing the system.

The bomb-testing problem is as follows: assume we have a bomb that is either a dud or a live bomb. The only way to interact with the bomb is to probe it with a photon: if the bomb is a dud, then the photon passes through unimpeded; if the bomb is live, then the bomb explodes. We would like to determine whether the bomb is live or not without exploding it. If we pass the photon through a beamsplitter before probing the bomb, we can implement the *controlled probe*, pictured below:



(4.1.1)

The $c = 0$ corresponds to the photon traveling in a trajectory away from the bomb, and $c = 1$ corresponds the the photon probing the bomb. The controlled gate is $I$ if the bomb is a dud, and $X$ if it is live. The bomb explodes only if the photon hits and the bomb is live. It was shown in [26] how to determine whether a bomb was live with arbitrarily low probability of explosion by making use of the quantum Zeno effect [25]. Specifically, writing $R(\theta) = \exp(i\theta X)$, the unitary operator rotating $|0\rangle$ to $|1\rangle$ in $\pi/(2\theta)$ steps, the following circuit determines whether the bomb is live with failure probability $O(\theta)$:



(4.1.2)

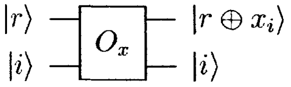
If the bomb is a dud, then the controlled probes do nothing, and repeated application of $R(\theta)$ rotates the control bit from $|0\rangle$ to $|1\rangle$. If the bomb is live, the bomb explodes with $O(\theta^2)$ probability in each application of the probe, projecting the control bit back to $|0\rangle$. After $O(1/\theta)$ iterations the control bit stays in $|0\rangle$, with only a $O(\theta)$ probability of explosion. Using $O(1/\theta)$ operations, we can thus tell a dud bomb apart from a live one with only $O(\theta)$ probability of explosion.

### 4.1.2 Quantum query complexity

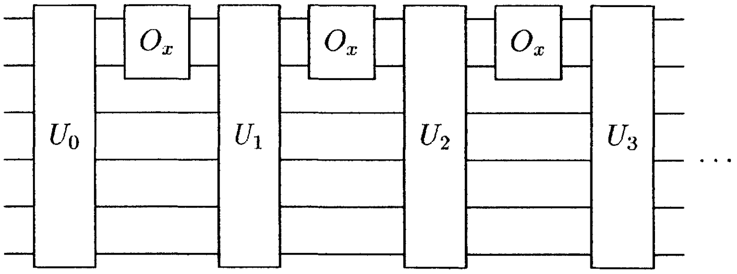
Throughout this chapter, all functions $f$ which we would like to calculate are assumed to have boolean input, i.e. the domain is $D \subseteq \{0,1\}^N$.

For a boolean strings $x \in \{0,1\}^N$, the quantum oracle $O_x$ is a unitary operator that acts on a one-qubit record register and an $N$-dimensional index register as follows ($\oplus$ is the XOR function):

$$O_x|r,i\rangle = |r \oplus x_i, i\rangle \tag{4.1.3}$$



We want to determine the value of a boolean function $f(x)$ using as few queries to the quantum oracle $O_x$ as possible. Algorithms for $f$ have the general form as the following circuit, where the $U_t$'s are unitaries independent of $x$:



The quantum query complexity $Q_\delta(f)$ is the minimum number of applications of $O_x$'s in the circuit required to determine $f(x)$ with error no more than $\delta$ for all $x$. By gap amplification (e.g. by performing the circuit multiple rounds and doing majority voting), it can be shown that the choice of $\delta$ only affects the query complexity by a $\log(1/\delta)$ factor. We therefore often set $\delta = 0.01$ and write $Q_{0.01}(f)$ as $Q(f)$.

## 4.2 Bomb query complexity

In this section we introduce a new query complexity model, which we call the *bomb query complexity*. A circuit in the bomb query model is a restricted quantum query circuit, with the following restrictions on the usage of the quantum oracle:

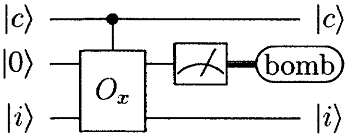
1. We have an extra control register $|c\rangle$ used to control whether $O_x$ is applied (we call the controlled version $CO_x$):

   $$CO_x|c,r,i\rangle = |c, r \oplus (c \cdot x_i), i\rangle. \tag{4.2.1}$$

   where $\cdot$ indicates boolean AND.

2. The record register, $|r\rangle$ in the definition of $CO_x$ above, *must* contain $|0\rangle$ before $CO_x$ is applied.

3. After $CO_x$ is applied, the record register is immediately measured in the computational basis (giving the answer $c \cdot x_i$), and the algorithm *terminates immediately if a 1 is measured* (if $c \cdot x_i = 1$). We refer to this as *the bomb blowing up* or *the bomb exploding.*



explodes if $c \cdot x_i = 1$ (4.2.2)

We define the *bomb query complexity* $B_{\epsilon,\delta}(f)$ to be the minimum number of times the above circuit needs to be applied in an algorithm such that the following hold for all input $x$:

- The algorithm reaches the end without the bomb exploding with probability at least $1-\epsilon$. We refer to the probability that the bomb explodes as the *probability of explosion.*

- The total probability that the bomb either explodes or fails to output $f(x)$ correctly is no more than $\delta \geq \epsilon$.

The above implies that the algorithm outputs the correct answer with probability at least $1 - \delta$.

The effect of the above circuit is equivalent to applying the following projector on $|c, i\rangle$:

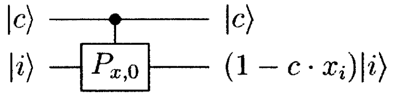
$$M_x = CP_{x,0} = \sum_{i=1}^{N} |0, i\rangle\langle 0, i| + \sum_{x_i=0} |1, i\rangle\langle 1, i| \tag{4.2.3}$$

$$= I - \sum_{x_i=1} |1, i\rangle\langle 1, i|. \tag{4.2.4}$$

$CP_{x,0}$ (which we will just call $M_x$ in our proofs later on) is the controlled version of $P_{x,0}$, the projector that projects onto the indices $i$ on which $x_i = 0$:

$$P_{x,0} = \sum_{x_i=0} |i\rangle\langle i|. \tag{4.2.5}$$

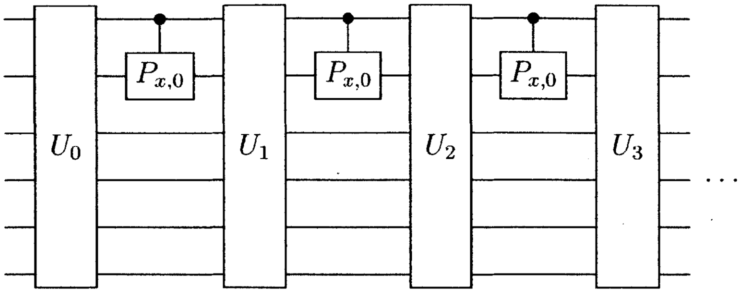
Thus Circuit 4.2.2 is equivalent to the following circuit :



(4.2.6)

In this notation, the square of the norm of a state is the probability that the state has survived to this stage, i.e. the algorithm has not terminated. The norm of $(1 - c \cdot x_i)|x_i\rangle$ is 1 if $c \cdot x_i = 0$ (the state survives this stage), and 0 otherwise (the bomb blows up).

A general circuit in this model looks like the following:



It is not at all clear that gap amplification can be done efficiently in the bomb query model to improve the error $\delta$; after all, repeating the circuit multiple times increases the chance that the bomb blows up. However, it turns out that the complexity $B_{\epsilon,\delta}(f)$ is closely related to $Q_\delta(f)$, and therefore the choice of $\delta$ affects $B_{\epsilon,\delta}(f)$ by at most a $\log^2(1/\delta)$ factor as long as $\delta \geq \epsilon$ (see Lemma 21). We therefore often omit $\delta$ by setting $\delta = 0.01$, and write $B_{\epsilon,0.01}(f)$ as $B_\epsilon(f)$. Sometimes we even omit the $\epsilon$.

Finally, note that the definition of the bomb query complexity $B(f)$ is inherently *asymmetric* with respect to 0 and 1 in the input: querying 1 causes the bomb to blow up, while querying 0 is safe. In Section 4.4.1, we define a *symmetric* bomb query model and its corresponding query complexity, $\tilde{B}_{\epsilon,\delta}(f)$. We prove that this generalized symmetric model is asymptotically equivalent to the original asymmetric model, $\tilde{B}_{\epsilon,\delta}(f) = \Theta(B_{\epsilon,\delta}(f))$, in Lemma 24. This symmetric version of the bomb query complexity will turn out to be useful in designing bomb query algorithms.

## 4.3 Main result

Our main result is the following:

**Theorem 20.** *For all functions $f$ with boolean input alphabet, and numbers $\epsilon$ satisfying* $0 < \epsilon \leq 0.01$,

$$B_{\epsilon,0.01}(f) = \Theta\left(\frac{Q_{0.01}(f)^2}{\epsilon}\right). \tag{4.3.1}$$

*Here 0.01 can be replaced by any constant no more than* $1/10$.

*Proof.* The upper bound $B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon)$ is proved in Theorem 22. The lower bound $B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon)$ is proved in Theorem 23. □

**Lemma 21.** *For all functions $f$ with boolean input alphabet, and numbers $\epsilon$, $\delta$ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

$$B_{\epsilon,0.1}(f) = O(B_{\epsilon,\delta}(f)), \quad B_{\epsilon,\delta}(f) = O(B_{\epsilon,0.1}(f) \log^2(1/\delta)). \tag{4.3.2}$$

*In particular, if $\delta$ is constant,*

$$B_{\epsilon,\delta}(f) = \Theta(B_{\epsilon,0.1}(f)). \tag{4.3.3}$$

*Proof.* This follows from Theorem 22 and the fact that $Q_{0.1}(f) = O(Q_\delta(f))$ and $Q_\delta(f) = O(Q_{0.1}(f)\log(1/\delta))$. □

Because of this result, we will often omit the 0.01 in $B_{\epsilon,0.01}$ and write simply $B_\epsilon$.

### 4.3.1 Upper bound

**Theorem 22.** *For all functions $f$ with boolean input alphabet, and numbers $\epsilon$, $\delta$ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

$$B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon). \tag{4.3.4}$$

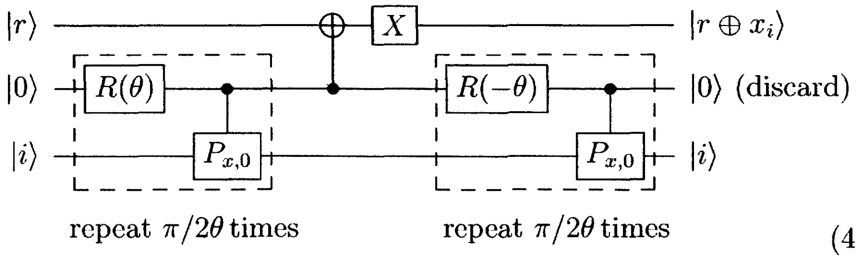
The proof follows the solution of Elitzur-Vaidman bomb-testing problem ([26], or Section 4.1.1). By taking advantage of the Quantum Zeno effect [25], using $O(\frac{Q(f)}{\epsilon})$ calls to $M_x$, we can simulate one call to $O_x$ with probability of explosion $O(\frac{\epsilon}{Q(f)})$. Replacing all $O_x$ queries with this construction results in a bounded error algorithm with probability of explosion $O(\frac{\epsilon}{Q(f)}Q(f)) = O(\epsilon)$.

