New Quantum Algorithms and Quantum Lower Bounds

Shengyu Zhang

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Department of

Computer Science

September, 2006

Abstract

The last decade witnessed a great surge of fruitful studies in the new paradigm of quantum computing. Remarkable progress has been made in all the areas including quantum algorithms, quantum complexity theory, quantum error-correcting code, quantum information theory, quantum cryptography and physical implementations of quantum computers. However, quantum algorithm design, probably the most important task in quantum computing, did not make as much progress as people expected. This thesis aims at both designing new quantum algorithms and studying why efficient quantum algorithms are so hard to design by proving many lower bounds on quantum query complexity for various problems.

We first review (and generalize) known techniques for designing quantum algorithms and for proving quantum lower bounds. Using these techniques, we then study the quantum query complexity for many problems, including some specific problems such as Graph Matching and other natural graph properties, the general tuple search, and local search. We also study the lowest possible quantum query complexity for a class of functions as a whole, such as graph properties, circular functions, and functions invariant of a transitive permutation group, where tight results are given in all these cases.

Finally for the quantum lower bounds techniques themselves, we study the main two techniques and show that they are incomparable in power. For the most widely used one, which has a lot of parameters to choose, we show limitations for it in terms of certificate complexity. This implies that we cannot use the method to prove better lower bounds for almost all known open problems in this area.

Acknowledgements

I would like to take this opportunity to thank a number of people who helped me during my years as a graduate student at Princeton University.

First, I'd like to thank my thesis advisor Andy Yao, who has been helping me so much on many levels, including both research problems themselves and how to be a good researcher in general. He gave me a great deal of scientific guidance with his distinctive perspectives and insights on the theory of computing, and I now more and more realize how much I have been benefiting from all that he taught me. I believe these invaluable four-year collaborations influence encouragements, evaluations, and suggestions. He also invited me to go to Chinese University of Hong Kong in January, 2005 to work with him for a month, which helped me a lot for the study of the local search problem.

Second, I'd also like to thank my co-authers, including Mung Chiang, Prashanth Hande, Wei Huang, Yaoyun Shi, Xiaoming Sun, Powel Wocjan, and Yufan Zhu; I learned a lot from the collaborations with them. I'd mention especially that as a former student of Andy, Yaoyun greatly helped me during all these years at every aspect within and outside the research. Thanks also to people that I had scientific discussions with, including Scott Aaronson, Andris Ambainis, Sean Hallgren, Peter Hoyer, Sasha Razborov, Martin Roetteler, Miklos Santha, Pranab Sen, Mario Szegedy, John Watrous, and Avi Wigderson; I really learned a lot from all these communications.

What I also benefit from are the wonderful courses taught at Princeton University by various people like Sanjeev Arora, Boaz Barak, Mung Chiang, Moses Charikar, Bernard Chazelle, Amit Sahai, and Robert Tarjan. I am also indebted to the people inviting and supporting me to visit their places: John Preskill and Leonard Schulman at IQI (Caltech), Miklos Santha at CNRS (Université ParisSud), Yaoyun Shi at University of Michigan, and Andy Yao at Chinese University of Hong Kong.

The manuscript of the thesis was read by my advisor Andy Yao and two readers in my committee: Nicholas Pippenger and Sasha Razborov. Thank all of them very much for reading the manuscript so carefully and giving me so many detailed comments, corrections and suggestions to improve the quality the thesis to a great extent.

As always, I am indebted to my parents so much; they always encouraged, trusted and supported me with no reason. Em... actually, there is a reason: the love. Thank you and I love you too!

Finally, I acknowledge the financial support of the National Science Foundation under grants CCR-0310466 and CCF-0426582.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Summary of Results

We will start the thesis by giving a brief review of the history of quantum computing, followed by some discussions on the motivations of *quantum algorithm design* and *lower bound on quantum query complexity*, two important and closely related areas in quantum computing. Then we summarize the results in the thesis, which combines several previous research papers by the author [78, 79, 80] or jointly done with other co-authors [65, 71].

1.1 A brief history of quantum computing

The idea of using the distinctive nature of quantum mechanics to accelerate computation dates back to as early as 1982, when Feynman pointed out that simulating quantum systems on classical computers seems to have exponential slowdown and asked whether it is possible to build computers using quantum effects to have substantial speedup [33]. Deutsch took another big step by introducing the notion of universal quantum computer, quantum circuits and universal quantum gate sets [27, 28]. Concrete problems, like those proposed by Deutsch and Josza [29] and by Simon [68] were then constructed and used to show that quantum computer can have exponential speedup on some computational tasks. However, these problems seemed too artificial to be of big practical interest. This line of research culminated in Shor's [66] polynomial time algorithms for factoring and discrete logarithm problems, two problems used in many public-key cryptosystems because of their classical hardness — the best known classical algorithms use exponential time. This discovery caused great excitement among both theoretical computer scientists and experimental physicists, and greatly helped to open up the whole new paradigm of *quantum computing*.

Various other areas including quantum complexity theory, quantum error-correcting code, quantum information theory, quantum cryptography and physical implementations of quantum computers were also rapidly developed. See the textbook by Nielsen and Chuang [52] for an extensively introduction to this new and fantastic paradigm.

1.2 Quantum query complexity: What is it and why do we study it?

Among all the areas in quantum computing and quantum information processing, quantum algorithm design is one of the most important tasks. On one hand, we can solve more problems efficiently with faster quantum computers, on the other hand, this threatens the current cryptosystems. Therefore, there has never been lack of effort on designing efficient quantum algorithms.

There really were more new algorithms discovered. Shortly after Shor's algorithms, Grover found a simple searching algorithm which has a quadratic speedup over the best classical one [34]. Because searching is such a fundamental and ubiquitous problem, Grover's search is widely used as a primitive for quantum algorithm design. However, since it gives only a quadratic speedup, it was natural that people hoped to discover more quantum algorithms with exponential speedup over the best known classical ones. Two major candidate problems that seemed promising are Graph Isomorphism and (Gap) Shortest-lattice Vector; both are classically hard and very important on many levels. People then found that all these problems can be put into a general formalism of Hidden Subgroup Problem (HSP).

The HSP for a (finitely-generated) abelian group, which contains Factoring and Discrete Logarithm as special cases, is not hard and people know polynomial time quantum algorithm for it. For non-abelian groups, people have found efficient quantum algorithms for some special cases, including Hallgren's [36] polynomial time algorithm for Pell's equation and class groups, van Dam, Hallgren and Ip's polynomial time algorithm for some hidden shift problems [73], Kuperberg's subexponential time algorithm for HSP over the dihedral group [44], and some others. Most of them are of algebraic or number theoretical nature, and the basic technique is the quantum Fourier transform; see [49] for a survey of HSP. But the general case of non-abelian group, or even for the symmetric group and dihedral group (which correspond to the Graph Isomorphism and (Gap) Shortest-lattice vector problems the latter showed by [59]), are much harder than first appeared.

Why is it so hard to design quantum algorithms? Different people may have different explanations, but the two possibilities addressed by Shor [67] may be representative: First, there might be very few problems that quantum computers can solve substantially faster than classical computers; and second, we do not have much intuition of how quantum computers work since our experience is almost all from classical world.

Considering the enormous payoff of efficient quantum algorithms and the difficulty of discovering them, Shor suggested first looking at those problems that already have efficient classical algorithms. On studying them, we may find new quantum algorithm techniques, which may be used to design algorithms for broader class of problems, some of which may have exponential speedup by quantum algorithms!

This other line of research for quantum algorithm design has already been active since Grover's search. Different than the the Hidden Subgroup problems, these quantum algorithms are for problems with a searching flavor. The main techniques used in designing algorithms for these problems are Grover's search, quantum walk, and classical techniques like random sampling. Interestingly, for these problems, we can not only design algorithms, but also prove lower bounds on the query complexity, which we will briefly discuss next.

The quantum query model is a natural extension of the decision tree, a well studied classical computation model. In these models, algorithms can access the input only by making queries, and we care about the query complexity, *i.e.* the minimum number of queries needed to compute a function. In the quantum query model, we can use superposition to get all the variable information by just one query. Unfortunately, the answers are also in a superposition, thus it is not straightforward to extract the function value, the information we want. Also, the operations between quantum queries must be unitary, which makes the design of quantum query algorithms harder. Again, we care about the quantum query complexity of a function, the minimum number of quantum queries needed to compute it.

Pinning down the quantum query complexity for a problem may seem more like a mathematical recreation at first sight. However, we want to argue that quantum query complexity has many more implications for at least the following reasons.

1. Essentially all known quantum algorithms are query algorithms! We do not know any other techniques to design quantum algorithms. Therefore, a strong quantum lower bound[1] *is* a good indiction for the hardness of the problem for quantum computers.

2. The hardness result is not necessarily bad news — impossibility results can have important implications for other areas, such as cryptography. For example, the quantum lower bounds for the Permutation Inversion [15, 10] and Collision [5] problems leave open the possibility of one-way functions for quantum computers. Another example is that lower bounds on quantum query complexity can give some oracle results in quantum complexity theory, such as [3].

3. At first glance, query algorithms are all sublinear; while with quantum computers we hope to solve those problems not in **P** (or even **BPP**). So why is a sublinear quantum algorithm interesting at all?

In some cases, we can formalize the problem using exponentially many variables. Thus a linear algorithm for the new problem is an exponential algorithm for the original problem! For example, we can formalize the factoring problem as a Hidden Subgroup Problem with the size of the group being exponential of the size of the input of the factoring problem. Any

[1]In literature, the phrase "quantum lower bounds" usually means lower bounds on quantum query complexity.

known classical algorithm for factoring needs polynomial time for the new HSP problem, which is exponential time for the original factoring problem. But Shor's algorithm is of polylog time for that HSP problem. Therefore we should not ignore the quantum query complexity simply because it is studying sublinear algorithms — there may be reductions between linear time and exponential time!

4. It may sound a bit counterintuitive, but it turns out that the study of quantum lower bounds has implicit contributions to the quantum algorithm design. Since algorithm design and lower bounds are closely coupled, pinning down the quantum query complexity for various problems gradually guides the research community to those problems that still have a gap between upper and lower bounds. Then one of the two possibilities is that the upper bound is not tight, which means our old algorithm design techniques are not enough. With more effort put on these problems, new algorithm design technique may come out of it.

As a good example, after Aaronson and Shi's lower bound of $\Omega(N^{2/3})$ for Element Distinctness [5] and Buhrman *et al.* upper bound of $O(N^{3/4})$ (using Grover's search) [22], Ambainis gave the matching upper bound by using discrete quantum walk in a novel way, opening up a new avenue of quantum algorithm design.