*Proof.* Let $\theta = \pi/(2L)$ for some large positive integer $L$ (chosen later), and let $R(\theta)$

be the rotation

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{4.3.5}$$

We claim that with $2L$ calls to the bomb oracle $M_x = CP_{x,0}$, we can simulate $O_x$ by the following circuit with probability of explosion less than $\pi^2/(2L)$ and error $O(1/L)$.



(4.3.6)

In words, we simulate $O_x$ acting on $|r, i\rangle$ by the following steps:

1. Append an ancilla qubit $|0\rangle$, changing the state into $|r, 0, i\rangle$.

2. Repeat the following $L$ times:

   (a) apply $R(\theta)$ on the second register

   (b) apply $M_x$ on the third register controlled by the second register.

   At this point, if the bomb hasn't blown up, the second register should contain $1 - x_i$.

3. Apply $CNOT$ on the first register controlled by the second register; this copies $1 - x_i$ to the first register.

4. Apply a $NOT$ gate to the first register.

5. Repeat the following $L$ times to uncompute the second (ancilla) register :

   (a) apply $R(-\theta)$ on the second register

(b) apply $M_x$ on the third register controlled by second register

6. Discard the second (ancilla) register.

We now calculate explicitly the action of the circuit on an arbitrary state to confirm our claims above. Consider how the circuit acts on the basis state $|r, 0, i\rangle$ (the second register being the appended ancilla). We break into cases:

- If $x_i = 0$, then $P_{x,0}|i\rangle = |i\rangle$, so the controlled projections do nothing. Thus in Step 2 the rotation $R(\theta)^L = R(\pi/2)$ is applied to the ancilla qubit, rotating it from 0 to 1. After Step 2 then, the state is $|r, 1, i\rangle$. Step 3 and 4 together do not change the state, while Step 5 rotates the ancilla back to 0, resulting in the final state $|r, 0, i\rangle$.

- If $x_i = 1$, then $P_{x,0}|i\rangle = 0$, and

$$M_x|0, i\rangle = |0, i\rangle, \quad M_x|1, i\rangle = 0 \quad (\text{for } x_i = 1) \tag{4.3.7}$$

  Therefore in Step 2 and Step 5, after each rotation $R(\pm\theta)$, the projection $CP_{x,0}$ projects the ancilla back to 0:

$$M_x R(\theta)|0, i\rangle = M_x(\cos\theta|0\rangle + \sin\theta|1\rangle)|i\rangle = \cos\theta|0, i\rangle \quad (\text{for } x_i = 1) \tag{4.3.8}$$

  Each application of $M_x R(\theta)$ thus has no change on the state other than to shrink its amplitude by $\cos\theta$. The CNOT in Step 3 has no effect (since the ancilla stays in 0), and Step 4 maps $|r\rangle$ to $|r \oplus 1\rangle$. Since there are $2L$ applications of this shrinkage (in Step 2 and 5), the final state is $\cos^{2L}\theta|r \oplus 1, 0, i\rangle$.

We can now combine the two cases: by linearity, the application of the circuit on a general state $\sum_{r,i} a_{r,i}|r, i\rangle$ (removing the ancilla) is

$$\sum_{r,i} a_{r,i}|r, i\rangle \to \sum_{r\in\{0,1\}, x_i=0} a_{r,i}|r, i\rangle + \sum_{r\in\{0,1\}, x_i=1} a_{r,i}\cos^{2L}(\theta)|r \oplus 1, i\rangle \tag{4.3.9}$$

$$= \sum_{r,i} a_{r,i}\cos^{2Lx_i}\left(\frac{\pi}{2L}\right)|r \oplus x_i, i\rangle \equiv |\psi'\rangle \tag{4.3.10}$$

Thus the effect of this construction simulates the usual quantum oracle $|r,i\rangle \to |r \oplus x_i, i\rangle$ with probability of explosion no more than

$$1 - \cos^{4L}\left(\frac{\pi}{2L}\right) \leq 1 - \left(1 - \frac{\pi^2}{4L^2}\right)^{2L} \leq \frac{\pi^2}{2L}. \tag{4.3.11}$$

Moreover, the difference between the output of our circuit, $|\psi'\rangle$, and the output on the quantum oracle, $|\psi\rangle = \sum_{r,i} a_{r,i}|r \oplus x_i, i\rangle$, is

$$\||\psi'\rangle - |\psi\rangle\| = \left\| \sum_{r\in\{0,1\},x_i=1} a_{r,i}(1 - \cos^{2L}(\theta))|r \oplus 1, i\rangle \right\| \tag{4.3.12}$$

$$\leq 1 - \cos^{2L}\frac{\pi}{2L} \leq \frac{\pi^2}{4L}. \tag{4.3.13}$$

Given this construction, we can now prove our theorem. Suppose we are given a quantum algorithm that finds $f(x)$ with $Q_{\delta'}(f)$ queries, making at most $\delta' = \delta - \epsilon$ error. We construct an algorithm using bomb oracles instead by replacing each of the applications of the quantum oracle $O_x$ by our circuit construction (4.3.6), where we choose

$$L = \left\lceil \frac{\pi^2}{2\epsilon} Q_{\delta'}(f) \right\rceil \tag{4.3.14}$$

Then the probability of explosion is no more than

$$\frac{\pi^2}{2L} Q_{\delta'}(f) \leq \epsilon \tag{4.3.15}$$

and the difference between the final states, $|\psi_f\rangle$ and $|\psi'_f\rangle$, is at most

$$\||\psi'_f\rangle - |\psi_f\rangle\| \leq \frac{\pi^2}{4L} Q_{\delta'}(f) \leq \frac{\epsilon}{2}. \tag{4.3.16}$$

Therefore

$$\left|\langle\psi'_f|P|\psi'_f\rangle - \langle\psi_f|P|\psi_f\rangle\right| \le \left|\langle\psi'_f|P|\psi'_f\rangle - \langle\psi_f|P|\psi'_f\rangle\right| + \left|\langle\psi'_f|P|\psi_f\rangle - \langle\psi_f|P|\psi_f\rangle\right| \tag{4.3.17}$$

$$\le \left\||\psi'_f\rangle\right\| \left\|P\left(|\psi'_f\rangle - |\psi_f\rangle\right)\right\| + \left\|P\left(|\psi'_f\rangle - |\psi_f\rangle\right)\right\| \left\||\psi_f\rangle\right\| \tag{4.3.18}$$

$$\le \epsilon/2 + \epsilon/2 = \epsilon \tag{4.3.19}$$

for any projector $P$ (in particular, the projector that projects onto the classical answer at the end of the algorithm). The algorithm accumulates at most $\epsilon$ extra error at the end, giving a total error of no more than $\delta' + \epsilon = \delta$. This algorithm makes $2LQ_{\delta'}(f) < \frac{\pi^2}{\epsilon}Q_{\delta'}^2(f) + 2Q_{\delta'}(f)$ queries to the bomb oracle, and therefore

$$B_{\epsilon,\delta}(f) < \frac{\pi^2}{\epsilon}Q_{\delta-\epsilon}(f)^2 + 2Q_{\delta-\epsilon}(f) \tag{4.3.20}$$

$$= O\left(\frac{Q_{\delta-\epsilon}(f)^2}{\epsilon}\right). \tag{4.3.21}$$

From this we can derive that $B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon)$:

$$B_{\epsilon,\delta}(f) < B_{\epsilon/2,\delta}(f) \tag{4.3.22}$$

$$= O\left(\frac{Q_{\delta-\epsilon/2}(f)^2}{\epsilon}\right), \quad \text{by 4.3.21} \tag{4.3.23}$$

$$= O\left(\frac{Q_\delta(f)^2}{\epsilon}\right), \quad \text{since } \frac{\delta}{2} \le \delta - \frac{\epsilon}{2}. \tag{4.3.24}$$

□

### 4.3.2 Lower bound

**Theorem 23.** *For all functions $f$ with boolean input alphabet, and numbers $\epsilon$, $\delta$ satisfying $0 < \epsilon \le \delta \le 1/10$,*

$$B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon). \tag{4.3.25}$$

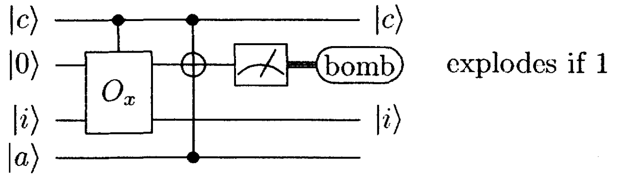
The proof of this result uses the generalized adversary bound $\mathrm{Adv}^{\pm}(f)$ [19]: we show that $B_\epsilon(f) = \Omega(\mathrm{Adv}^{\pm}(f)^2/\epsilon)$, and then use the known result that $Q(f) = O(\mathrm{Adv}^{\pm}(f))$ [22]. The complete proof is given in Appendix 4.7.1.

## 4.4 Generalizations and Applications

We now discuss applications of the result $B_\epsilon(f) = \Theta(Q(f)^2/\epsilon)$ that could be useful.

### 4.4.1 Generalizing the bomb query model

We consider modifying the bomb query model as follows. We require that the input string $x$ can only be accessed by the following circuit:



(4.4.1)

Compare with Circuit 4.2.2; the difference is that there is now an extra register $|a\rangle$, and the bomb explodes only if both $x_i = a$ and the control bit is 1. In other words, the bomb explodes if $c \cdot (x_i \oplus a) = 1$. The three registers $c$, $i$, and $a$ are allowed to be entangled, however. If we discard the second register afterwards, the effect of this circuit, written as a projector, is

$$\tilde{M}_x = \sum_{i\in[N],a\in\{0,1\}} |0,i,a\rangle\langle 0,i,a| + \sum_{i,a:x_i=a} |1,i,a\rangle\langle 1,i,a|. \tag{4.4.2}$$

Let $\tilde{B}_{\epsilon,\delta}(f)$ be the required number of queries to this modified bomb oracle $\tilde{M}_x$ to calculate $f(x)$ with error no more than $\delta$, with a probability of explosion no more than $\epsilon$. Using Theorem 20, we show that $\tilde{B}$ and $B$ are equivalent up to a constant:

**Lemma 24.** *If $f : D \to E$, where $D \subseteq \{0,1\}^N$, and $\delta \leq 1/10$ is a constant, then $B_{\epsilon,\delta}(f) = \Theta(\tilde{B}_{\epsilon,\delta}(f))$.*

*Proof.* It should be immediately obvious that $B_{\epsilon,\delta}(f) \geq \tilde{B}_{\epsilon,\delta}(f)$, since a query in the $B$ model can be simulated by a query in the $\tilde{B}$ model by simply setting $a = 0$. In the following we show that $B_{\epsilon,\delta}(f) = O(\tilde{B}_{\epsilon,\delta}(f))$.

For each string $x \in \{0,1\}^N$, define the string $\tilde{x} \in \{0,1\}^{2N}$ by concatenating two copies of $x$ and flipping every bit of the second copy. In other words,

$$\tilde{x}_i = \begin{cases} x_i & \text{if } i \leq N \\ 1 - x_{i-N} & \text{if } i > N \end{cases}. \tag{4.4.3}$$

Let $\tilde{D} = \{\tilde{x} : x \in D\}$. Given a function $f : D \to \{0,1\}$, define $\tilde{f} : \tilde{D} \to \{0,1\}$ by $\tilde{f}(\tilde{x}) = f(x)$.

We claim that a $\tilde{B}$ query to $x$ can be simulated by a $B$ query to $\tilde{x}$. This can be seen by comparing $\tilde{M}_x$:

$$\tilde{M}_x = \sum_{i\in[N],a} |0,i,a\rangle\langle 0,i,a| + \sum_{i\in[N],a:x_i=a} |1,i,a\rangle\langle 1,i,a|. \tag{4.4.4}$$

and $M_{\tilde{x}}$:

$$M_{\tilde{x}} = \sum_{\tilde{i}\in[2N]} |0,\tilde{i}\rangle\langle 0,\tilde{i}| + \sum_{\tilde{i}\in[2N]:\tilde{x}_{\tilde{i}}=0} |1,\tilde{i}\rangle\langle 1,\tilde{i}|. \tag{4.4.5}$$

Recalling the definition of $\tilde{x}$ in 4.4.3, we see that these two projectors are exactly equal if we encode $\tilde{i}$ as $(i,a)$, where $i \equiv \tilde{i} \mod N$ and $a = \lfloor i/N \rfloor$.

Since $\tilde{f}(\tilde{x}) = f(x)$, we thus have $\tilde{B}_{\epsilon,\delta}(f) = B_{\epsilon,\delta}(\tilde{f})$. Our result then readily follows; it can easily be checked that $Q(f) = Q(\tilde{f})$, and therefore by Theorem 20,

$$\begin{aligned} \tilde{B}_{\epsilon,\delta}(f) = B_{\epsilon,\delta}(\tilde{f}) &= \Theta\left(\frac{Q(\tilde{f})^2}{\epsilon}\right) \\ &= \Theta\left(\frac{Q(f)^2}{\epsilon}\right) \end{aligned} \tag{4.4.6}$$

□

There are some advantages to allowing the projector $\tilde{M}_x$ instead of $M_x$. First of all, the inputs 0 and 1 in $x$ are finally manifestly symmetric, unlike that in $M_x$ (the bomb originally blew up if $x_i = 1$, but not if $x_i = 0$). Moreover, we now allow the algorithm to *guess* an answer to the query (this answer may be entangled with the index register $i$), and the bomb blows up only if the guess is wrong, controlled on $c$. This flexibility may allow more leeway in designing algorithms for the bomb query model, as we soon utilize.

### 4.4.2 Using classical algorithms to design bomb query algorithms

We now demonstrate the possibility that we can prove *nonconstructive* upper bounds on $Q(f)$ for some functions $f$, by creating bomb query algorithms and using that $Q(f) = \Theta(\sqrt{\epsilon B_\epsilon(f)})$. Consider for example the following classical algorithm for the OR function:

**Algorithm 25** (Classical algorithm for OR)**.** Pick some arbitrary ordering of the $N$ bits, and query them one by one, terminating as soon as a 1 is seen. Return 1 if a 1 was queried; otherwise return 0.

We can convert this immediately to a bomb query algorithm for OR, by using the construction in the proof of Theorem 22. That construction allows us to implement the operation $O_x$ in $O(\epsilon^{-1})$ queries, with $O(\epsilon)$ error and probability of explosion if $x_i = 1$ (but no error if $x_i = 0$). Thus we have the following:

**Algorithm 26** (Bomb algorithm for OR)**.** Query the $N$ bits one-by-one, and apply the construction of Theorem 22 one bit at a time, using $O(1/\epsilon)$ operations each time. Terminate as soon as a 1 is seen, and return 1; otherwise return 0 if all bits are 0.

Since the algorithm ends as soon as a 1 is found, the algorithm only accumulates $\epsilon$ error in total. Thus this shows $B_\epsilon(OR) = O(N/\epsilon)$.

Note, however, that we have already shown that $Q(f) = \Theta(\sqrt{\epsilon B_\epsilon(f)})$ for boolean $f$. An $O(N/\epsilon)$ bomb query algorithm for OR therefore implies that $Q(OR) = O(\sqrt{N})$.

We have showed the existence of an $O(\sqrt{N})$ quantum algorithm for the OR function, without actually constructing one!

We formalize the intuition in the above argument by the following theorem:

**Theorem 27.** *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Suppose there is a classical randomized query algorithm $\mathcal{A}$, that makes at most $T$ queries, and evaluates $f$ with bounded error. Let the query results of $\mathcal{A}$ on random seed $s_{\mathcal{A}}$ be $x_{p_1}, x_{p_2}, \cdots, x_{p_{\tilde{T}(x)}}$, $\tilde{T}(x) \leq T$, where $x$ is the hidden query string.*

*Suppose there is another (not necessarily time-efficient) randomized algorithm $\mathcal{G}$, with random seed $s_{\mathcal{G}}$, which takes as input $x_{p_1}, \cdots, x_{p_{t-1}}$ and $s_{\mathcal{A}}$, and outputs a guess for the next query result $x_{p_t}$ of $\mathcal{A}$. Assume that $\mathcal{G}$ makes no more than an expected total of $G$ mistakes (for all inputs $x$). In other words,*

$$\mathbf{E}_{s_{\mathcal{A}},s_{\mathcal{G}}} \left\{ \sum_{t=1}^{\tilde{T}(x)} \left| \mathcal{G}(x_{p_1}, \cdots, x_{p_{t-1}}, s_{\mathcal{A}}, s_{\mathcal{G}}) - x_{p_t} \right| \right\} \leq G \quad \forall x. \tag{4.4.7}$$

*Note that $\mathcal{G}$ is given the random seed $s_{\mathcal{A}}$ of $\mathcal{A}$, so it can predict the next query index of $\mathcal{A}$.*
*Then $B_\epsilon(f) = O(TG/\epsilon)$, and thus (by Theorem 20) $Q(f) = O(\sqrt{TG})$.*

As an example, in our simple classical example for OR we have $T = N$ (the algorithm takes at most $N$ steps) and $G = 1$ (the guessing algorithm always guesses the next query to be 0; since the algorithm terminates on a 1, it makes at most one mistake).

*Proof of theorem 27.* We generalize the argument in the OR case. We take the classical algorithm and replace each classical query by the construction of Theorem 22, using $O(G/\epsilon)$ bomb queries each time. On each query, the bomb has a $O(\epsilon/G)$ chance of exploding when the guess is wrong, and no chance of exploding when the guess is correct. Therefore the total probability of explosion is $O(\epsilon/G) \cdot G = O(\epsilon)$. The total number of bomb queries used is $O(TG/\epsilon)$.

For the full technical proof, see Appendix 4.7.2. □

### 4.4.3 Explicit quantum algorithm for Theorem 27

In this section we give an explicit quantum algorithm, in the setting of Theorem 27, that reproduces the given query complexity. This algorithm is very similar to the one given by R. Kothari for the oracle identification problem [40].

**Theorem 28.** *Under the assumptions of Theorem 27, there is an explicit quantum algorithm for $f$ with query complexity $O(\sqrt{TG})$.*

*Proof.* We will construct this algorithm (Algorithm 30) shortly. We need the following quantum search algorithm as a subroutine:

**Theorem 29** (Finding the first marked element in a list). *Suppose there is an ordered list of $N$ elements, and each element is either marked or unmarked. Then there is a bounded-error quantum algorithm for finding the* ***first*** *marked element in the list (or determines that no marked elements exist), such that:*

- *If the first marked element is the $d$-th element of the list, then the algorithm uses an expected $O(\sqrt{d})$ time and queries.*

- *If there are no marked elements, then the algorithm uses $O(\sqrt{N})$ time and queries, but always determines correctly that no marked elements exist.*

This algorithm is straightforward to derive given the result in [53], and was already used in Kothari's algorithm [40]. We give the algorithm (Algorithm 47) and its analysis in Appendix 4.7.3.

We now give our explicit quantum algorithm.

**Algorithm 30** (Simulating a classical query algorithm by a quantum one).

*Input.* Classical randomized algorithm $\mathcal{A}$ that computes $f$ with bounded error. Classical randomized algorithm $\mathcal{G}$ that guesses queries of $\mathcal{A}$. Oracle $O_x$ for the hidden string $x$.

*Output.* $f(x)$ with bounded error.

The quantum algorithm proceeds by attempting to produce the list of queries and results that $\mathcal{A}$ would have made. More precisely, given a randomly chosen random seed $s_{\mathcal{A}}$, the quantum algorithm outputs (with constant error) a list of pairs $(p_1(x), x_{p_1(x)}), \cdots, (p_{\tilde{T}(x)}(x), x_{p_{\tilde{T}(x)}(x)})$. This list is such that on random seed $s_{\mathcal{A}}$, the $i$-th query algorithm of $\mathcal{A}$ is made at the position $p_i(x)$, and the query result is $x_{p_i(x)}$. The quantum algorithm then determines the output of $\mathcal{A}$ using this list.

The main idea for the algorithm is this: we first assume that the guesses made by $\mathcal{G}$ are correct. By repeatedly feeding the output of $\mathcal{G}$ back into $\mathcal{A}$ and $\mathcal{G}$, we can obtain a list of query values for $\mathcal{A}$ without any queries to the actual black box. We then search for the first deviation of the string $x$ from the predictions of $\mathcal{G}$; assuming the first deviation is the $d_1$-th query, by Theorem 29 the search takes $O(\sqrt{d_1})$ queries (ignoring error for now). We then know that all the guesses made by $\mathcal{G}$ are correct up to the $(d_1 - 1)$-th query, and incorrect for the $d_1$-th query.

With the corrected result of the first $d_1$ queries, we now continue by assuming again the guesses made by $\mathcal{G}$ are correct starting from the $(d_1 + 1)$-th query, and search for the location of the next deviation, $d_2$. This takes $O(\sqrt{d_2 - d_1})$ queries; we then know that all the guesses made by $\mathcal{G}$ are correct from the $(d_1 + 1)$-th to $(d_2 - 1)$-th query, and incorrect for the $d_2$-th one. Continuing in this manner, we eventually determine all query results of $\mathcal{A}$ after an expected $G$ iterations.

We proceed to spell out our algorithm. For the time being, we assume that algorithm for Theorem 29 has no error and thus requires no error reduction.

1. Initialize random seeds $s_{\mathcal{A}}$ and $s_{\mathcal{G}}$ for $\mathcal{A}$ and $\mathcal{G}$. We will simulate the behavior of $\mathcal{A}$ and $\mathcal{G}$ on these random seeds. Initialize $d = 0$. $d$ is such that we have determined the values of all query results of $\mathcal{A}$ up to the $d$-th query. Also initialize an empty list $\mathcal{L}$ of query pairs.

2. Repeat until either all query results of $\mathcal{A}$ are determined, or $100G$ iterations of this loop have been executed:

   (a) Assuming that $\mathcal{G}$ always guesses correctly starting from the $(d + 1)$-th query, compute from $\mathcal{A}$ and $\mathcal{G}$ a list of query positions $p_{d+1}, p_{d+2}, \cdots$ and results $\tilde{a}_{d+1}, \tilde{a}_{d+2}, \cdots$. This requires no queries to the black box.

(b) Using our algorithm for finding the first marked element (Theorem 29, Algorithm 47), find the first index $d^* > d$ such that the actual query result of $\mathcal{A}$ differs from the guess by $\mathcal{G}$, i.e. $x_{p_d} \neq \tilde{a}_d$; or return that no such $d^*$ exists. This takes $O(\sqrt{d^* - d})$ time in the former case, and $O(\sqrt{T - d})$ time in the latter.