5. More surprisingly, the study of quantum lower bounds yielded new lower bound techniques for *classical* query complexity (Aaronson [2])! In this thesis, we will also use the technique to get tight results for both quantum and classical query complexity for some problems.

1.3 Summary of results and the organization of the thesis

In Chapter 2 we give notations, definitions and basic theorems used in the thesis. In particular, we will summarize the two known techniques for proving quantum lower bounds: the polynomial method and the quantum adversary method. Different versions of the quantum adversary method are discussed, and a further generalization is presented.

In Chapter 3 we study the quantum query complexity for some specific problems. These include graph properties like Bipartiteness, Bipartite Matching, and Graph Matching. A lower bound of $\Omega(n^{1.5})$ is proved for all the three properties, improving the previous bound of $\Omega(n)$.

Also included is the promised version of the *tuple search*, which is to search for a k-tuple satisfying some pre-defined relation. When $k = 2$, it is essentially the (m, n) Claw-Finding problem. We get the tight bound $\Theta((mn)^{1/3})$ for this problem, improving both previous upper bound of $O(n^{1/2}m^{1/4} + m^{1/2})$ and previous lower bound of $\Omega(m^{1/2})$ (where without loss of generality they assume $m \geq n$) by Buhrman, Durr, Heiligman, Hoyer, Magniez, Santha and de Wolf [22].

Chapter 4 is devoted to Local Search, a problem defined as follows. On a fixed known graph, find a local minimum of a black-box function mapping each vertex to an integer. Local Search is of both practical and theoretical importance to combinatorial optimization, complexity theory and many other areas in theoretical computer science.

We study the problem in both the randomized and the quantum query models and give new lower and upper bound techniques in both models. The lower bound technique works for any graph that contains a product graph as a subgraph. Applying it to the Boolean hypercube $\{0, 1\}^n$ and the constant dimensional grids $[n]^d$, two particular product graphs that recently drew much attention, we get the following tight results:

$$RLS(\{0,1\}^n) = \Theta(2^{n/2}n^{1/2}), \quad QLS(\{0,1\}^n) = \Theta(2^{n/3}n^{1/6}); \tag{1.1}$$

$$RLS([n]^d) = \Theta(n^{d/2}),\ \forall d \geq 4, \quad QLS([n]^d) = \Theta(n^{d/3}),\ \forall d \geq 6. \tag{1.2}$$

Here $RLS(G)$ and $QLS(G)$ are the randomized and quantum query complexities of Local Search on G, respectively. For the other few small d's omitted in above formula, our new bounds also significantly improve the old ones. These improve the previous results by Aaronson [2], Ambainis (unpublished) and Santha and Szegedy[63].

Our new algorithms work well when the underlying graph expands slowly. As an application to $[n]^2$, a new quantum algorithm using $O(\sqrt{n}(\log\log n)^{1.5})$ queries is given. This improves the previous best known upper bound of $O(n^{2/3})$ (Aaronson, [2]), and implies that Local Search on grids exhibits different properties in low dimensions.

In Chapter 5, we consider the lowest possible quantum query complexity for a class of functions as a whole. In decision tree models, considerable attention has been paid on the effect of symmetry on computational complexity. That is, for a permutation group Γ, how low can the complexity be for any Boolean function invariant under Γ? We investigate this question for quantum decision trees for graph properties, directed graph properties, and circular functions. In particular, we prove that the n-vertex Scorpion graph property has quantum query complexity $\tilde{\Theta}(n^{1/2})$, which implies that the minimum quantum complexity for graph properties is strictly less than that for *monotone* graph properties (known to be $\Omega(n^{2/3})$). A directed graph property, SINK, is also shown to have the $\tilde{\Theta}(n^{1/2})$ quantum query complexity. Furthermore, we give an N-ary circular function which has the quantum query complexity $\tilde{\Theta}(N^{1/4})$. Finally, we show that for any permutation group Γ, as long as Γ is transitive, the quantum query complexity of any function invariant to Γ is at least $\Omega(N^{1/4})$, which implies that our examples are (almost) the best ones in the sense of pinning down the complexity for the corresponding permutation group.

In Chapter 6, we study the relations between the quantum query complexity and other complexity measures. First we show a lower bound for the quantum query complexity in terms of the total influence. In the second part, we show that in the tuple search task, if the components in the desired tuple are distributed, then we can consider both the quantum query complexity and the quantum communication complexity simultaneously. It is shown that we can trade one resource for another.

We then study in Chapter 7 the quantum lower bound techniques themselves. Since many lower bounds have been proved using the polynomial method and the quantum adversary method, it is natural to compare the power of the two techniques. Also, as we will see from the proof for the lower bounds on Local Search, the application of the quantum adversary method could be very complicated since it has a lot of parameters to choose. Sometimes other techniques, like heavy analysis at random walks on graphs for Local Search, may be involved to apply the method. Therefore, for a particular problem, it is natural to wonder whether we can use the quantum adversary method in some clever way to improve the previous lower bound.

Here we will answer both questions. For the quantum adversary method, we will show various limitations in terms of certificate complexity, a well studied classical complexity measure. Using these limitations, first we can get that the quantum adversary method is not always tight. In particular, for the Element Distinctness problem, the polynomial method can be used to get tight lower bound but we cannot use the quantum adversary method to achieve it, no matter how we pick all the parameters.

More importantly, these limitations imply that for many problems such as TRIANGLE, k-CLIQUE, BIPARTITENESS, BIPARTITE/GRAPH MATCHING and AND-OR TREE which draw wide interest and whose quantum query complexities are still open, the best known lower bounds cannot be further improved by using the quantum adversary method!

Chapter 2

Preliminaries and Basic Theorems

In this chapter we give some notations, definitions and basic theorems. Some notations or definitions are just used in one chapter, then we will put those in the beginning of that chapter.

Theorem 7 and Theorem 8 are from paper [78]

2.1 General notations

In the thesis, we will consider various computation tasks. There could be different formalizations of a computation task; one standard way is as follows. For a function $f : I^n \rightarrow O$, we are required to design an algorithm such that for any input $x = x_1x_2...x_n$, the algorithm runs on the input x and gives the desired output $f(x)$. Here I and O are input and output sets, respectively, and the function is n-ary, which means that it takes n variables as an input. Sometimes we could also consider other generalizations, such as partial functions, where the input set is a subset of I^n, and searching problems (also called relations), where for each input, there may be more than one "correct" output, and the algorithm succeeds by outputting any one of them.

We use $[n]$ to denote the set $\{1, 2, ..., n\}$. For an n-bit binary string $x = x_1...x_n \in \{0, 1\}^n$, let $x^{(i)} = x_1...x_{i-1}(1-x_i)x_{i+1}...x_n$, *i.e.* the string obtained from x by flipping the coordinate i. In general, for a subset $I \subseteq [n]$, x^I is obtained from x by flipping all the coordinates in I.

2.2 A brief review of quantum mechanics in its mathematical framework

The mathematical model for quantum mechanics is very simple.

1. Any isolated quantum system corresponds to a quantum state $|\psi\rangle$, which is mathematically a unit vector in a Hilbert space $\mathcal{H}$. The Dirac notation "ket" $|.\rangle$ is used to denote a quantum state.

2. The evolution of a closed quantum system is specified by a unitary operator U in $\mathcal{H}$.

3. A measurement on orthogonal subspaces S_1 and S_2 (where $S_1 \oplus S_2 = \mathcal{H}$) causes the system collapse to $P_1|\psi\rangle/\|P_1|\psi\rangle\|$ or $P_2|\psi\rangle/\|P_1|\psi\rangle\|$, with probability $\|P_1|\psi\rangle\|^2$ and $\|P_2|\psi\rangle\|^2$, respectively, where P_i is the projector on the subspace S_i. When it collapses to S_i, then an outcome i is observed.

4. A quantum system including two components whose corresponding states are $|\psi_1\rangle$ and $|\psi_2\rangle$ corresponds to the state $|\psi_1\rangle \otimes |\psi_2\rangle$, where $\otimes$ is the tensor product.

For a comprehensive introduction to quantum computing, please refer to the textbooks [52, 43].

2.3 Query (decision tree) computation models: deterministic, randomized and quantum

A deterministic query algorithm, classically also called a decision tree algorithm, is defined as follows. For a function $f : I^n \to O$, the query algorithm accesses the input $x \in I^n$ only by making queries of the form of "$x_i =?$". The decision of which query to make for the next step can depend on the variable values already got from previous queries. Each query has cost 1, and all the other computation between queries is free. The naive way to compute any function f is just to query all the variables and thus get all the variable information, which of course fixes the function value on this input. However, for some functions, we can do the computation by only making a small number of queries. For a function f, the minimum number of queries needed to compute $f(x)$ for any input x is the deterministic query complexity of f, denoted by $D(f)$. More formally,

$$D(f) = \min_{\mathcal{A}:\ deterministic} \max_{x}\ cost(\mathcal{A}, x) \tag{2.1}$$

where the minimum is over all the deterministic query algorithms that compute f correctly on every input, and the maximum is over all the inputs x, and the cost is the number of queries $\mathcal{A}$ makes on the input x.

A randomized query algorithm is the same except that the algorithm can toss coins at each step to decide which variable x_i to ask next. The randomized query complexity, denoted by $R(f)$, is the minimum expected number of queries needed for a randomized query algorithm to compute $f(x)$ for any input x. More formally,

$$R(f) = \min_{\mathcal{A}:\ randomized} \max_{x}\ \mathbf{E}_{rand(\mathcal{A})}[cost(\mathcal{A}, x)] \tag{2.2}$$

We can also allow a small error probability ϵ. There are two variants of the cost measures: the worst case and the average case. The worst case cost of a randomized algorithm $\mathcal{A}$ is the number of queries that $\mathcal{A}$ makes on the worst input and the worst-case outcomes of $\mathcal{A}$'s coins. The average case cost of a randomized algorithm $\mathcal{A}$ is the expected number of queries that $\mathcal{A}$ makes on the worst input, where the expectation is over $\mathcal{A}$'s randomness. More formally,

$$R_{2,\epsilon}^{worst}(f) = \min_{\mathcal{A}:\ randomized, \epsilon-error} \max_{x, rand(\mathcal{A})}\ cost(\mathcal{A}, x) \tag{2.3}$$

$$R_{2,\epsilon}^{expected}(f) = \min_{\mathcal{A}:\ randomized, \epsilon-error} \max_{x} \mathbf{E}_{rand(\mathcal{A})}[cost(\mathcal{A}, x)] \tag{2.4}$$

Since $R_{2,\epsilon+\delta}^{worst}(f) \leq \frac{1}{\delta} R_{2,\epsilon}^{expected}(f)$ by Markov's Inequality, we usually do not distinguish these two measures explicitly and just use $R_2(f)$ to denote the double-sided constant error probability randomized query complexity of f.