(c) We break into cases:

i. If an index $d^*$ was found in Step 2b, then the algorithm decides the next mistake made by $\mathcal{G}$ is at position $d^*$. It thus adds the query pairs $(p_{d+1}, \tilde{a}_{d+1}), \cdots, (p_{d^*-1}, \tilde{a}_{d^*-1})$, and the pair $(p_{d^*}, 1 - \tilde{a}_{d^*})$, to the list $\mathcal{L}$. Also set $d = d^*$.

ii. If no index $d^*$ was found in Step 2b, the algorithm decides that all remaining guesses by $\mathcal{G}$ are correct. Thus the query pairs $(p_{d+1}, \tilde{a}_{d+1})$, $\cdots$, $(p_{\tilde{T}(x)}, \tilde{a}_{\tilde{T}(x)})$ are added to $\mathcal{L}$, where $\tilde{T}(x) \leq T$ is the number of queries made by $\mathcal{A}$.

3. If the algorithm found all query results of $\mathcal{A}$ in $100G$ iterations of step 2, use $\mathcal{L}$ to calculate the output of $\mathcal{A}$; otherwise the algorithm fails.

We now count the total number of queries. Suppose $g \leq 100G$ is the number of iterations of Step 2; if all query results have been determined, $g$ is the number of wrong guesses by $\mathcal{G}$. Say the list of $d$'s found is $d_0 = 0, d_1, \cdots, d_g$. Let $d_{g+1} = T$. Step 2 is executed for $g + 1$ times, and the total number of queries is

$$O\left(\sum_{i=1}^{g+1} \sqrt{d_i - d_{i-1}}\right) = O\left(\sqrt{Tg}\right) = O\left(\sqrt{TG}\right) \tag{4.4.8}$$

by the Cauchy-Schwarz inequality.

We now analyze the error in our algorithm. The first source of error is cutting off the loop in Step 2: by Markov's inequality, for at least 99% of random seeds $s_{\mathcal{G}}, s_{\mathcal{G}}$, $\mathcal{G}$ makes no more than $100G$ wrong guesses. For these random seeds all query results of $\mathcal{A}$ are determined. Cutting off the loop thus gives at most 0.01 error.

The other source of error is the error of Algorithm 47 used in Step 2b: we had assumed that it could be treated as zero-error, but we now remove this assumption. Assuming each iteration gives error $\delta'$, the total error accrued could be up to $O(g\delta')$. It seems as if we would need to set $\delta' = O(1/G)$ for the total error to be constant, and thus gain an extra logarithmic factor in the query complexity.

However, in his paper for oracle identification [40], Kothari showed that multiple calls to Algorithm 47 can be composed to obtain a bounded-error algorithm based on span programs without an extra logarithmic factor in the query complexity; refer to [40, Section 3] for details. Therefore we can replace the iterations of Step 2 with Kothari's span program construction and get a bounded error algorithm with complexity $O(\sqrt{TG})$.

□

Note that while Algorithm 30 has query complexity $O(\sqrt{TG})$, the time complexity may be much higher. After all, Algorithm 30 proceeds by simulating $\mathcal{A}$ query-by-query, although the number of actual queries to the oracle is smaller. Whether or not we can get a algorithm faster than $\mathcal{A}$ using this approach may depend on the problem at hand.

## 4.5 Improved upper bounds on quantum query complexity

We now use Theorem 28 to improve the quantum query complexity of certain graph problems.

### 4.5.1 Single source shortest paths for unweighted graphs

**Problem 31** (Single source shortest paths (SSSP) for unweighted graphs). The adjacency matrix of a directed graph $n$-vertex graph $G$ is provided as a black box; a query on the pair $(v, w)$ returns 1 if there is an edge from $v$ to $w$, and 0 otherwise. We are given a fixed vertex $v_{start}$. Call the length of a shortest path from $v_{start}$ to another vertex $w$ the *distance* $d_w$ of $w$ from $v_{start}$; if no path exists, define $d_w = \infty$. Our task is to find $d_w$ for all vertices $w$ in $G$.

In this section we shall show the following:

**Theorem 32.** *The quantum query complexity of single-source shortest paths in an unweighted graph is* $\Theta(n^{3/2})$ *in the adjacency matrix model.*

*Proof.* The lower bound of $\Omega(n^{3/2})$ is known [54]. We show the upper bound by applying Theorem 28 to a classical algorithm. The following well-known classical algorithm (commonly known as *breadth first search*, BFS) solves this problem:

**Algorithm 33** (Classical algorithm for unweighted SSSP).

1. Initialize $d_w := \infty$ for all vertices $w \neq v_{start}$, $d_{v_{start}} := 0$, and $\mathcal{L} := (v_{start})$. $\mathcal{L}$ is the ordered list of vertices for which we have determined the distances, but whose outgoing edges we have not queried.

2. Repeat until $\mathcal{L}$ is empty:

    - Let $v$ be the first (in order of time added to $\mathcal{L}$) vertex in $\mathcal{L}$. For all vertices $w$ such that $d_w = \infty$:

        - Query $(v, w)$.
        - If $(v, w)$ is an edge, set $d_w := d_v + 1$ and add $w$ to the end of $\mathcal{L}$.

    - Remove $v$ from $\mathcal{L}$.

We omit the proof of correctness of this algorithm (see for example [55]). This algorithm uses up to $T = O(n^2)$ queries. If the guessing algorithm always guesses that $(v, w)$ is not an edge, then it makes at most $G = n - 1$ mistakes; hence $Q(f) = O(\sqrt{TG}) = O(n^{3/2})$.[1]

□

[1] It seems difficult to use our method to give a corresponding result for the adjacency list model; after all, the result of a query is much harder to guess when the input alphabet is non-boolean.

The previous best known quantum algorithm for unweighted SSSP, to our best knowledge, was given by Furrow [27]; it has query complexity $O(n^{3/2}\sqrt{\log n})$.

We now consider the quantum query complexity of unweighted $k$-source shortest paths (finding $k$ shortest-path trees rooted from $k$ beginning vertices). If we apply Algorithm 33 on $k$ different starting vertices, then the expected number of wrong guesses is no more than $G = k(n-1)$; however, the total number of edges we query need not exceed $T = O(n^2)$, since an edge never needs to be queried more than once. Therefore

**Corollary 34.** *The quantum query complexity of unweighted $k$-source shortest paths in the adjacency matrix model is $O(k^{1/2}n^{3/2})$, where $n$ is the number of vertices.*

We use this idea – that $T$ need not exceed $O(n^2)$ when dealing with graph problems – again in the following section.

### 4.5.2 Maximum bipartite matching

**Problem 35** (Maximum bipartite matching). We are given as black box the adjacency matrix of an $n$-vertex bipartite graph $G = (V = X \cup Y, E)$, where the undirected set of edges $E$ only run between the bipartite components $X$ and $Y$. A *matching* of $G$ is a list of edges of $G$ that do not share vertices. Our task is to find a maximum matching of $G$, i.e. a matching that contains the largest possible number of edges.

In this section we show that

**Theorem 36.** *The quantum query complexity of maximum bipartite matching is $O(n^{7/4})$ in the adjacency matrix model, where $n$ is the number of vertices.*

*Proof.* Once again we apply Theorem 28 to a classical algorithm. Classically, this problem is solved in $O(n^{5/2})$ time by the Hopcroft-Karp [56] algorithm (here $n = |V|$). We summarize the algorithm as follows (this summary roughly follows that of [28]):

**Algorithm 37** (Hopcroft-Karp algorithm for maximum bipartite matching [56]).

1. Initialize an empty matching $\mathcal{M}$. $\mathcal{M}$ is a matching that will be updated until it is maximum.

2. Repeat the following steps until $\mathcal{M}$ is a maximum matching:

   (a) Define the *directed* graph $H = (V', E')$ as follows:

$$\begin{aligned} V' &= X \cup Y \cup \{s, t\} \\ E' &= \{(s, x) \mid x \in X, (x, y) \notin \mathcal{M} \text{ for all } y \in Y\} \\ &\quad \cup \{(x, y) \mid x \in X, y \in Y, (x, y) \in E, (x, y) \notin \mathcal{M}\} \\ &\quad \cup \{(y, x) \mid x \in X, y \in Y, (x, y) \in E, (x, y) \in \mathcal{M}\} \\ &\quad \cup \{(y, t) \mid y \in Y, (x, y) \notin \mathcal{M} \text{ for all } x \in X\} \qquad (4.5.1) \end{aligned}$$

   where $s$ and $t$ are two extra auxilliary vertices. Note that if $(s, x_1, y_1, x_2, y_2, \cdots, x_\ell, y_\ell, t)$ is a path in $H$ from $s$ to $t$, then $x_i \in X$ and $y_i \in Y$ for all $i$. Additionally, the edges (aside from the first and last) alternate from being in $\mathcal{M}$ and not being in $\mathcal{M}$: $(x_i, y_i) \notin \mathcal{M}$, $(y_i, x_{i+1}) \in \mathcal{M}$. Such a path is called an *augmenting path* in the literature.

   We note that a query to the adjacency matrix of $E'$ can be simulated by a query to the adjacency matrix of $E$.

   (b) Using the breadth-first search algorithm (Algorithm 33), in the graph $H$, find the length of the shortest path, or distance, of all vertices from $s$. Let the distance from $s$ to $t$ be $2\ell + 1$.

   (c) Find a maximal set $S$ of vertex-disjoint shortest paths from $s$ to $t$ in the graph $H$. In other words, $S$ should be a list of paths from $s$ to $t$ such that each path has length $2\ell + 1$, and no pair of paths share vertices except for $s$ and $t$. Moreover, all other shortest paths from $s$ to $t$ share at least one vertex (except for $s$ and $t$) with a path in $S$. We describe how to find such a maximal set in Algorithm 38.

   (d) If $S$ is empty, the matching $M$ is a maximum matching, and we terminate. Otherwise continue:

   (e) Let $(s, x_1, y_1, x_2, y_2, \cdots, x_\ell, y_\ell, t)$ be a path in $S$. Remove the $\ell - 1$ edges $(x_{i+1}, y_i)$ from $\mathcal{M}$, and insert the $\ell$ edges $(x_i, y_i)$ into $\mathcal{M}$. This increases

$|\mathcal{M}|$ by 1. Repeat for all paths in $S$; there are no conflicts since the paths in $S$ are vertex-disjoint.

Once again, we omit the proof of correctness of this algorithm; the correctness is guaranteed by Berge's Lemma [57], which states that a matching is maximum if there are no more augmenting paths for the matching. Moreover, $O(\sqrt{n})$ iterations of Step 2 suffice [56].

We now describe how to find a maximal set of shortest-length augmenting paths in Step 2(c). This algorithm is essentially a modified version of depth-first search:

**Algorithm 38** (Finding a maximal set of vertex-disjoint shortest-length augmenting paths).

*Input.* The directed graph $H$ defined in Algorithm 37, as well as the distances $d_v$ of all vertices $v$ from $s$ (calculated in Step 2(b) of Algorithm 37).

1. Initialize a set of paths $S := \emptyset$, set of vertices $R := \{s\}$, and a stack[2] of vertices $\mathcal{L} := (s)$. $\mathcal{L}$ contains the ordered list of vertices that we have begun, but not yet finished, processing. $R$ is the set of vertices that we have processed. $S$ is the set of vertex-disjoint shortest-length augmenting paths that we have found.

2. Repeat until $\mathcal{L}$ is empty:

    (a) If the vertex in the front of $\mathcal{L}$ is $t$, we have found a new vertex-disjoint path from $s$ to $t$:

    - Trace the path from $t$ back to $s$ by removing elements from the front of $\mathcal{L}$ until $s$ is at the front. Add the corresponding path to $S$.
    - Start again from the beginning of Step 2.

    (b) Let $v$ be the vertex in the front of $\mathcal{L}$ (i.e. the vertex *last* added to, and still in, $\mathcal{L}$). Recall the distance from $s$ to $v$ is $d_v$.