The quantum query model, formally introduced in [20], has a working state in the form of $\sum_{i,a,z} \alpha_{i,a,z} |i, a, z\rangle$. A quantum query on the input x proceeds as follows.

$$\sum_{i,a,z} \alpha_{i,a,z} |i, a, z\rangle \rightarrow \sum_{i,a,z} \alpha_{i,a,z} |i, a \oplus x_i, z\rangle \tag{2.5}$$

where i is the position that we are currently interested in, the register where a is holds the answer x_i by XOR-ing x_i to a, and z is something not involved in this step of query computation. A T-query quantum query algorithm works as a sequence of operations

$$U_0 \to O_x \to U_1 \to O_x \to ... \to U_{T-1} \to O_x \to U_T \tag{2.6}$$

where O_x is as defined above, and each U_t is a unitary operation that does not depend on the input x. In the quantum query model, it is more natural to consider the bounded error case because practical quantum systems have various kinds of errors anyway. The ϵ quantum query complexities, denoted by $Q_{2,\epsilon}(f)$, is the minimum numbers of queries we need to make to compute f by a quantum query algorithm with error probability at most ϵ. Again, the quantum query complexity is $Q_2(f) = Q_{2,1/3}(f)$.

For more details on the query models, the corresponding query complexities, many other complexity measures, and known relations among them, we refer to [23] by Buhrman and de Wolf as an excellent survey.

2.4 Some other complexity measures

In this section, we review some complexity measures.

We first define sensitivity, block sensitivity and certificate complexity of a function f. Recall that for a subset $I \subseteq [n]$, x^I is obtained from x by flipping all the coordinates in I.

Definition 1 *The sensitivity of a Boolean function f on input $x = x_1x_2...x_n \in \{0,1\}^n$ is*

$$s(f,x) = |\{i \in \{1,...,n\} : f(x) \neq f(x^{(i)})\}|, \tag{2.7}$$

where $x^{(i)}$ is the n-bit string obtained from x by flipping x_i. The sensitivity of f is

$$s(f) = \max_{x \in \{0,1\}^n} s(f,x). \tag{2.8}$$

Definition 2 *The block sensitivity of f on x is the maximum number b such that there are b disjoint sets $B_1, ..., B_b \subseteq [n]$ for which $f(x) \neq f(x^{B_i})$. The block sensitivity of f is $bs(f) = max_x bs(f,x)$.*

Definition 3 *A certificate set CS_x of f on x is a set of indices such that $f(x) = f(y)$ whenever $y_i = x_i$ for all $i \in CS_x$. The certificate complexity $C(f, x)$ of f on x is the size of a smallest certificate set of f on x. The b-certificate complexity of f is $C_b(f) =$* $\max_{x:f(x)=b} C(f, x)$. *The certificate complexity of f is $C(f) = \max\{C_0(f), C_1(f)\}$.*

It is obvious that $s(f) \le bs(f) \le C(f)$.

Definition 4 *A polynomial $p(x)$ represents a function $f : \{0, 1\}^n \to \mathbb{R}$ if $p(x) = f(x)$ for all $x \in \{0, 1\}^n$. The degree $deg(f)$ of a function is the degree of the multilinear polynomial representing f.*

This is well defined because it is not hard to see that there exists a unique multilinear polynomial presenting f.

Definition 5 *A polynomial $p(x)$ approximates a function $f : \{0, 1\}^n \to \{0, 1\}$ up to ϵ if $|p(x) - f(x)| \le \epsilon$ for all $x \in \{0, 1\}^n$. The ϵ-approximate degree $\widetilde{deg}_\epsilon(f)$ of a function is the minimum degree of a multilinear polynomial approximates f up to ϵ. The approximate degree $\widetilde{deg}(f) = \widetilde{deg}_{1/3}(f)$.*

2.5 Two main tools for proving quantum lower bounds

There are mainly two methods for proving quantum lower bounds: the polynomial method and the quantum adversary method. We will briefly introduce these two tools now, and give a generalized version of the quantum adversary method. See [39] for a comprehensive survey of this research area.

2.5.1 The polynomial method

The polynomial method is essentially the following theorem by Beals, Buhrman, Cleve, Mosca and de Wolf [20].

Theorem 1 *(Beals,* et al. *[20])*

$$Q_{2,\epsilon}(f) \ge \widetilde{deg}_\epsilon(f), \quad \textit{and in particular,}\ Q_2(f) \ge \widetilde{deg}(f) \tag{2.9}$$

An important result from this theorem is that the deterministic and quantum query complexities for *any* total Boolean function are polynomially related.

Theorem 2 *(Beals,* et al. *[20])*

$$D(f) \leq O(Q_2(f)^6). \tag{2.10}$$

So to prove a quantum lower bound, it is enough to prove the lower bound for the approximate degree. Some theorems are known from the approximation theory, and one example is as follows whose proof can be found in Chapter 4 of [30].

Theorem 3 *(Bernstein Inequality) For any univariate polynomial $p(t)$ with degree d and $||p||_{[-1,1]} = 1$, we have*

$$|p'(t)| \leq d/\sqrt{1-t^2}, \qquad \forall t \in (-1,1) \tag{2.11}$$

where $||p||_D = \sup_{t\in D} |p(t)|$.

It is worth to remark that only lower bounds for univariate polynomials are known, following the Theorem 3 and some other ones in a similar spirit. So it seems hard to use the polynomial method to pin down the exact quantum query complexity for non-symmetric functions. (It is easy to apply the method to get the quantum query complexity for any symmetric function [20].)

2.5.2 The quantum adversary method

Unlike the polynomial method, the quantum adversary method looks complicated. However, as will be seen later, it is powerful enough to achieve good lower bounds for many nonsymmetric functions. The quantum adversary method was first proposed by Ambainis [10], where two versions of the method were given.

Theorem 4 (Ambainis, [10]) *Let $f : \{0,1\}^N \to \{0,1\}$ be a function and X, Y be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ be a relation such that*

1. *$\forall x \in X$, there are at least m different $y \in Y$ such that $(x,y) \in R$.*

2. *$\forall y \in Y$, there are at least m' different $x \in X$ such that $(x,y) \in R$.*

3. *$\forall x \in X$,$\forall i \in [N]$, there are at most l different $y \in Y$ such that $(x,y) \in R$, $x_i \neq y_i$.*

4. *$\forall y \in Y$,$\forall i \in [N]$, there are at most l' different $x \in X$ such that $(x,y) \in R$, $x_i \neq y_i$.*

Then

$$Q_2(f) = \Omega\left(\sqrt{\frac{mm'}{ll'}}\right). \tag{2.12}$$

Since this method looks complicated, it is worth to make some remarks about its intuition before giving the second version of the method. Note that if $f(x) \neq f(y)$, thus any quantum algorithm needs to distinguish x and y. The first condition says that for x, there are a large number m of y's to distinguish from x; the third condition says that querying one position can only distinguish x from a small number l of y's. Thus we need some number like m/l of queries to distinguish x and all y's that $(x,y) \in R$. Similarly for m' and l'. Of course, the argument is far from rigorous; it however gives a intuitive feeling why this might be a lower bound.

The second version of the method is as follows.

Theorem 5 (Ambainis, [10]) *Let $f : I^N \to \{0,1\}$ be a Boolean function where I is a finite set, and X, Y be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ satisfy*

1. *$\forall x \in X$, there are at least m different $y \in Y$ such that $(x,y) \in R$.*

2. *$\forall y \in Y$, there are at least m' different $x \in X$ such that $(x,y) \in R$.*

Let

$$l_{x,i} = |\{y : (x,y) \in R, x_i \neq y_i\}|, \qquad l_{y,i} = |\{x : (x,y) \in R, x_i \neq y_i\}| \tag{2.13}$$

and

$$l_{max} = \max_{x,y,i:(x,y)\in R, i\in[N], x_i\neq y_i} l_{x,i} l_{y,i}. \tag{2.14}$$

Then

$$Q_2(f) = \Omega\left(\sqrt{\frac{mm'}{l_{max}}}\right). \tag{2.15}$$

Clearly, the second one generalizes the first one. Ambainis later generalized Theorem 4 in another way by putting different weights on different input pairs.

Definition 6 *Let $f : I^N \to [M]$ be an N-variate function. Let $R \subseteq I^N \times I^N$ be a relation such that $f(x) \neq f(y)$ for any $(x,y) \in R$. A weight scheme consists of three weight functions $w(x,y) > 0$, $u(x,y,i) > 0$ and $v(x,y,i) > 0$ satisfying*

$$u(x,y,i)v(x,y,i) \geq w^2(x,y) \tag{2.16}$$

for all $(x,y) \in R$ and $i \in [N]$ with $x_i \neq y_i$. We further put

$$w_x = \textstyle\sum_{y':(x,y')\in R} w(x,y'), \qquad w_y = \displaystyle\sum_{x':(x',y)\in R} w(x',y), \tag{2.17}$$

$$u_{x,i} = \textstyle\sum_{y':(x,y')\in R, x_i\neq y'_i} u(x,y',i), \qquad v_{y,i} = \displaystyle\sum_{x':(x',y)\in R, x'_i\neq y_i} v(x',y,i). \tag{2.18}$$

Theorem 6 (Ambainis, [9]) *Let $f : I^N \to \{0,1\}$ where I is a finite set, and $X \subseteq f^{-1}(0)$, $Y \subseteq f^{-1}(1)$ and $R \subseteq X \times Y$. Let w, u, v be a weight scheme for X, Y, R. Then*

$$Q_2(f) = \Omega\left(\sqrt{\min_{x\in X, i\in[N]} \frac{w_x}{u_{x,i}} \cdot \min_{y\in Y, j\in[N]} \frac{w_y}{v_{y,j}}}\right) \tag{2.19}$$

Since Theorem 5 and Theorem 6 generalize Theorem 4 in different ways, two immediate questions can be asked. First, are their powers comparable? Second, can we combine both approaches and give an further generalized method? For the first question, it turns out that Theorem 6 is always no worse than Theorem 5.