[2] A stack is a data structure such that elements that are first inserted into the stack are removed last.

(c) Find $w$ such that $w \notin R$, $d_w = d_v + 1$, and $(v, w)$ (as an edge in $H$) has not been queried in this algorithm. If no such vertex $w$ exists, remove $v$ from $\mathcal{L}$ and start from the beginning of Step 2.

(d) Query $(v, w)$ on the graph $H$.

(e) If $(v, w)$ is an edge, add $w$ to the *front* of $\mathcal{L}$. If $w \neq t$, add $w$ to $R$.

3. Output $S$, the maximal set of vertex-disjoint shortest-length augmenting paths.

We now return to Algorithm 37 and count $T$ and $G$. There is obviously no need to query the same edge more than once, so $T = O(n^2)$. If the algorithm always guesses, on a query $(v, w)$, that there is no edge between $(v, w)$, then it makes at most $G = O(n^{3/2})$ mistakes: in Step 2(b) there are at most $O(n)$ mistakes (see Algorithm 33), while in Step 2(c)/Algorithm 38 there is at most one queried edge leading to each vertex aside from $t$, and edges leading to $t$ can be computed without queries to the adjacency matrix of $H$. Since Step 2 is executed $O(\sqrt{n})$ times, our counting follows.

Thus there is a quantum query algorithm with complexity $Q = O(\sqrt{TG}) = O(n^{7/4})$.

□

To our knowledge, this is the first known nontrivial upper bound on the query complexity of maximum bipartite matching.[3] The time complexity of this problem was studied by Ambainis and Spalek in [28]; they gave an upper bound of $O(n^2 \log n)$ time in the adjacency matrix model. A lower bound of $\Omega(n^{3/2})$ for the query complexity of this problem was given in [58, 59].

For readers familiar with network flow, the arguments in this section also apply to Dinic's algorithm for maximum flow [60] on graphs with unit capacity, i.e. where the capacity of each edge is 0 or 1. On graphs with unit capacity, Dinic's algorithm is essentially the same as Hopcroft-Karp's, except that augmenting paths are over a general, nonbipartite flow network. (The set $S$ in Step 2(c) of Algorithm 37 is

[3] The trivial upper bound is $O(n^2)$, where all pairs of vertices are queried.

generally referred to as a *blocking flow* in this context.) It can be shown that only $O(\min\{m^{1/2}, n^{2/3}\})$ iterations of Step 2 are required [61, 62], where $m$ is the number of edges of the graph. Thus $T = O(n^2)$, $G = O(\min\{m^{1/2}, n^{2/3}\}n)$, and therefore

**Theorem 39.** *The quantum query complexity of the maximum flow problem in graphs with unit capacity is $O(\min\{n^{3/2}m^{1/4}, n^{11/6}\})$, where $n$ and $m$ are the number of vertices and edges in the graph, respectively.*

It is an open question whether a similar result for maximum matching in a general nonbipartite graph can be proven, perhaps by applying Theorem 28 to the classical algorithm of Micali and Vazirani [63].

## 4.6 Projective query complexity

We end this chapter with a brief discussion on another query complexity model, which we will call the *projective query complexity*. This model is similar to the bomb query model in that the only way of accessing $x_i$ is through a classical measurement; however, in the projective query model the algorithm does not terminate if a 1 is measured. Our motivation for considering the projective query model is that its power is intermediate between the classical and quantum query models. To the best of our knowledge, this model was first considered in 2002 in unpublished results by S. Aaronson [41].

A circuit in the projective query complexity model is a restricted quantum query circuit, with the following restrictions on the use of the quantum oracle:

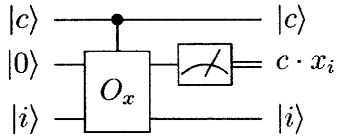
1. We have an extra control register $|c\rangle$ used to control whether $O_x$ is applied (we call the controlled version $CO_x$):

$$CO_x|c, r, i\rangle = |c, r \oplus (c \cdot x_i), i\rangle. \tag{4.6.1}$$

   where $\cdot$ indicates boolean AND.

2. The record register, $|r\rangle$ in the definition of $CO_x$ above, *must* contain $|0\rangle$ before $CO_x$ is applied.

3. After $CO_x$ is applied, the record register is immediately measured in the computational basis, giving the answer $c \cdot x_i$. The result, a classical bit, can then be used to control further quantum unitaries (although only controlling the next unitary is enough, since the classical bit can be stored).



(4.6.2)

We wish to evaluate a function $f(x)$ with as few calls to this *projective oracle* as possible. Let the number of oracle calls required to evaluate $f(x)$, with at most $\delta$ error, be $P_\delta(f)$. By gap amplification, the choice of $\delta$ only affects $P_\delta(f)$ by a factor of $\log(1/\delta)$, and thus we will often omit $\delta$.

We can compare the definition in this section with the definition of the bomb query complexity in Section 4.2: the only difference is that if $c \cdot x_i = 1$, the algorithm terminates in the bomb model, while the algorithm can continue in the projective model. Therefore the following is evident:

**Observation 40.** *$P_\delta(f) \le B_{\epsilon,\delta}(f)$, and therefore $P(f) = O(Q(f)^2)$.*

Moreover, it is clear that the projective query model has power intermediate between classical and quantum (a controlled query in the usual quantum query model can be simulated by appending a 0 to the input string), and therefore letting $R_\delta(f)$ be the classical randomized query complexity,

**Observation 41.** *$Q_\delta(f) \le P_\delta(f) \le R_\delta(f)$.*

For explicit bounds on $P$, Regev and Schiff [30] have shown that for computing the OR function, the projective query complexity loses the Grover speedup:

**Theorem 42** ([30]). *$P(OR) = \Omega(N)$.*

Note that this result says nothing about $P(AND)$, since the definition of $P(f)$ is asymmetric with respect to 0 and 1 in the input.[4]

We observe that there could be a separation in both parts of the inequality $Q \leq P \leq B$:

$$Q(OR) = \Theta(\sqrt{N}), \quad P(OR) = \Theta(N), \quad B(OR) = \Theta(N) \tag{4.6.3}$$

$$Q(PARITY) = \Theta(N), \quad P(PARITY) = \Theta(N), \quad B(PARITY) = \Theta(N^2) \tag{4.6.4}$$

In the latter equation we used the fact that $Q(PARITY) = \Theta(N)$ [17]. It therefore seems difficult to adapt our lower bound method in Section 4.3.2 to $P(f)$.

It would be interesting to find a general lower bound for $P(f)$, or to establish more clearly the relationship between $Q(f)$, $P(f)$, and $R(f)$.

## 4.7 Proofs

### 4.7.1 Proof of the adversary lower bound for $B(f)$ (Theorem 23)

Before we give the proof of the general result that $B(f) = \Omega(Q(f)^2)$ (Theorem 23)), we will illustrate the proof by means of an example, the special case where $f$ is the AND function.

**Theorem 43.** *For $\delta < 1/10$, $B_{\epsilon,\delta}(AND) = \Omega(\frac{N}{\epsilon})$.*

*Proof.* Let $|\psi_t^0\rangle$ be the unnormalized state of the algorithm with $x = 1^n$, and $|\psi_t^k\rangle$ be the unnormalized state with $x = 1 \cdots 101 \cdots 1$, $x_k = 0$, right before the $(t+1)$-th call to $M_x$. Then

$$|\psi_{t+1}^x\rangle = U_{t+1} M_x |\psi_t^x\rangle \tag{4.7.1}$$

[4] We could have defined a symmetric version of $P$, say $\tilde{P}$, by allowing an extra guess on the measurement result, similar to our construction of $\tilde{B}$ in Section 4.4.1. Unfortunately, Regev and Schiff's result, Theorem 42, do not apply to this case, and we see no obvious equivalence between $P$ and $\tilde{P}$.

for some unitary $U_{t+1}$. For ease of notation, we'll write $M_0 \equiv M_{1^n}$ and $M_k = M_{1\cdots 101\cdots 1}$, where the $k$-th bit is 0 in the latter case. When acting on the control and index bits,

$$M_0 = \sum_{i=1}^{N} |0,i\rangle\langle 0,i|$$
$$M_k = \sum_{i=1}^{N} |0,i\rangle\langle 0,i| + |1,k\rangle\langle 1,k|. \tag{4.7.2}$$

Since the $M_i$'s are projectors, $M_i^2 = M_i$. Define

$$\epsilon_t^i = \langle\psi_t^i|(I - M_i)|\psi_t^i\rangle, \quad i = 0, 1, \cdots, N. \tag{4.7.3}$$

Note that $\langle\psi_{t+1}^i|\psi_{t+1}^i\rangle = \langle\psi_t^i|M_i^2|\psi_t^i\rangle = \langle\psi_t^i|M_i|\psi_t^i\rangle = \langle\psi_t^i|\psi_t^i\rangle - \epsilon_t^i$, for all $i = 0, \cdots, N$ (including 0!), and hence

$$\sum_{t=0}^{T-1} \epsilon_t^i = \langle\psi_0^i|\psi_0^i\rangle - \langle\psi_T^i|\psi_T^i\rangle \leq \epsilon. \tag{4.7.4}$$

We now define the progress function. Let

$$W_t^k = \langle\psi_t^0|\psi_t^k\rangle \tag{4.7.5}$$

and let the progress function be a sum over $W^k$'s:

$$W_t = \sum_{k=1}^{N} W_t^k = \sum_{k=1}^{N} \langle\psi_t^0|\psi_t^k\rangle. \tag{4.7.6}$$

We can lower bound the total change in the progress function by (see [18] for a proof; their proof equally applies to unnormalized states)

$$W_0 - W_T \geq (1 - 2\sqrt{\delta(1-\delta)})N. \tag{4.7.7}$$

We now proceed to upper bound $W_0 - W_T$. Note that

$$\begin{aligned} W_t^k - W_{t+1}^k &= \langle \psi_t^0 | \psi_t^k \rangle - \langle \psi_t^0 | M_0 M_k | \psi_t^k \rangle \\ &= \langle \psi_t^0 | (I - M_0) M_k | \psi_t^k \rangle + \langle \psi_t^0 | M_0 (I - M_k) | \psi_t^k \rangle \\ &\quad + \langle \psi_t^0 | (I - M_0)(I - M_k) | \psi_t^k \rangle \end{aligned} \tag{4.7.8}$$

and since $M_0(I - M_k) = 0$, $(I - M_0)M_k = |1, k\rangle\langle 1, k|$, we have

$$\begin{aligned} W_t^k - W_{t+1}^k &\leq \langle \psi_t^0 | 1, k\rangle\langle 1, k | \psi_t^k \rangle + \left\| (I - M_0) | \psi_t^0 \rangle \right\| \left\| (I - M_k) | \psi_t^k \rangle \right\| \\ &\leq \left\| \langle 1, k | \psi_t^0 \rangle \right\| + \sqrt{\epsilon_t^0 \epsilon_t^k}. \end{aligned} \tag{4.7.9}$$

where we used 4.7.3. Summing over $k$ and t, we obtain

$$\begin{aligned} W_0 - W_T &\leq \sum_{t=0}^{T-1} \sum_{k=1}^{N} \left[ \left\| \langle 1, k | \psi_t^0 \rangle \right\| + \sqrt{\epsilon_t^0 \epsilon_t^k} \right] \\ &\leq \sqrt{TN} \sqrt{\sum_{t=0}^{T-1} \sum_{k=1}^{N} \langle \psi_t^0 | 1, k \rangle \langle 1, k | \psi_t^0 \rangle} + \sum_{t=0}^{T-1} \sum_{k=1}^{N} \frac{\epsilon_t^0 + \epsilon_t^k}{2} \\ &\leq \sqrt{TN} \sqrt{\sum_{t=0}^{T-1} \langle \psi_t^0 | (I - M_0) | \psi_t^0 \rangle} + N\epsilon \\ &\leq \sqrt{TN \sum_{t=0}^{T-1} \epsilon_t^0} + N\epsilon \\ &\leq \sqrt{\epsilon TN} + N\epsilon \end{aligned} \tag{4.7.10}$$

where in the second line we used Cauchy-Schwarz and the AM-GM inequality. Combined with $W_0 - W_T \geq (1 - 2\sqrt{\delta(1-\delta)})N$ (Eq. 4.7.7), this immediately gives us

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - \epsilon)^2 N}{\epsilon}. \tag{4.7.11}$$

□

We now proceed to prove the general result. This proof follows the presentation given in A. Childs's online lecture notes [64], which we found quite illuminating.

**Theorem 23.** *For all functions $f$ with boolean input alphabet, and numbers $\epsilon$, $\delta$ satisfying* $0 < \epsilon \leq \delta \leq 1/10$,

$$B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon). \tag{4.7.12}$$

*Proof.* We prove the lower bound on $B_{\epsilon,\delta}$ by showing that it is lower bounded by $\Omega(\mathrm{Adv}^{\pm}(f)^2/\epsilon)$, where $\mathrm{Adv}^{\pm}(f)$ is the generalized (i.e. allowing negative weights) adversary bound [19] for $f$. We can then derive our theorem from the result [22] that $Q(f) = O(\mathrm{Adv}^{\pm}(f))$.

We generalize the bound on the $f = AND$ case to an adversary bound for $B_{\epsilon,\delta}$ on arbitrary $f$. Define the projectors

$$\begin{aligned} \Pi_0 &= \sum_{i=1}^{N} |0,i\rangle\langle 0,i| \\ \Pi_i &= |1,i\rangle\langle 1,i|, \quad i = 1,\cdots,n. \end{aligned} \tag{4.7.13}$$

It is clear that

$$\Pi_0 + \sum_{i=1}^{N} \Pi_i = I. \tag{4.7.14}$$

Note that $M_x = CP_{x,0}$ is

$$M_x = \Pi_0 + \sum_{i:x_i=0} \Pi_i. \tag{4.7.15}$$

Define $|\psi_t^x\rangle$ as the state of the algorithm right before the $(t+1)$-th query with input $x$; then

$$|\psi_{t+1}^x\rangle = U_{t+1} M_x |\psi_t^x\rangle \tag{4.7.16}$$

for some unitary $U_{t+1}$. Now if we let

$$\epsilon_t^x = \langle\psi_t^x|(I - M_x)|\psi_t^x\rangle \tag{4.7.17}$$

then it follows that $\langle\psi_t^x|\psi_t^x\rangle - \langle\psi_{t+1}^x|\psi_{t+1}^x\rangle = \epsilon_t^x$, and thus

$$\sum_{t=0}^{T-1} \epsilon_t^x = \langle\psi_0^x|\psi_0^x\rangle - \langle\psi_T^x|\psi_T^x\rangle \leq \epsilon. \tag{4.7.18}$$

We proceed to define the progress function. Let $S$ be the set of allowable input strings $x$. Let $\Gamma$ be an *adversary matrix*, i.e. an $S \times S$ matrix such that

1. $\Gamma_{xy} = \Gamma_{yx} \quad \forall x, y \in S$; and

2. $\Gamma_{xy} = 0 \quad$ if $f(x) = f(y)$.

Let $a$ be the normalized eigenvector of $\Gamma$ with eigenvalue $\pm\|\Gamma\|$, where $\pm\|\Gamma\|$ is the largest (by absolute value) eigenvalue of $\Gamma$. Define the progress function

$$W_t = \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle\psi_t^x|\psi_t^y\rangle. \tag{4.7.19}$$

For $\epsilon \leq \delta < 1/10$ we have that[5] (see [19] for a proof; their proof applies equally well to unnormalized states)

$$|W_0 - W_T| \geq (1 - 2\sqrt{\delta(1-\delta)} - 2\delta)\|\Gamma\| \tag{4.7.20}$$

[5] As described in [19], the $2\delta$ term can be removed if the output is boolean (0 or 1).

We now proceed to upper bound $|W_0 - W_T| \leq \sum_t |W_t - W_{t-1}|$. Note that

$$\begin{aligned}
W_t - W_{t+1} &= \sum_{x,y \in S} \Gamma_{xy} a_x^* a_y \left( \langle \psi_t^x | \psi_t^y \rangle - \langle \psi_{t+1}^x | \psi_{t+1}^y \rangle \right) \\
&= \sum_{x,y \in S} \Gamma_{xy} a_x^* a_y \left( \langle \psi_t^x | \psi_t^y \rangle - \langle \psi_t^x | M_x M_y | \psi_t^y \rangle \right) \\
&= \sum_{x,y \in S} \Gamma_{xy} a_x^* a_y \left( \langle \psi_t^x | (I - M_x) M_y | \psi_t^y \rangle + \langle \psi_t^x | M_x (I - M_y) | \psi_t^y \rangle \right. \\
&\quad \left. + \langle \psi_t^x | (I - M_x)(I - M_y) | \psi_t^y \rangle \right) \qquad (4.7.21)
\end{aligned}$$

We bound the three terms separately. For the first two terms, use

$$\begin{aligned}
(I - M_x) M_y &= \sum_{i : x_i = 1, y_i = 0} \Pi_i \\
&= (I - M_x) \sum_{i : x_i \neq y_i} \Pi_i \qquad (4.7.22)
\end{aligned}$$

Define the $S \times S$ matrix $\Gamma_i$ as

$$\Gamma_i = \begin{cases} \Gamma_{xy} & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases} \qquad (4.7.23)$$

The first term of 4.7.21 is

$$\begin{aligned}
\sum_{x,y \in S} \sum_{i : x_i \neq y_i} \Gamma_{xy} a_x^* a_y \langle \psi_t^x | (I - M_x) \Pi_i | \psi_t^y \rangle &= \sum_{x,y \in S} \sum_{i=1}^{N} (\Gamma_i)_{xy} \, a_x^* a_y \langle \psi_t^x | (I - M_x) \Pi_i | \psi_t^y \rangle \\
&= \sum_{i=1}^{N} \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \qquad (4.7.24)
\end{aligned}$$

where

$$\begin{aligned}
Q_i &= \sum_{x \in S} a_x \Pi_i | \psi_t^x \rangle \langle x | \qquad (4.7.25) \\
\tilde{Q}_i &= \sum_{x \in S} a_x \Pi_i (I - M_x) | \psi_t^x \rangle \langle x | . \qquad (4.7.26)
\end{aligned}$$

Although both $Q_i$ and $\tilde{Q}_i$ depend on $t$, we suppress the $t$ dependence in the notation. Similarly, the second term of 4.7.21 is equal to $\sum_{i=1}^N \operatorname{tr}(\tilde{Q}_i \Gamma_i Q_i^\dagger)$. We can also rewrite the third term of 4.7.21 as

$$\sum_{x,y \in S} \Gamma_{xy} a_x^* a_y \langle \psi_t^x | (I - M_x)(I - M_y) | \psi_t^y \rangle = \operatorname{tr}(Q' \Gamma Q'^\dagger) \tag{4.7.27}$$

where

$$Q' = \sum_{x \in S} a_x (I - M_x) | \psi_t^x \rangle \langle x |. \tag{4.7.28}$$

Therefore, adding absolute values,

$$|W_t - W_{t+1}| \leq \sum_{i=1}^N \left[ \left| \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \right| + \left| \operatorname{tr}(\tilde{Q}_i \Gamma_i Q_i^\dagger)) \right| \right] + \left| \operatorname{tr}(Q' \Gamma Q'^\dagger) \right| \tag{4.7.29}$$

To continue, we need the following lemma:

**Lemma 44.** *For any $m, n > 0$ and matrices $X \in \mathbb{C}^{m \times n}$, $Y \in \mathbb{C}^{n \times n}$, $Z \in \mathbb{C}^{n \times m}$, we have $|\operatorname{tr}(XYZ)| \leq \|X\|_F \|Y\| \|Z\|_F$. Here $\|\cdot\|$ and $\|\cdot\|_F$ denote the spectral norm and Frobenius norm, respectively.*

This lemma can be proved by using that $|\operatorname{tr}(XYZ)| \leq \|Y\| \|ZX\|_{tr}$ and $\|ZX\|_{tr} \leq \|X\|_F \|Z\|_F$, which follows from [65, Exercise IV.2.12 and Corollary IV.2.6]. A more accessible proof is found online at [64].

Then by Lemma 44,

$$\sum_{i=1}^N \left| \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \right| \leq \sum_{i=1}^N \|\Gamma_i\| \|Q_i\|_F \|\tilde{Q}_i\|_F \tag{4.7.30}$$

Since

$$\begin{aligned}\sum_{i=1}^{N} \|Q_i\|_F^2 &= \sum_{i=1}^{N} \sum_{x \in S} |a_x|^2 \, \|\Pi_i |\psi_t^x\rangle\|^2 \\ &= \sum_{x \in S} |a_x|^2 \langle \psi_t^x | \sum_{i=1}^{N} \Pi_i | \psi_t^x \rangle \\ &\leq \sum_{x \in S} |a_x|^2 \\ &= 1 \end{aligned} \tag{4.7.31}$$

and

$$\begin{aligned}\sum_{i=1}^{N} \|\tilde{Q}_i\|_F^2 &= \sum_{i=1}^{N} \sum_{x \in S} |a_x|^2 \, \|\Pi_i (I - M_x) |\psi_t^x\rangle\|^2 \\ &= \sum_{x \in S} |a_x|^2 \langle \psi_t^x | (I - M_x) \left( \sum_{i=1}^{N} \Pi_i \right) (I - M_x) | \psi_t^x \rangle \\ &\leq \sum_{x \in S} |a_x|^2 \langle \psi_t^x | (I - M_x) | \psi_t^x \rangle \\ &= \sum_{x \in S} |a_x|^2 \epsilon_t^x \end{aligned} \tag{4.7.32}$$

we have, by Cauchy-Schwarz,

$$\sum_{i=1}^{N} \|Q_i\|_F \|\tilde{Q}_i\|_F \leq \sqrt{\sum_{x \in S} |a_x|^2 \epsilon_t^x} \tag{4.7.33}$$

Therefore by 4.7.30 and 4.7.33,

$$\sum_{i=1}^{N} \left| \mathrm{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \right| \leq \sqrt{\sum_{x \in S} |a_x|^2 \epsilon_t^x} \max_{i \in [N]} \|\Gamma_i\|. \tag{4.7.34}$$

Similartly for $\mathrm{tr}(Q'\Gamma Q'^{\dagger})$, we have

$$\begin{aligned}\|Q'\|_F^2 &= \sum_{x\in S} |a_x|^2 \, \|(I - M_x)|\psi_t^x\rangle\|^2 \\ &= \sum_{x\in S} |a_x|^2 \langle\psi_t^x|(I - M_x)|\psi_t^x\rangle \\ &= \sum_{x\in S} |a_x|^2 \epsilon_t^x \qquad (4.7.35)\end{aligned}$$

and using Lemma 44,

$$\begin{aligned}\mathrm{tr}(Q'\Gamma Q'^{\dagger}) &\leq \|Q'\|_F^2 \|\Gamma\| \qquad (4.7.36) \\ &= \sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \qquad (4.7.37)\end{aligned}$$