Theorem 7 *Any lower bounds achieved by Theorem 5 can also be achieved by Theorem 6.*

Proof For any X, Y, R in Theorem 5, we set the weight functions in Theorem 6 as follows. Let $w(x,y) = 1$, $u(x,y,i) = \sqrt{l_{max}}/l_{x,i}$ and $v(x,y,i) = \sqrt{l_{max}}/l_{y,i}$. It's easy to check that

$$u(x,y,i)v(x,y,i) = \frac{l_{max}}{l_{x,i}l_{y,i}} \geq 1 = w(x,y)^2 \tag{2.20}$$

Note that $u(x,y,i)$ is independent of y, so we have $u_{x,i} = l_{x,i}u(x,y,i) = \sqrt{l_{max}}$. Symmetrically, it follows that $v_{y,i} = \sqrt{l_{max}}$. Thus, by denoting $m_x = |\{y : (x,y) \in R\}|$ and $m_y = |\{x : (x,y) \in R\}|$, we have

$$\min_{x,i} \frac{w_x}{u_{x,i}} \min_{y,i} \frac{w_y}{v_{y,i}} = \min_{x,i} \frac{m_x}{\sqrt{l_{max}}} \min_{y,i} \frac{m_y}{\sqrt{l_{max}}} = \frac{m}{\sqrt{l_{max}}} \frac{m'}{\sqrt{l_{max}}} = \frac{mm'}{l_{max}},$$

which means that for any X, Y, R in Theorem 5, the lower bound result can be also achieved by Theorem 6. □

For the second question, the answer is also affirmative. The following theorem is slightly more general than the one in [78] (Zhang) which works only for Boolean functions. The proof is almost the same.

Theorem 8 *For any f, R and any weight scheme w, u, v as in Definition 6, we have*

$$Q_2(f) = \Omega\left(\min_{(x,y)\in R, i\in[N]:\ x_i \neq y_i} \sqrt{\frac{w_x w_y}{u_{x,i} v_{y,i}}}\right). \tag{2.21}$$

Proof As mentioned before, the query computation is a sequence of operations $U_0 \rightarrow O_x \rightarrow U_1 \rightarrow \ldots \rightarrow U_T$ on some fixed initial state, say $|0\rangle$. Note that here T is the number of queries. Denote

$$|\psi_x^k\rangle = U_{k-1}O_x \ldots U_1 O_x U_0 |0\rangle. \tag{2.22}$$

Note that $|\psi_x^0\rangle = |0\rangle$ for all input x. Because the computation is correct with high probability $(1-\epsilon)$, for any $(x,y) \in R$, the two final states must be at some distance to be distinguishable. In other words, we can assume that $|\langle \psi_x^T | \psi_y^T \rangle| \leq c$ for some constant $c < 1$. Now suppose that

$$|\psi_x^{k-1}\rangle = \sum_{i,a,z} \alpha_{i,a,z} |i,a,z\rangle, \qquad |\psi_y^{k-1}\rangle = \sum_{i,a,z} \beta_{i,a,z} |i,a,z\rangle, \tag{2.23}$$

and the oracle works as follows:

$$O_x |\psi_x^{k-1}\rangle = \sum_{i,a,z} \alpha_{i,a,z} |i, a \oplus x_i, z\rangle = \sum_{i,a,z} \alpha_{i,a\oplus x_i,z} |i,a,z\rangle, \tag{2.24}$$

$$O_y |\psi_y^{k-1}\rangle = \sum_{i,a,z} \beta_{i,a,z} |i, a \oplus y_i, z\rangle = \sum_{i,a,z} \beta_{i,a\oplus y_i,z} |i,a,z\rangle. \tag{2.25}$$

So we have

$$\begin{aligned}
\langle\psi_x^k|\psi_y^k\rangle &= \sum_{i,a,z}\alpha^*_{i,a\oplus x_i,z}\beta_{i,a\oplus y_i,z} && (2.26)\\
&= \sum_{i,a,z:x_i=y_i}\alpha^*_{i,a\oplus x_i,z}\beta_{i,a\oplus y_i,z}+\sum_{i,a,z:x_i\neq y_i}\alpha^*_{i,a\oplus x_i,z}\beta_{i,a\oplus y_i,z} && (2.27)\\
&= \langle\psi_x^{k-1}|\psi_y^{k-1}\rangle+\sum_{i,a,z:x_i\neq y_i}\alpha^*_{i,a\oplus x_i,z}\beta_{i,a\oplus y_i,z}-\sum_{i,a,z:x_i\neq y_i}\alpha^*_{i,a,z}\beta_{i,a,z}. && (2.28)
\end{aligned}$$

Thus

$$\begin{aligned}
1-c &\leq 1-|\langle\psi_x^T|\psi_y^T\rangle| && (2.29)\\
&= \sum_{k=1}^{T}(|\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle|-|\langle\psi_x^k|\psi_y^k\rangle|) && (2.30)\\
&\leq \sum_{k=1}^{T}|\langle\psi_x^{k-1}|\psi_y^{k-1}\rangle-\langle\psi_x^k|\psi_y^k\rangle| && (2.31)\\
&= \sum_{k=1}^{T}|\sum_{i,a,z:x_i\neq y_i}(\alpha^*_{i,a\oplus x_i,z}\beta_{i,a\oplus y_i,z}-\alpha^*_{i,a,z}\beta_{i,a,z})| && (2.32)\\
&\leq \sum_{k=1}^{T}\sum_{i,a,z:x_i\neq y_i}(|\alpha_{i,a\oplus x_i,z}||\beta_{i,a\oplus y_i,z}|+|\alpha_{i,a,z}||\beta_{i,a,z}|). && (2.33)
\end{aligned}$$

Summing up the inequalities for all $(x,y)\in R$, with weight $w(x,y)$ multiplied, yields

$$\begin{aligned}
&(1-c)\sum_{(x,y)\in R}w(x,y) && (2.34)\\
\leq&\sum_{k=1}^{T}\sum_{(x,y)\in R}\sum_{i,a,z:x_i\neq y_i}w(x,y)(|\alpha_{i,a\oplus x_i,z}||\beta_{i,a\oplus y_i,z}|+|\alpha_{i,a,z}||\beta_{i,a,z}|) && (2.35)\\
\leq&\sum_{k=1}^{T}\sum_{(x,y)\in R}\sum_{i,a,z:x_i\neq y_i}\sqrt{u(x,y,i)v(x,y,i)}(|\alpha_{i,a\oplus x_i,z}||\beta_{i,a\oplus y_i,z}|+|\alpha_{i,a,z}||\beta_{i,a,z}|) && (2.36)\\
=&\sum_{k=1}^{T}\sum_{i,a,z}\sum_{(x,y)\in R:x_i\neq y_i}\sqrt{u(x,y,i)v(x,y,i)}(|\alpha_{i,a\oplus x_i,z}||\beta_{i,a\oplus y_i,z}|+|\alpha_{i,a,z}||\beta_{i,a,z}|) && (2.37)
\end{aligned}$$

by the definition of weight scheme. We then use inequality $2AB\leq A^2+B^2$ to get

$$\begin{aligned}
&\sqrt{u(x,y,i)v(x,y,i)}|\alpha_{i,a\oplus x_i,z}||\beta_{i,a\oplus y_i,z}|\\
\leq&\frac{1}{2}(u(x,y,i)\sqrt{\frac{v_{y,i}}{u_{x,i}}\frac{w_x}{w_y}}|\alpha_{i,a\oplus x_i,z}|^2+v(x,y,i)\sqrt{\frac{u_{x,i}}{v_{y,i}}\frac{w_y}{w_x}}|\beta_{i,a\oplus y_i,z}|^2), && (2.38)
\end{aligned}$$

and

$$\begin{aligned}
&\sqrt{u(x,y,i)v(x,y,i)}|\alpha_{i,a,z}||\beta_{i,a,z}| \\
\leq\ & \frac{1}{2}\left(u(x,y,i)\sqrt{\frac{v_{y,i}}{u_{x,i}}\frac{w_x}{w_y}}|\alpha_{i,a,z}|^2 + v(x,y,i)\sqrt{\frac{u_{x,i}}{v_{y,i}}\frac{w_y}{w_x}}|\beta_{i,a,z}|^2\right). \qquad (2.39)
\end{aligned}$$

Denote $A = \min_{x,y,i:(x,y)\in R, x_i\neq y_i} \frac{w_x w_y}{u_{x,i}v_{y,i}}$. Note that

$$\sum_{y:(x,y)\in R, x_i\neq y_i} u(x,y,i) = u_{x,i}, \qquad \sum_{x:(x,y)\in R, x_i\neq y_i} v(x,y,i) = v_{y,i} \qquad (2.40)$$

by the definition of $u_{x,i}$ and $v_{y,i}$. We have

$$\begin{aligned}
& (1-c)\sum_{(x,y)\in R} w(x,y) && (2.41)\\
\leq\ & \frac{1}{2}\sum_{k=1}^{T}\sum_{i,a,z}[\sum_{x\in X}\sqrt{\frac{u_{x,i}v_{y,i}}{w_x w_y}}w_x(|\alpha_{i,a\oplus x_i,z}|^2 + |\alpha_{i,a,z}|^2) \\
& \qquad + \sum_{y\in Y}\sqrt{\frac{u_{x,i}v_{y,i}}{w_x w_y}}w_y(|\beta_{i,a\oplus y_i,z}|^2 + |\beta_{i,a,z}|^2)] && (2.42)\\
\leq\ & \frac{1}{2}\sum_{k=1}^{T}[\sum_{x\in X}\sqrt{1/A}w_x\sum_{i,a,z}(|\alpha_{i,a\oplus x_i,z}|^2 + |\alpha_{i,a,z}|^2) \\
& \qquad + \sum_{y\in Y}\sqrt{1/A}w_y\sum_{i,a,z}(|\beta_{i,a\oplus y_i,z}|^2 + |\beta_{i,a,z}|^2)] && (2.43)\\
=\ & \sqrt{1/A}\sum_{k=1}^{T}(\sum_{x\in X}w_x + \sum_{y\in Y}w_y) && (2.44)\\
=\ & 2T\sqrt{1/A}\sum_{(x,y)\in R} w(x,y) && (2.45)
\end{aligned}$$

by noting that $\sum_x w_x = \sum_y w_y = \sum_{(x,y)\in R} w(x,y)$. Therefore,

$$T = \Omega(\sqrt{A}) = \Omega\left(\min_{(x,y)\in R, i\in[N]:\ x_i\neq y_i}\sqrt{\frac{w_x w_y}{u_{x,i}v_{y,i}}}\right), \qquad (2.46)$$

as desired. □

There are also some other generalizations [14, 46]. Recently Spalek and Szegedy made the picture clear by showing that all these generalizations are equivalent in power [69].

The relational adversary method

Inspired by the quantum adversary method, Aaronson [2] gave a nice technique to get a lower bound for randomized query complexity. We restate it using language similar to that of Theorem 8.