Thus continuing from 4.7.29, we have that

$$|W_t - W_{t+1}| \leq 2\sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x \max_{i\in[N]} \|\Gamma_i\|} + \sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \qquad (4.7.38)$$

Finally, if we sum the above over $t$ we obtain

$$|W_0 - W_T| \leq 2 \max_{i\in[N]} \|\Gamma_i\| \sum_{t=0}^{T-1} \sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} + \sum_{t=0}^{T-1} \sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \qquad (4.7.39)$$

The first term can be bounded using Cauchy-Schwarz:

$$\begin{aligned}\sum_{t=0}^{T-1} \sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} &\leq \sqrt{T \sum_{t=0}^{T-1} \sum_{x\in S} |a_x|^2 \epsilon_t^x} \\ &\leq \sqrt{\epsilon T} \qquad (4.7.40)\end{aligned}$$

where we used $\sum_t \epsilon_t^x \leq \epsilon$ and $\sum_x |a_x|^2 = 1$. The second term can be summed easily:

$$\begin{aligned}\sum_{t=0}^{T-1}\sum_{x\in S}|a_x|^2\epsilon_t^x\|\Gamma\| &\leq \sum_{x\in S}|a_x|^2\epsilon\|\Gamma\| \\ &= \epsilon\|\Gamma\|. \end{aligned} \tag{4.7.41}$$

Therefore

$$|W_0 - W_T| \leq 2\sqrt{\epsilon T}\max_{i\in[N]}\|\Gamma_i\| + \epsilon\|\Gamma\|. \tag{4.7.42}$$

Combined with our lower bound $|W_0 - W_T| \geq (1 - 2\sqrt{\delta(1-\delta)} - 2\delta)\|\Gamma\|$, this immediately gives

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - 2\delta - \epsilon)^2}{4\epsilon}\frac{\|\Gamma\|^2}{\max_{i\in[N]}\|\Gamma_i\|^2}. \tag{4.7.43}$$

Recalling that [19]

$$\mathrm{Adv}^{\pm}(f) = \max_{\Gamma}\frac{\|\Gamma\|}{\max_{i\in[N]}\|\Gamma_i\|}, \tag{4.7.44}$$

we obtain[6]

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - 2\delta - \epsilon)^2}{4\epsilon}\mathrm{Adv}^{\pm}(f)^2. \tag{4.7.45}$$

We now use the tight characterization of the quantum query complexity by the general weight adversary bound:

**Theorem 45** ([22, Theorem 1.1]). *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Then $Q_{0.01}(f) = O(\mathrm{Adv}^{\pm}(f))$.*

Combined with our result above, we obtain

$$B_{\epsilon,\delta}(f) = \Omega\left(\frac{Q_{0.01}(f)^2}{\epsilon}\right). \tag{4.7.46}$$

[6] For boolean output (0 or 1) the $2\delta$ term can be dropped, as we previously noted (Footnote 5).

□

### 4.7.2 Proof of Theorem 27

We restate and prove Theorem 27:

**Theorem 27.** *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Suppose there is a classical randomized query algorithm $\mathcal{A}$, that makes at most $T$ queries, and evaluates $f$ with bounded error. Let the query results of $\mathcal{A}$ on random seed $s_{\mathcal{A}}$ be $x_{p_1}, x_{p_2}, \cdots, x_{p_{\tilde{T}(x)}}$, $\tilde{T}(x) \leq T$, where $x$ is the hidden query string.*

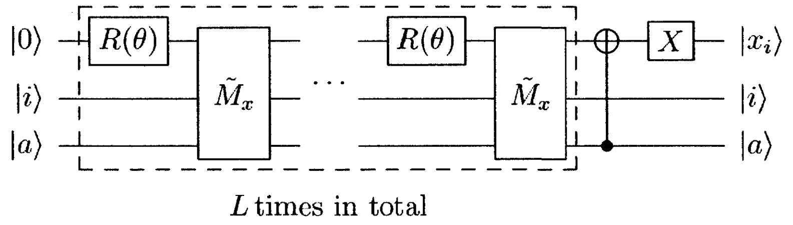
*Suppose there is another (not necessarily time-efficient) randomized algorithm $\mathcal{G}$, with random seed $s_{\mathcal{G}}$, which takes as input $x_{p_1}, \cdots, x_{p_{t-1}}$ and $s_{\mathcal{A}}$, and outputs a guess for the next query result $x_{p_t}$ of $\mathcal{A}$. Assume that $\mathcal{G}$ makes no more than an expected total of $G$ mistakes (for all inputs $x$). In other words,*

$$\mathbf{E}_{s_{\mathcal{A}}, s_{\mathcal{G}}} \left\{ \sum_{t=1}^{\tilde{T}(x)} \left| \mathcal{G}(x_{p_1}, \cdots, x_{p_{t-1}}, s_{\mathcal{A}}, s_{\mathcal{G}}) - x_{p_t} \right| \right\} \leq G \quad \forall x. \tag{4.7.47}$$

*Note that $\mathcal{G}$ is given the random seed $s_{\mathcal{A}}$ of $\mathcal{A}$, so it can predict the next query index of $\mathcal{A}$.*
*Then $B_\epsilon(f) = O(TG/\epsilon)$, and thus (by Theorem 20) $Q(f) = O(\sqrt{TG})$.*

*Proof.* For the purposes of this proof, we use the characterization of $B$ by the modified bomb construction given in section 4.4.1. This proof is substantially similar to that of theorem 22.

The following circuit finds $x_i$ with zero probability of explosion if $x_i = a$, and with an $O(1/L)$ probability of explosion if $x_i \neq a$ (in both cases the value of $x_i$ found by the circuit is always correct):



(4.7.48)

where $\theta = \pi/(2L)$ for some large number $L$ to be picked later, and

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \\ \sin\theta & \cos\theta \end{pmatrix} \tag{4.7.49}$$

The boxed part of the circuit is then simply $[\tilde{M}_x(R(\theta)\otimes I \otimes I)]^L$, applied to the state $|0, i, a\rangle$. We can analyze this circuit by breaking into cases:

- If $x_i = a$, then $\tilde{M}_x|\psi\rangle|i,a\rangle = |\psi\rangle|i,a\rangle$ for any state $|\psi\rangle$ in the control register. Thus the $\tilde{M}_x$'s act as identities, and the circuit simply applies the rotation $R(\theta)^L = R(\pi/2)$ to the control register, rotating it from 0 to 1. We thus obtain the state $|1, i, a\rangle$; the final CNOT and X gates add $a \oplus 1 = x_i \oplus 1$ to the first register, giving $|x_i, i, a\rangle$.

- If $x_i \neq a$, then

$$\tilde{M}_x|0,i,a\rangle = |0,i,a\rangle, \quad \tilde{M}_x|1,i,a\rangle = 0 \quad (\text{for } x_i \neq a) \tag{4.7.50}$$

Therefore after each rotation $R(\theta)$, the projection $\tilde{M}_x$ projects the control qubit back to 0:

$$\tilde{M}_x(R(\theta)\otimes I\otimes I)|0,i,a\rangle = \tilde{M}_x(\cos\theta|0\rangle + \sin\theta|1\rangle)|i,a\rangle = \cos\theta|0,i,a\rangle \ (\text{for } x_i \neq a) \tag{4.7.51}$$

In this case the effect of $\tilde{M}_x(R(\theta)\otimes I\otimes I)$ is to shrink the amplitude by $\cos(\theta)$;

$L$ applications results in the state $\cos^L(\theta)|0, i, a\rangle$. The final CNOT and X gates add $a \oplus 1 = x_i$ to the first register, giving $|x_i, i, a\rangle$.

The probability of explosion is 0 if $x_i = a$. If $x_i \neq a$, the probability of explosion is

$$1 - \cos^{2L}\left(\frac{\pi}{2L}\right) \leq \frac{\pi^2}{4L}. \tag{4.7.52}$$

Pick

$$L = \left\lceil \frac{\pi^2 G}{4\epsilon} \right\rceil. \tag{4.7.53}$$

Then the probability of explosion is 0 if $x_i = a$, and no more than $\epsilon/G$ if $x_i \neq a$. If the bomb does not explode, then the circuit *always* finds the correct value of $x_i$.

We now construct the bomb query algorithm based on $\mathcal{A}$ and $\mathcal{G}$. The bomb query algorithm follows $\mathcal{A}$, with each classical query replaced by the above construction. There are no more than $TL \approx \pi^2 TG/(4\epsilon)$ bomb queries. At each classical query, we pick the guess $a$ to be the guess provided by $\mathcal{G}$. The bomb only has a chance of exploding if the guess is incorrect; hence for all $x$, the total probability of explosion is no more than

$$\frac{\epsilon}{G} \mathbf{E}_{s_\mathcal{A}, s_\mathcal{G}} \left\{ \sum_{t=1}^{\tilde{T}(x)} \left| \mathcal{G}(x_{p_1}, \cdots, x_{p_{t-1}}, s_\mathcal{A}, s_\mathcal{G}) - x_{p_t} \right| \right\} \leq \epsilon \tag{4.7.54}$$

Thus replacing the classical queries of $\mathcal{A}$ with our construction gives a bomb query algorithm with probability of explosion no more than $\epsilon$; aside from the probability of explosion, this bomb algorithm makes no extra error over the classical algorithm $\mathcal{A}$. The number of queries this algorithm uses is

$$\tilde{B}_{\epsilon,\delta+\epsilon}(f) \leq \left\lceil \frac{\pi^2 G}{4\epsilon} \right\rceil T, \tag{4.7.55}$$

where $\delta$ is the error rate of the classical algorithm. Therefore by Lemma 24 and Lemma 21,

$$B_\epsilon(f) = O(B_{\epsilon,\delta+\epsilon}(f)) = O(\tilde{B}_{\epsilon,\delta+\epsilon}(f)) = O\left(TG/\epsilon\right) \tag{4.7.56}$$

□

### 4.7.3 Proof of Theorem 29

We restate and prove Theorem 29:

**Theorem 29** (Finding the first marked element in a list). *Suppose there is an ordered list of $N$ elements, and each element is either marked or unmarked. Then there is a bounded-error quantum algorithm for finding the* ***first*** *marked element in the list, or determines that no marked elements exist, such that:*

- *If the first marked element is the $d$-th element of the list, then the algorithm uses an expected $O(\sqrt{d})$ time and queries.*
- *If there are no marked elements, then the algorithm uses $O(\sqrt{N})$ time and queries.*

*Proof.* We give an algorithm that has the stated properties. We first recall a quantum algorithm for finding the minimum in a list of items:

**Theorem 46** ([53]). *Given a function $g$ on a domain of $N$ elements, there is a quantum algorithm that finds the minimum of $g$ with expected $O(\sqrt{N})$ time and evaluations of $g$, making $\delta < 1/10$ error.*

We now give our algorithm for finding the first marked element in a list. For simplicity, assume that $N$ is a power of 2 (i.e. $\log_2 N$ is an integer).

**Algorithm 47.**

1. For $\ell = 2^0, 2^1, 2^2, \cdots, 2^{\log_2 N} = N$:
    - Find the first marked element within the first $\ell$ elements, or determine no marked element exists. This can be done by defining

$$g(i) = \begin{cases} \infty & \text{if } i \text{ is unmarked} \\ i & \text{if } i \text{ is marked,} \end{cases} \tag{4.7.57}$$

and using Theorem 46 to find the minimum of $g$. This takes $O(\sqrt{\ell}) = O(\sqrt{d})$ queries and makes $\delta < 1/10$ error for each $\ell$. If a marked element $i^*$ is found, the algorithm outputs $i^*$ and stops.