Theorem 9 [Aaronson, [2]] *Let $F : I^N \to [M]$ be an N-variate function. Let $R \subseteq I^N \times I^N$ be a relation such that $F(x) \neq F(y)$ for any $(x, y) \in R$. For any weight function $w : R \to \mathbb{R}^+$, we have*

$$R_2(F) = \Omega\left(\min_{(x,y)\in R, i\in[N], x_i\neq y_i} \max\left\{\frac{w_x}{w_{x,i}}, \frac{w_y}{w_{y,i}}\right\}\right) \tag{2.47}$$

where

$$w_{x,i} = \sum_{y':(x,y')\in R, x_i\neq y'_i} w(x, y'), \qquad w_{y,i} = \sum_{x':(x',y)\in R, x'_i\neq y_i} w(x', y). \tag{2.48}$$

Note that we can think of Theorem 9 as having a weight scheme too, but requiring that $u(x, y, i) = v(x, y, i) = w(x, y)$. This simple observation is used in the proofs of the theorems for Local Search problems in Chapter 4.

Chapter 3

Quantum Query Complexities of Specific Problems

In this chapter, we will study the quantum query complexity for several specific problems. In Section 3.1, we will give new quantum lower bounds for some graph properties. In Section 3.2, we will study tuple search problems. As a special case, we solve the quantum query complexity of a specific problem called Claw-Finding. Here we get the upper bound by the discrete quantum walk, where we generalize Ambainis' idea by performing two (or more) discrete quantum walks simultaneously to achieve the optimal algorithm.

The results in Section 3.1 are from paper [78]. The same $\Omega(n^{1.5})$ lower bound for Matching was independently obtained by Berzina, Dubrovsky, Freivalds, Lace and Scegulnaja in [18], and the same lower bound for Bipartiteness was independently obtained by Durr (mentioned in [46]).

The results in Section 3.2 are from paper [79].

3.1 Quantum query complexities for some natural graph properties

We consider the following three graph properties. We assume that the input for the graph property problems are given by adjacency matrix.

1. Bipartiteness: *Given an undirected graph, decide whether it is a bipartite graph.*
2. Graph Matching: *Given an undirected graph, decide whether it has a perfect matching.*
3. Bipartite Matching: *Given an undirected bipartite graph, decide whether it has a perfect matching.*

We will use Theorem 5 to show the following results.

Theorem 10 *All the three graph properties, Bipartiteness, Bipartite Matching and Graph Matching, have $Q_2(f) = \Omega(n^{1.5})$.*

Proof 1. Bipartiteness. Without loss of generality, we assume n is even, because otherwise we can use the following argument on arbitrary $n-1$ (out of total n) nodes and leave the n^{th} node isolated. Let

$X = \{G : G$ is composed of a single n-length cycle$\}$,

$Y = \{G : G$ is composed of two cycles with each length being an odd number between $n/3$ and $2n/3\}$, and

$R = \{(G, G') \in X \times Y : \exists$ four nodes v_1, v_2, v_3, v_4 such that the only difference between graphs G and G' is that $(v_1, v_2), (v_3, v_4)$ are edges in G but not in G' and $(v_1, v_3), (v_2, v_4)$ are edges in G' but not in $G\}$.

Note that a graph is bipartite if and only if it contains no cycle with odd length. Therefore, any graph in X is a bipartite graph because n is even, and any graph in Y is not bipartite graph because it contains two odd-length cycles. Then all the remaining analysis is the same as calculation in the proof for Graph Connectivity (undirected graph and matrix input) in [31]. For completeness, it is not hard to find that $m = \Theta(n^2)$, $m' = \Theta(n^2)$, and for any fixed x, y, i such that $x_i \neq y_i$, either $l_{x,i} = O(1)$ and $l_{y,i} = O(n)$, or $l_{x,i} = O(n)$ and $l_{y,i} = O(1)$. Thus by Theorem 5, we get $Q_2(Bipartiteness) = \Omega(n^{1.5})$.

2. Bipartite Matching. Let X be the set of the bipartite graphs like Figure 3.1(a) where τ and σ are two permutations of $\{1, ..., n\}$, and $\frac{n}{3} \leq k \leq \frac{2n}{3}$. Let Y be the set of the bipartite graphs like Figure 3.1(b), where τ' and σ' are two permutations of $\{1, ..., n\}$, and also $\frac{n}{3} \leq k' \leq \frac{2n}{3}$. It is easy to see that all graphs in X have no perfect matching, while all graphs in Y have a perfect matching.

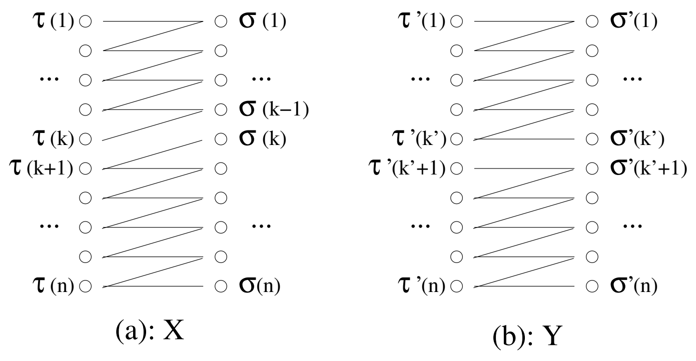

Figure 3.1: X and Y: a pair of inputs

Let R be the set of all pairs of $(x, y) \in X \times Y$ as in Figure 3.2, where graph y is obtained from x by choosing two horizontal edges $(\tau(i), \sigma(i))$, $(\tau(j), \sigma(j))$, removing them, and adding two edges $(\tau(i), \sigma(j))$, $(\tau(j), \sigma(i))$.

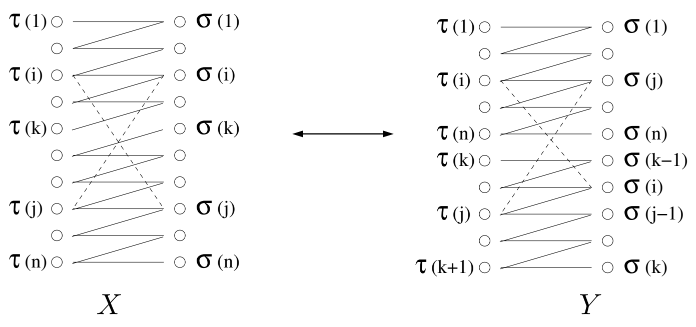

Figure 3.2: $R \subseteq X \times Y$: the relation

Now it is not hard to calculate the m, m', l_{max} in Theorem 5. For example, to get m we study x in two cases. When $\frac{n}{3} \leq k \leq \frac{n}{2}$, any edge $(\tau(i), \sigma(i))$ where $i \in [k - n/3, k]$ has at least $n/6$ choices for edge $(\tau(j), \sigma(j))$ because the only requirement for choosing is that $k' \in [n/3, 2n/3]$ and $k' = i + n - j$. The case when $\frac{n}{2} \leq k \leq \frac{2n}{3}$ can be handled symmetrically. Thus $m = \Theta(n^2)$. The same argument yields $m' = \Theta(n^2)$. Finally, for l_{max}, we note that if the edge $e = (\tau(i), \sigma(i))$ for some i, then $l_{x,e} = O(n)$ and $l_{y,e} = 1$; if the edge $e = (\tau(i), \sigma(j))$ for some i, j, then $l_{x,e} = 1$ and $l_{y,e} = O(n)$. For all other edges e, $l_{x,e} = l_{y,e} = 0$. Putting all cases together, we have $l_{max} = O(n)$. Thus by Theorem 5, we know that $Q_2(\mathit{Bipartite\ Matching}) = \Omega(n^{1.5})$.

3. Graph Matching. This can be easily shown by noting that Bipartite Matching is a special case of Graph Matching. □

3.2 Promised tuple search and the Claw-Finding problem

3.2.1 Introduction

Recently Ambainis [12] proposed a novel algorithm for k-Element Distinctness, which is to decide whether there are k equal elements in a given set A of size N. As later pointed out by Magniez, Santha and Szegedy [50] and by Childs and Eisenberg [25], Ambainis' algorithm actually gives an $O(N^{k/(k+1)})$ algorithm for the general k-Subset Finding problem, defined as follows.

k-Subset Finding: Given N elements $x_1, ...x_N \in [M]$, and a k-ary relation $R \subseteq [M]^k$, decide whether there is a k-size set $\{i_1, ..., i_k\}$ such that $(x_{i_1}, ..., x_{i_k}) \in R$. If yes, output a solution; otherwise reject.

Intuitively speaking, this is a tuple search: we search for a tuple satisfying some predefined relation. This generalizes Grover's search [34], which can be viewed as the special case of $k = 1$. We can also define the Unique k-Subset Finding problem, which is the same as k-Subset Finding except that it is promised that there is at most one solution set $\{i_1, ..., i_k\}$. As pointed out in [50], by a standard random reduction, we can solve k-Subset

FINDING with the same complexity as the UNIQUE k-SUBSET FINDING. Therefore, in what follows we mostly study the unique version of the problem.

Unlike the case of single element search (as in Grover's search), two new problems arise distinctively for tuple search. One is to consider what if we already have some information about one component in the desired tuple. For example, we know that there is a desired pair $(i_1, ..., i_j)$ such that $i_1 \in S$ for some subset $S \in [n]$ which is small, say $\sqrt{n}$. Could we do better than the standard Ambainis' search? The other question is to consider the scenario in which the components in the desired tuple are distributed, and we study both the communication cost and the query cost simultaneously in the tuple finding task. The second question is addressed in Chapter 6.

To make the first question more precise, consider the following problems.

UNIQUE (m,n) 2-SUBSET FINDING: We are given $x_1, ..., x_N \in [M]$, two sets of indices $J_1, J_2 \subseteq [N]$ with $|J_1| = m, |J_2| = n$, and a relation $R \subseteq [M] \times [M]$, with the promise that there exists at most one pair $(x_{j_1}, x_{j_2}) \in R$ such that $j_1 \in J_1$, $j_2 \in J_2$ and $j_1 \neq j_2$. Output the unique pair if it exists, and reject otherwise.

CLAW FINDING: The above problem with the restrictions that R is the Equality relation and $J_1 \cap J_2 = \emptyset$.

The best previous result for the CLAW FINDING is given by Buhrman, Durr, Heiligman, Hoyer, Magniez, Santha and de Wolf [22] [1] :

$$\Omega(m^{1/2}) \leq Q_2(\text{CLAW-FINDING}) \leq O((n^{1/2}m^{1/4} + m^{1/2}) \log n) \tag{3.1}$$

where without loss of generality, they assume $m \geq n$. In this paper, we improve it to the following tight bounds.