2. If no marked element was found in Step 1, the algorithm decides that no marked element exists.

We now claim that Algorithm 47 has the desired properties. Let us break into cases:

- If no marked items exist, then no marked item can possibly be found in Step 1, so the algorithm correctly determines that no marked items exist in Step 2. The number of queries used is

$$\sum_{i=0}^{\log_2 N} \sqrt{2^i} = O(\sqrt{N}) \tag{4.7.58}$$

as desired.

- Suppose the first marked item is the $d$-th item in the list. Then in Step 1(a), if $\ell \geq d$, there is at least a $1 - \delta$ probability that the algorithm will detect that a marked item exists in the first $\ell$ elements and stop the loop. Letting $\alpha = \lceil \log_2 d \rceil$, the total expected number of queries is thus

$$\sum_{i=0}^{\alpha-1} \sqrt{2^i} + \sum_{i=\alpha}^{\log_2 N} \delta^{i-\alpha}\sqrt{2^i} + O(\sqrt{d}) \leq \frac{2^{\alpha/2} - 1}{\sqrt{2} - 1} + \sqrt{2^\alpha}\frac{1}{1 - \sqrt{2}\delta} + O(\sqrt{d}) \tag{4.7.59}$$

$$= O(\sqrt{2^\alpha}) + O(\sqrt{d}) \tag{4.7.60}$$

$$= O(\sqrt{d}). \tag{4.7.61}$$

The probability of not finding the marked item at the first $\ell \geq d$ is at most $\delta$ , and thus the total error of the algorithm is bounded by $\delta$.

□

# Bibliography

[1] D. G. Edward Farhi, A. Hassidim, A. Lutomirski, and P. Shor, "Quantum money from knots," arXiv:1004.5127 [quant-ph].

[2] P. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing* **26** no. 5, (1997) 1484–1509, arXiv:quant-ph/9508027.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC).* May, 1996. arXiv:quant-ph/9605043.

[4] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," arXiv:quant-ph/0001106.

[5] S. Wiesner, "Conjugate coding," *ACM SIGACT News* **15** no. 1, (1983) .

[6] A. Brodutch, D. Nagaj, O. Sattath, and D. Unruh, "An adaptive attack on Wiesner's quantum money," arXiv:1404.1507 [quant-ph].

[7] A. Lutomirski, "An online attack against wiesner's quantum money," arXiv:1010.0256 [quant-ph].

[8] A. Lutomirski, "Component mixers and a hardness result for counterfeiting quantum money," 2011. arXiv:1107.0321 [quant-ph].

[9] S. Aaronson and P. Christiano, "Quantum money from hidden subspaces," arXiv:1203.4740 [quant-ph].

[10] A. Lutomirski, S. Aaronson, E. Farhi, D. Gosset, A. Hassidim, J. Kelner, and P. Shor, "Breaking and making quantum money: toward a new quantum cryptographic protocol," 2009. arXiv:0912.3825 [quant-ph].

[11] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM Journal on Computing* **26** no. 5, (1997) 1510–1523, arXiv:quant-ph/9701001.

[12] A. Ambainis, "Quantum walk algorithm for element distinctness," *SIAM Journal on Computing* **37** no. 1, (2007) 210–239, arXiv:quant-ph/0311001.

[13] M. Szegedy, "Quantum speed-up of Markov chain based algorithms," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2004.

[14] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," *SIAM Journal on Computing* **40** no. 1, (2011) 142–164, arXiv:quant-ph/0608026.

[15] S. Jeffery, R. Kothari, and F. Magniez, "Nested quantum walks with quantum data structures," in *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1474–1485. 2013. arXiv:1210.1199 [quant-ph].

[16] A. Belovs, "Span programs for functions with constant-sized 1-certificates," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 77–84. 2012. arXiv:1105.4024 [quant-ph].

[17] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," in *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 352. 1998. arXiv:quant-ph/9802049.

[18] A. Ambainis, "Quantum lower bounds by quantum arguments," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 636–643. 2000. arXiv:quant-ph/0002066.

[19] P. Høyer, T. Lee, and R. Špalek, "Negative weights make adversaries stronger," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 526–535. 2007. arXiv:quant-ph/0611054.

[20] B. W. Reichardt, "Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function," in *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 544–551. 2009. arXiv:0904.2759 [quant-ph].

[21] B. W. Reichardt, "Reflections for quantum query algorithms," in *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 560–569. 2011. arXiv:1005.1601 [quant-ph].

[22] T. Lee, R. Mittal, B. W. Reichardt, R. Špalek, and M. Szegedy, "Quantum query complexity of state conversion," in *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 344–353. 2011. arXiv:1011.3020 [quant-ph].

[23] S. Kimmel, "Quantum adversary (upper) bound," *Chicago Journal of Theoretical Computer Science* no. 4, (2013) , arXiv:1101.0797 [quant-ph].

[24] A. C. Elitzur and L. Vaidman, "Quantum mechanical interaction-free measurements," *Foundations of Physics* **23** no. 7, (July, 1993) 987–997, arXiv:hep-th/9305002.

[25] B. Misra and E. C. G. Sudarshan, "The Zeno's paradox in quantum theory," *Journal of Mathematical Physics* **18** no. 4, (1977) 756.

[26] P. Kwiat, H. Weinfurter, T. Herzog, A. Zeilinger, and M. A. Kasevich, "Interaction-free measurement," *Physical Review Letters* **74** no. 24, (1995) 4763.

[27] B. Furrow, "A panoply of quantum algorithms," *Quantum Information and Computation* **8** no. 8, (September, 2008) 834–859, arXiv:quant-ph/0606127.

[28] A. Ambainis and R. Špalek, "Quantum algorithms for matching and network flows," in *Lecture Notes in Computer Science*, vol. 3884, pp. 172–183. Springer, 2006. arXiv:quant-ph/0508205.

[29] S. Dörn, "Quantum algorithms for matching problems," *Theory of Computing Systems* **45** no. 3, (October, 2009) 613–628.

[30] O. Regev and L. Schiff, "Impossibility of a quantum speed-up with a faulty oracle," in *Lecture Notes in Computer Science*, vol. 5125, pp. 773–781. Springer, 2008. arXiv:1202.1027 [quant-ph].

[31] G. Mitchison and R. Jozsa, "Counterfactual computation," *Proceedings of the Royal Society A* **457** no. 2009, (2001) 1175–1194, arXiv:quant-ph/9907007.

[32] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters, and P. G. Kwiat, "Counterfactual quantum computation through quantum interrogation," *Nature* **439** (February, 2006) 949–952.

[33] G. Mitchison and R. Jozsa, "The limits of counterfactual computation," arXiv:quant-ph/0606092.

[34] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters, and P. Kwiat, "Counterfactual computation revisited," arXiv:quant-ph/0607101.

[35] L. Vaidman, "The impossibility of the counterfactual computation for all possible outcomes," arXiv:quant-ph/0610174.

[36] O. Hosten and P. G. Kwiat, "Weak measurements and counterfactual computation," arXiv:quant-ph/0612159.

[37] H. Salih, Z.-H. Li, M. Al-Amri, and M. S. Zubairy, "Protocol for direct counterfactual quantum communication," *Physical Review Letters* **110** (2013) 170502, arXiv:1206.2042 [quant-ph].

[38] L. Vaidman, "Comment on "protocol for direct counterfactual quantum communication" [arxiv:1206.2042]," arXiv:1304.6689 [quant-ph].

[39] T.-G. Noh, "Counterfactual quantum cryptography," *Physical Review Letters* **103** (2009) 230501, arXiv:0809.3979 [quant-ph].

[40] R. Kothari, "An optimal quantum algorithm for the oracle identification problem," in *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, E. W. Mayr and N. Portier, eds., vol. 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 482–493. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2014. arXiv:1311.7685 [quant-ph].

[41] S. Aaronson. Personal communication, 2014.

[42] E. Crosson, E. Farhi, C. Y.-Y. Lin, H.-H. Lin, and P. Shor, "Different strategies for optimization using the quantum adiabatic algorithm,".

[43] D. Nagaj, R. D. Somma, and M. Kieferova, "Quantum speedup by quantum annealing," *Phys. Rev. Lett.* **109** no. 5, (2012) 050501, arXiv:1202.6257 [quant-ph].

[44] T. Caneva, T. Calarco, R. Fazio, G. E. Santoro, and S. Montangero, "Speeding up critical system dynamics through optimized evolution," *Phys. Rev. A* **84** no. 1, (2011) 012312, arXiv:1011.6634 [cond-mat.other].

[45] E. Farhi, J. Goldstone, and S. Gutmann, "Quantum adiabatic evolution algorithms with different paths," arXiv:quant-ph/0208135.

[46] S. Bravyi, D. P. DiVincenzo, R. I. Oliveira, and B. M. Terhal, "The complexity of stoquastic local hamiltonian problems," *Quant. Inf. Comp.* **8** no. 5, (2008) 0361–0385, arXiv:quant-ph/0606140.

[47] S. Bravyi, A. J. Bessen, and B. M. Terhal, "Merlin-arthur games and stoquastic complexity," arXiv:quant-ph/0611021.

[48] A. W. Sandvik, "Computational studies of quantum spin systems," *AIP Conf. Proc.* **1297** (2010) 135, arXiv:1101.3281 [cond-mat.str-el].

[49] E. Farhi, D. Gosset, I. Hen, A. W. Sandvik, P. Shor, A. P. Young, and F. Zamponi, "Performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs," *Phys. Rev. A* **86** no. 5, (2012) 052334, arXiv:1208.3757 [quant-ph].

[50] M. Zhandry, "How to construct quantum random functions," *FOCS* (2012) 679–687.

[51] M. A. Neilsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[52] C. Y.-Y. Lin and H.-H. Lin, "Upper bounds on quantum query complexity inspired by the elitzur-vaidman bomb tester," arXiv:1410.0932v2 [quant-ph].

[53] C. Dürr and P. Høyer, "A quantum algorithm for finding the minimum," arXiv:quant-ph/9607014.

[54] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla, "Quantum query complexity of some graph problems," arXiv:quant-ph/0401091.

[55] T. H. Cormen, C. E. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* MIT Press and McGraw-Hill, 3rd ed., 2009.

[56] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing* **2** no. 4, (1973) 225–231.

[57] C. Berge, "Two theorems in graph theory," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, pp. 842–844. 1957.

[58] A. Berzina, A. Dubrovsky, R. Freivalds, L. Lace, and O. Scegulnaja, "Quantum query complexity for some graph problems," in *Lecture Notes in Computer Science*, vol. 2932, pp. 140–150. Springer, 2004.

[59] S. Zhang, "On the power of Ambainis's lower bounds," *Theoretical Computer Science* **339** no. 2-3, (2005) 241–256, arXiv:quant-ph/0311060.

[60] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math Doklady* **11** (1970) 1277–1280.

[61] A. V. Karzanov, "O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh," in *Matematicheskie Voprosy Upravleniya Proizvodstvom*, L. Lyusternik, ed., vol. 5, pp. 81–94. Moscow State University Press, 1973.

[62] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM Journal on Computing* **4** no. 4, (1975) 507–518.

[63] S. Micali and V. V. Vazirani, "An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs," in *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 17–27. 1980.

[64] A. Childs. `http://www.math.uwaterloo.ca/~amchilds/teaching/w13/l15.pdf`, 2013.

[65] R. Bhatia, *Matrix Analysis.* Springer-Verlag, 1997.