Theorem 11 *For both* UNIQUE (m,n) 2-SUBSET FINDING *and* CLAW FINDING, *we have*

$$Q_2(f) = \Theta((mn)^{1/3} + \sqrt{n} + \sqrt{m}) \tag{3.2}$$

[1]The query model they assumed is slightly different than the standard one we consider, but for many problems including this Claw-Finding one, the query complexities in these two different models are the same up to at most a log factor.

3.2.2 Review of Ambainis' search and the generic algorithm

We first review Ambainis' search algorithm for Unique k-Element Distinctness [12] as follows. The working state is a superposition of basis in the form of $|S, x_S, i\rangle$. Here S is a r-size subset of $[N]$ for some integer r; x_S contains the variable values x_j's for all $j \in S$; i is an index not in S. A basic tool used in the algorithm is a subroutine called **Quantum Walk** the following box of Algorithm 3.1.

```
Algorithm 3.1: Quantum Walk on S in A
Input: State $|S, x_S, i\rangle$ and $A$ with $S \subseteq A$, and $i \in A - S$. Suppose that $|S| = r, |A| = N$.
  1. $|S, x_S, i\rangle \to |S, x_S\rangle \left((-1 + \frac{2}{N-r})|i\rangle + \frac{2}{N-r} \sum_{j \in A-S-\{i\}} |j\rangle\right)$
  2. $|S, x_S, i\rangle \to |S \cup \{i\}, x_{S\cup\{i\}}, i\rangle$ by one query.
  3. $|S, x_S, i\rangle \to |S, x_S\rangle \left((-1 + \frac{2}{r+1})|i\rangle + \frac{2}{r+1} \sum_{j \in S-\{i\}} |j\rangle\right)$
  4. $|S, x_S, i\rangle \to |S - \{i\}, x_{S-\{i\}}, i\rangle$ by one query.
```

Intuitively, Step 1 and 3 make a diffusion of the position i in the set $A - S$ and S. The role of the diffusions will be clear in the **Algorithm 3.2** and the analysis of it by Lemma 12.

Mathematically, a key fact shown by Ambainis [12] is the following. Let $I = \{i_1, ..., i_k\}$ where $(i_1, ..., i_k)$ is the unique k-tuple of equal elements. Define a $(2k + 1)$-dimentional subspace

$$\tilde{H} = span\{|\psi_{j,l}\rangle : \ j = 0, ..., k; \ l = 0, 1; \ (j, l) \neq (k, 1)\} \tag{3.3}$$

where $|\psi_{j,l}\rangle$ is the uniform superposition of states $\{|S, x_S, i\rangle : |S| = r, \ i \in A - S, \ j = |S \cap I|, \ l = \lambda_{i \in I}\}$ (with $\lambda_\phi = 1$ if ϕ is true, and 0 otherwise). Then first, one step of **Quantum Walk** maps $\tilde{H}$ to $\tilde{H}$ itself. Second, the operation of **Quantum Walk**, when restricted on $\tilde{H}$, has $2k + 1$ eigenvalues, one of which is 1 and the corresponding eigenvalue is the starting state $|\psi_{start}\rangle$. The other $2k$ eigenvalues are in the form of $e^{\pm\theta_1 i}, ..., e^{\pm\theta_k i}$, where $\theta_j = (2\sqrt{j} + o(1))/\sqrt{r}$. Though the original k is supposed to be at least 2, we observe that the above fact also holds for case $k = 1$. This will be used in our proof of Theorem 11.

Using the following key lemma, Ambainis gave **Algorithm 3.2** for Unique k-Element Distinctness.

Lemma 12 (Ambainis [12]) *Let $\mathcal{H}$ be a finite dimensional Hilbert space and $|\psi_1\rangle, \ldots, |\psi_m\rangle$ be an orthonormal basis for $\mathcal{H}$. Let $|\psi_{good}\rangle$, $|\psi_{start}\rangle$ be two states in $\mathcal{H}$ which are superpositions of $|\psi_1\rangle, \ldots, |\psi_m\rangle$ with real amplitudes and $\langle\psi_{good}|\psi_{start}\rangle = \alpha$. Let U_1, U_2 be unitary transformations on $\mathcal{H}$ satisfying:*

1. *U_1 is the transformation that flips the phase on $|\psi_{good}\rangle$ (*i.e. *$U_1|\psi_{good}\rangle = -|\psi_{good}\rangle$) and leaves any state orthogonal to $|\psi_{good}\rangle$ unchanged.*

2. *U_2 is a transformation which is described by a real-valued $m * m$ matrix in the basis $|\psi_1\rangle, \ldots, |\psi_m\rangle$. Moreover, $U_2|\psi_{start}\rangle = |\psi_{start}\rangle$ and, if $|\psi\rangle$ is an eigenvector of U_2 perpendicular to $|\psi_{start}\rangle$, then $U_2|\psi\rangle = e^{i\theta}|\psi\rangle$ for $\theta \in [\epsilon, 2\pi - \epsilon]$.*

Then, there exists $t = O(\frac{1}{\alpha})$ such that $\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle = \Omega(1)$.

Algorithm 3.2: for Unique k-Element Distinctness
Input: $x_1, ..., x_N \in [M]$, with the promise that there exists at most one k-size set $I = \{i_1, ..., i_k\} \subseteq [N]$ such that $x_{i_1} = ... = x_{i_k}$.
Output: I and $x_I = \{x_{i_1}, ..., x_{i_k}\}$ if they exist; otherwise reject.

```
1. Set up the initial state $|\psi_{start}\rangle = \frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{S\subseteq[N],|S|=r,i\in[N]-S} |S, x_S, i\rangle$.
2. Do $\Theta((\frac{N}{r})^{k/2})$ times
    (a) Change the phase of those $S$ containing $I$ as a subset, i.e. $|S, x_S, i\rangle \to -|S, x_S, i\rangle$.
    (b) Do Quantum Walk on $S$ in $[N]$ for $\Theta(\sqrt{r})$ times.
3. Measure the resulting state and give the corresponding answer.
```

By Lemma 12, if the (unique) k-size subset I exists, then after Step 2, the state is close to $|\psi_{good}\rangle = \frac{1}{\sqrt{\binom{N-k}{r-k}(N-r)}} \sum_{|S|=r,\ I\subseteq S,\ i\in[N]-S} |S, x_S, i\rangle$, thus the algorithm can output $I = \{i_1, ..., i_k\}$ and $x_I = \{x_{i_1}, ..., x_{i_k}\}$ in Step 3 (with high probability). If such I does not exist, the state after Step 2 is still $|\psi_{start}\rangle$, and thus the algorithm rejects in Step 3. Therefore, by letting $r = N^{k/k+1}$, we have an algorithm using $O(N^{k/k+1})$ queries.

3.2.3 Proof of Theorem 11

We prove Theorem 11 in this section. For the upper bounds, we give **Algorithm 3.3**, which refines Ambainis' **Algorithm 3.2** by maintaining two sets of registers instead of one set.

```
Algorithm 3.3: for Unique (m,n) 2-Subset Finding
Input: $x_1, ..., x_N \in [M]$. $J_1, J_2 \subseteq [N]$, $|J_1| = m, |J_2| = n$. $R \subseteq [M] \times [M]$ such that there
is at most one $(x_{j_1}, x_{j_2}) \in R$ with $j_1 \in J_1$, $j_2 \in J_2$ and $j_1 \neq j_2$.
Output: The unique pair $(j_1, j_2)$ if it exists; otherwise reject.

  1. Set up the initial state $|\psi_{start}\rangle = \frac{1}{\sqrt{T}} \sum_{S_b \subseteq J_b, |S_b| = r_b, i_b \in J_b - S_b} |S_1, x_{S_1}, i_1, S_2, x_{S_2}, i_2\rangle$,
     where $T = \binom{m}{r_1}\binom{n}{r_2}(m - r_1)(n - r_2)$ and $b = 1, 2$.
  2. Do $\Theta(\sqrt{\frac{mn}{r_1 r_2}})$ times
     (a) Check whether the unique $(j_1, j_2)$ is in $S_1 \times S_2$. If yes, do the following phase
         flip: $|S_1, x_{S_1}, i_1, S_2, x_{S_2}, i_2\rangle \to -|S_1, x_{S_1}, i_1, S_2, x_{S_2}, i_2\rangle$.
     (b) Do Quantum Walk on $S_1$ in $J_1$ for $t_1 = \lceil \frac{\pi}{4}\sqrt{r_1} \rceil$ times.
         Do Quantum Walk on $S_2$ in $J_2$ for $t_2 = \lceil \frac{\pi}{8}\sqrt{r_2} \rceil$ times.
  3. Measure the resulting state and give the corresponding answer.
```

The following theorem actually shows the upper bound in Theorem 1.

Theorem 13 *Algorithm 3.3 outputs the desired results correctly, and we can pick r_1, r_2 to make number of queries be*

$$\begin{cases} O((mn)^{1/3}) & \text{if } \sqrt{n} \le m \le n^2 \qquad (\text{by letting } r_1 = r_2 = (mn)^{1/3}) \\ O(\sqrt{n}) & \text{if } m < \sqrt{n} \qquad (\text{by letting } r_1 = m, r_2 \in [m, (mn)^{1/3}]) \\ O(\sqrt{m}) & \text{if } m > n^2 \qquad (\text{by letting } r_1 \in [n, (mn)^{1/3}], r_2 = n) \end{cases} \tag{3.4}$$

Proof Correctness: First, if there is no desired pair, then the algorithm actually does nothing, so the state after Step 2 is still $|\psi_{start}\rangle$. Thus in Step 3, we cannot find the desired pair after the measurement, and we will reject.

On the other side, if there is the pair, we shall use Lemma 12 to show that we can find it. Suppose $(j_1, j_2) \in J_1 \times J_2$ is the desired pair. First, define $\tilde{H}_1$ as in (3.3), with $|\psi_{j,l}\rangle$

being the uniform superposition of states

$$\{|S_1, x_{S_1}, i_1\rangle : S_1 \subseteq J_1, |S_1| = r_1,\ i_1 \in J_1 - S_1,\ j = \lambda_{j_1 \in S_1},\ l = \lambda_{i_1 = j_1}\}. \tag{3.5}$$

Note that it is exactly the "$k = 1$" case of (3.3), so W_1, the operator of **Quantum Walk** on S_1 in J_1, when restricted on $\tilde{H}_1$, has 3 eigenvalues. One of the eigenvalues is 1, and the corresponding eigenvector is

$$|\psi_{start,1}\rangle = \frac{1}{\sqrt{\binom{m}{r_1}(m - r_1)}} \sum_{S_1 \subseteq J_1, |S_1| = r_1, i_1 \in J_1 - S_1} |S_1, x_{S_1}, i_1\rangle.$$

The other two eigenvalues are $e^{\pm i\theta_1}$, and $\theta_1 = (2 + o(1))/\sqrt{r_1}$. Therefore, $W_1^{t_1}$ has 3 eigenvalues: 1 (with the eigenvector $|\psi_{start,1}\rangle$) and $e^{\pm i\theta_1'}$ where $\theta_1' = \frac{\pi}{2} + o(1)$.

$\tilde{H}_2$ is defined symmetrically, as well as W_2, $|\psi_{start,2}\rangle$ and θ_2. As a result, $W_2^{t_2}$ has 3 eigenvalues: 1 (with the eigenvector $|\psi_{start,2}\rangle$) and $e^{\pm i\theta_2'}$ where $\theta_2' = \frac{\pi}{4} + o(1)$. The whole step 2(b) restricted on $\tilde{H}_1 \otimes \tilde{H}_2$ is the operation $W = (I_1 \otimes W_2)(W_1 \otimes I_2)$. Now note that the eigenvalues of W are given by

$$\{\lambda \cdot \mu : \lambda \text{ is an eigenvalue of } W_1 \text{ on } \tilde{H}_1, \text{ and } \mu \text{ is an eigenvalue of } W_2 \text{ on } \tilde{H}_2\}.$$

Therefore, W has 9 eigenvalues: $\{e^{i(b_1\theta_1' + b_2\theta_2')} : b_1, b_2 \in \{-1, 0, 1\}\}$. It is easy to check that one of these eigenvalues is 1, and the corresponding eigenvector is $|\psi_{start,1}\rangle \otimes |\psi_{start,2}\rangle$, which is exactly the $|\psi_{start}\rangle$ in Algorithm 3.3. All the other 8 eigenvalues are in the form of $e^{\pm i\theta}$, for some $\theta \in [\pi/4 - o(1), 2\pi - \pi/4 + o(1)]$.

Finally, we calculate $\alpha = \langle\psi_{start}|\psi_{good}\rangle$:

$$\alpha = \sqrt{\mathbf{Pr}_{|S_1| = r_1, |S_2| = r_2}[(j_1, j_2) \in S_1 \times S_2]} = \Theta\left(\sqrt{\frac{r_1 r_2}{mn}}\right). \tag{3.6}$$

So the number of iterations in Step 2 is $1/\alpha = \Theta(\sqrt{\frac{mn}{r_1 r_2}})$ and the correctness holds by Lemma 12.

It is easy to verify that the number of queries used is

$$O\left(r_1 + r_2 + \sqrt{\frac{mn}{r_1 r_2}}(\sqrt{r_1} + \sqrt{r_2})\right) = O\left(r_1 + r_2 + \frac{\sqrt{mn}}{\sqrt{r_1}} + \frac{\sqrt{mn}}{\sqrt{r_2}}\right). \tag{3.7}$$

Now we need to minimize it, with restrictions $r_1 \leq m$ (because S_1 is a subset of J_1) and $r_2 \leq n$. For the $(r_1 + \frac{\sqrt{mn}}{\sqrt{r_1}})$ part, it is not hard to see that if $m \geq \sqrt{n}$ then $\min_{r_1 \leq m}(r_1 + \frac{\sqrt{mn}}{\sqrt{r_1}}) = (mn)^{1/3}$ and the minimum is achieved when $r_1 = (mn)^{1/3}$; otherwise $\min_{r_1 \leq m}(r_1 + \frac{\sqrt{mn}}{\sqrt{r_1}}) = m + \sqrt{n}$ and the minimum is obtained when $r_1 = m$. Analyze the $r_2 + \sqrt{mn}/\sqrt{r_2}$ part similarly, and we can get the conclusion of the theorem. □

Next we prove the lower bound part in Theorem 11. Note that since Claw-Finding is a special case of (m, n) 2-Subset Finding, it is enough to show the lower bound for Q_2(Claw-Finding).

Proof (of the lower bound in Theorem 11) It is sufficient to prove the lower bound of $\Omega((mn)^{1/3})$. We will show it by a reduction to the 2-Collision problem, which is to distinguish whether a function $f : [N] \to [N]$ is one-to-one or two-to-one. This problem is shown by Aaronson, Shi [5] (and Ambainis [11] for small range) to have $\Omega(N^{1/3})$ lower bound of quantum query complexity. Assume that we can solve (m, n) 2-Subset Finding with $o((mn)^{1/3})$ queries, then we can have an $o(N^{1/3})$ algorithm for the 2-Collision problem as follows. Let $f : [N] \to [N]$ be a function, where $N = mn$, and we are to decide whether it is one-to-one or two-to-one. First pick a random set $S_1 \subseteq [N]$ of size m and then pick another random set $S_2 \subseteq [N] - S_1$ of size n. If f is one-to-one, then $f(i_1) \neq f(i_2)$ for any $i_1 \in S_1$ and $i_2 \in S_2$, since $S_1 \cap S_2 = \emptyset$. On the other hand, if f is two-to-one, then by a standard probability calculation we know that with constant probability there will be $i_1 \in S_1$ and $i_2 \in S_2$ such that $f(i_1) = f(i_2)$. Therefore, whether f is two-to-one or one-to-one is, up to a constant probability, equivalent to whether there are $i_1 \in S_1$ and $i_2 \in S_2$ such that $f(i_1) = f(i_2)$, which can be decided with $o((mn)^{1/3}) = o(N^{1/3})$ queries, by our assumption. This contradicts to the $\Omega(N^{1/3})$ lower bound of 2-Collision [5, 11], so

$$Q_2(\textsc{Unique } (m,n) \text{ 2-}\textsc{Subset Finding}) = \Omega\left((mn)^{1/3}\right). \tag{3.8}$$

□

Chapter 4

Randomized and Quantum Query Complexities for Local Search on Different Graphs

The results in this chapter are from paper [80].

4.1 Introduction

Many important combinatorial optimization problems arising in both theory and practice are **NP**-hard, which forces people to resort to heuristic searches in practice. One popular approach is Local Search, by which one first defines a *neighborhood structure*, then finds a solution that is locally optimal with respect to this neighborhood structure. In the past two decades, Local Search approach has been extensively developed and "has reinforced its position as a standard approach in combinatorial optimization" in practice [1]. Besides the practical applications, the problem also has many connections to the complexity theory, especially to the complexity classes **PLS** [1] and **TFNP** [2] . For example, the 2SAT-FLIP problem is Local Search on the Boolean hypercube graph $\{0,1\}^n$, with the objective function

[1]Polynomial Local Search, introduced by Johnson, Papadimitriou, and Yannakakis [41].

[2]The family of total function problems, introduced by Megiddo and Papadimitriou [51].

being the sum of the weights of the clauses that the truth assignment $x \in \{0,1\}^n$ satisfies. This problem is complete in **PLS**, implying that the Boolean hypercube $\{0,1\}^n$ has a central position in the studies of Local Search. Local Search is also related to physical systems including folding proteins and to quantum adiabatic algorithms [2]. We refer readers to the papers [2, 56, 63] for more discussions and the book [6] for a comprehensive introduction.

Precisely, Local Search on an undirected graph $G = (V, E)$ is defined as follows. Given a function $f : V \to \mathbb{N}$, find a vertex $v \in V$ such that $f(v) \leq f(w)$ for all neighbors w of v. A class of *generic algorithms* that has been widely used in practice is as follows: we first set out with an initial point $v \in V$, then repeatedly search a better/best neighbor until it reaches a local minimum. Though empirically this class of algorithms works very well in most applications, relatively few theoretical results are known about how good the generic algorithms are, especially for the randomized (and quantum) algorithms.

Among models for theoretical studies, the query model has drawn much attention [2, 7, 8, 47, 48, 63]. In this model, f is given by a black-box, *i.e.* $f(v)$ can be accessed by querying v. We only care about the number of queries made, and all other computations are free. If we are allowed to toss coins to decide the next query, then we have a randomized query algorithm. If we are allowed use quantum mechanics to query all the positions (and get corresponding answers) in superposition, then we have a quantum query algorithm. The deterministic, randomized and quantum query complexities are the minimum numbers of queries needed to compute the function by a deterministic, randomized and quantum query algorithm, respectively. We use $RLS(G)$ and $QLS(G)$ to denote the randomized and quantum query complexities of Local Search on graph G, respectively. Previous upper bounds on a general N-vertex graph G are $RLS(G) = O(\sqrt{N\delta})$ by Aldous [7] and $QLS(G) = O(N^{1/3}\delta^{1/6})$ by Aaronson [2], where δ is the maximum degree of G. Both algorithms actually fall into the category of generic algorithms mentioned above, with the initial point picked as a best one over a certain number of random samples. Immediately, two questions can be asked:

1. On what graphs are these simple algorithms optimal?

2. For other graphs, what better algorithms can we have?

Clearly the first one is a lower bound question and the second one is an upper bound question.

Previously for lower bounds, Aaronson [2] showed the following results on two special classes of graphs: the Boolean hypercube $\{0,1\}^n$ and the constant dimensional grid $[n]^d$:

$$RLS(\{0,1\}^n) = \Omega(2^{n/2}/n^2), \quad QLS(\{0,1\}^n) = \Omega(2^{n/4}/n); \tag{4.1}$$

$$RLS([n]^d) = \Omega(n^{d/2-1}/\log n), \quad QLS([n]^d) = \Omega(n^{d/4-1/2}/\sqrt{\log n}). \tag{4.2}$$

It has also been shown that $QLS([n]^2) = \Omega(n^{1/4})$ by Santha and Szegedy in [63], besides their main result that the deterministic and the quantum query complexities of Local Search on any graph are polynomially related. However, the question

3. What are the true values of QLS and RLS on $\{0,1\}^n$ and $[n]^d$?

remains an open problem, explicitly stated in an earlier version of [2] and also (partially) in [63].

In this section, we answer questions 1 and 2 for large classes of graphs by giving both new lower and upper bound techniques for randomized and quantum query algorithms. As a consequence, we completely solve question 3, except for a few small d's where our new bounds also significantly improve the old ones.

Our lower bound technique works for any graph that contains a product graph as a subgraph. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their product $G_1 \times G_2$ is the graph $G = (V, E)$ where $V = V_1 \times V_2$ and

$$E = \{(v_1 \otimes v_2, v_1' \otimes v_2) : (v_1, v_1') \in E_1, v_2 \in V_2\} \cup \{(v_1 \otimes v_2, v_1 \otimes v_2') : (v_2, v_2') \in E_2, v_1 \in V_1\} \tag{4.3}$$

We will also use the notion of random walk on graphs to state the theorem. Given a graph $G = (V, E)$, a random walk is a mapping $W : V \to 2^V$ where $W(u) \subseteq \{u\} \cup \{v : (u, v) \in E\}$. Intuitively, at each step the random walk W goes from the current vertex u to a uniformly random vertex in $W(u)$. The walk W is *regular* if $|W(u)| = c$ for each $u \in V$. Denote by

$p(u, v, t)$ the probability that the random walk starting at u is at v after exactly t steps. Let $p_t = \max_{u,v} p(u, v, t)$. The following theorem is a special case of the general one (Theorem 20) in Section 4.3.

Theorem 14 *Suppose G contains the product graph $G_1 \times G_2$ as a subgraph, and L is the length of the longest self-avoiding path in G_2. Let $T = \lfloor L/2 \rfloor$, then for any regular random walk W on G_1, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t}\right), \quad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}}\right).$$

The proof uses the quantum adversary method in its version Theorem 8. Recall that recently Spalek and Szegedy showed that the various versions of the quantum adversary method are equivalent in power [69]. However, in proving a particular problem, some of the methods might be easier to apply than the others. In our case, the technique in [78] turns out to work very well. Our proofs for the randomized lower bounds will use the relational adversary method, which was proposed by Aaronson [2] inspired by the quantum adversary method.

Both the quantum adversary method and the relational adversary method are parameterized by input sets and weight functions of input pairs. While previous works [2, 63] also use random walks on graphs, a key innovation that distinguishes our work from previous ones and yields better lower bounds is that we decompose the graph into two parts, the tensor product of which is the original graph. We perform the random walk only in one part, and perform a simple one-way walk in a self-avoiding path in the other part, which serves as a "clock" to record the number of steps taken by the random walk in the first part. The tensor product of these two walks is a random path in the original graph. A big advantage of adding a clock is that the "passing probability", the probability that the random path *passes* a vertex v *within* T steps, is now the "hitting probability", the probability that the random walk in the first graph *hits* v *after* exactly t steps, because the time elapses one-way and never comes back. The fact that the hitting probability is much smaller than the passing probability enables us to achieve the better lower bounds. Another advantage of the clock is that since the walk in the second part is self-avoiding, the resulting random path in the original graph is self-avoiding too, which makes the analysis much easier.

Applying it to the two graphs $\{0,1\}^n$ and $[n]^d$, we improve previous results and show tight bounds on both RLS and QLS (except for a few cases in the low dimensional grids).

Theorem 15

$$RLS(\{0,1\}^n) = \Theta(2^{n/2}n^{1/2}), \qquad QLS(\{0,1\}^n) = \Theta(2^{n/3}n^{1/6}). \tag{4.4}$$

Theorem 16

$$RLS([n]^d) = \begin{cases} \Theta(n^{d/2}) & \textit{if } d \geq 4, \\ \Omega((n^3/\log n)^{1/2}) & \textit{if } d = 3, \\ \Omega(n^{2/3}) & \textit{if } d = 2. \end{cases} \quad QLS([n]^d) = \begin{cases} \Theta(n^{d/3}) & \textit{if } d \geq 6, \\ \Omega((n^5/\log n)^{1/3}) & \textit{if } d = 5, \\ \Omega(n^{6/5}) & \textit{if } d = 4, \\ \Omega(n^{3/4}) & \textit{if } d = 3, \\ \Omega(n^{2/5}) & \textit{if } d = 2. \end{cases} \tag{4.5}$$

It is worth noting that to apply Theorem 14, we need to know not only the mixing time of the random walk in G_1, but also *its behavior before mixing*. So the applications are not simply using standard upper bounds on the mixing times, but involve heavy analysis on the whole mixing processes.

When proving Theorem 16 by Theorem 14, one difficulty arises: to decompose the grid $[n]^d$ into two parts $[n]^m$ and $[n]^{d-m}$, we implicitly require that m is an integer. This gives us lower bounds weaker than Theorem 16, especially for low dimension cases. We get around this problem by cutting one of the m dimensions into many blocks, and use different block to distinguish different time windows. Between adjacent blocks are pairwise disjoint path segments, which thus thread all the blocks into a very long one. Using this technique, we can apply Theorem 14 for any read-number dimension $m \leq d-1$.

In the second part of the chapter, we consider upper bounds for Local Search. While the generic algorithms [2, 7] are simple and proven to be optimal for many graphs such as the ones mentioned above, they are far from optimal for some other graphs. For example, it is not hard to see an $O(\log N)$ *deterministic* algorithm for the line graph G. Therefore, a natural question is to characterize those graphs on which Local Search is easy. It turns out that the expansion speed plays a key role. For a graph $G = (V, E)$, the distance $l(u, v)$ between two vertices u and v is the length of the shortest path connecting them. (Here the length of a path is the number of edges on the path.) Let $c(k) = \max_{v \in V} |\{u : l(u, v) \le k\}|$. Clearly, the smaller $c(k)$ is, the more slowly the graph expands. (Actually $c(k)$ is an upper bound of the standard definition of the expanding speed.) We say a graph is of polynomial growth if $c(k) = O(k^\alpha)$ for some constant $\alpha \ge 1$. As a special case of Theorem 26 in Section 4.5, the following upper bounds for the graphs of polynomial growth.

Theorem 17 *If* $c(k) = O(k^\alpha)$ *for some constant* $\alpha \ge 1$*, then*

$$RLS(G) = \begin{cases} O\left(d^{\alpha-1} \log\log d\right) & \text{if } \alpha > 1, \\ O(\log d \log\log d) & \text{if } \alpha = 1. \end{cases} \quad QLS(G) = \begin{cases} O\left(d^{\frac{\alpha-1}{2}} (\log\log d)^{1.5}\right) & \text{if } \alpha > 1, \\ O(\log d \log\log d) & \text{if } \alpha = 1. \end{cases} \tag{4.6}$$

where d *is the diameter of the graph* G*.*

As a special case, on the line graph we get $\alpha = 1$ and hence $RLS = O(\log n \log\log n)$, which helps to explain why Local Search on the line graph is easy. Also, it immediately gives a new upper bound for $QLS([n]^2)$ as follows. Together with Theorem 16, this implies that Local Search on grids exhibits different properties in low dimensions.

Theorem 18 $QLS([n]^2) = O(\sqrt{n}(\log\log n)^{1.5})$

Other related results. After the preliminary version of this paper appeared, Verhoeven independently showed an upper bound in terms of the genus of the graph [74], giving an $O(\sqrt{n} \log\log n)$ quantum algorithm for $[n]^2$. There is also an unpublished result on $QLS(\{0,1\}^n)$: it is mentioned in [2] that Ambainis showed $QLS(\{0,1\}^n) = \Omega(2^{n/3}/n^{O(1)})$. 3

3 Another unpublished result was mentioned in [63] that Verhoeven showed $RLS([n]^2) = \Omega(n^{1-\delta})$ for any constant $\delta > 0$. But according to Santha (personal communication), one of the two authors of [63], the proof was never written up and this question should be considered now to be still open.

4.2 Preliminaries and notations

For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we say that G_1 is a subgraph of G_2 if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. Clearly, any local optimum in G_2 is also a local optimum in G_1 (but not the other way around in general), therefore any lower bound for G_1 is also a lower bound for G_2.

We let the variables $v_1 \otimes v_2$ range over the set $V_1 \times V_2$. There are various ways to define a product graph $G_1 \times G_2 = (V_1 \times V_2, E)$ by different choices of E. Three possibilities are

1. $E = \{(v_1 \otimes v_2, v_1' \otimes v_2) : (v_1, v_1') \in E_1, v_2 \in V_2\} \cup \{(v_1 \otimes v_2, v_1 \otimes v_2') : (v_2, v_2') \in E_2, v_1 \in V_1\}$;
2. $E' = \{(v_1 \otimes v_2, v_1' \otimes v_2') : (v_1, v_1') \in E_1 \cup I_{V_1}$ and $(v_2, v_2') \in E_2 \cup I_{V_2}\} - I_{V_1 \times V_2}$, where $I_V = \{(v, v) : v \in V\}$;
3. $E'' = \{(v_1 \otimes v_2, v_1' \otimes v_2') : (v_1, v_1') \in E_1 \cup I_{V_1}$ or $(v_2, v_2') \in E_2 \cup I_{V_2}\} - I_{V_1 \times V_2}$.

It is clear that $E \subseteq E' \subseteq E''$, and our lower bound theorem will use the first definition E, making the theorem as general as possible.

A path X in a graph $G = (V, E)$ is a sequence $(v_1, ..., v_l)$ of vertices such that for any pair (v_i, v_{i+1}) of vertices, either $v_i = v_{i+1}$ or $(v_i, v_{i+1}) \in E$. We use $set(X)$ to denote the set of distinct vertices on path X. A path is self-avoiding if v_1, ..., v_l are all distinct. The length of a path $(v_1, ..., v_l)$ is $l - 1$. For two vertices $u, v \in V$, the distance $l_G(u, v)$ is the length of a shortest path from u to v. The subscript G may be omitted if no confusion is caused.

The (k, l)-hypercube $G_{k,l} = (V, E)$ where $V = [k]^l$ and whose edge set is $E = \{(u, v) : \exists i \in \{0, ..., l-1\},\ suchthat\ |u_i - v_i| = 1, \text{ and } u_j = v_j,\ \forall j \neq i\}$. Sometimes we abuse notation by using $[k]^l$ to denote $G_{k,l}$. Note that both the Boolean hypercube and the constant dimension grid are special hypercubes.[4]

In an N-vertex graph $G = (V, E)$, a Hamilton path is a path $X = (v_1, ..., v_N)$ such that $(v_i, v_{i+1}) \in E$ for any $i \in [N-1]$ and $set(X) = V$. It is easy to check by induction

[4]Here we identify the Boolean hypercube $\{0, 1\}^n$ and $G_{2,n}$ since they are isomorphic.

that every hypercube $[k]^l$ has a Hamilton path. Actually, for $l = 1$, $[k]$ has a Hamilton path $(1, ..., k)$. Now suppose $[k]^l$ has a Hamilton path P; then a Hamilton path for $[k]^{l+1}$ can be constructed as follows. First fix the last coordinate to be 1 and go through P, then change the last coordinate to be 2 and go through P in the reverse order, and then change the last coordinate to be 3 and go through P, and so on. For each (k, l), let $HamPath_{k,l} = (v_1, ..., v_N)$ be the Hamilton path constructed as above (where $N = k^l$), and we define the successor function $H_{k,l}(v_i) = v_{i+1}$ for $i \in [N-1]$.

4.3 Lower bounds for Local Search on product graphs

In this section we prove a theorem which is stronger than Theorem 14 due to a relaxation on the conditions of the random walk. Suppose we are given a graph $G = (V, E)$, a starting vertex v_0 and an assignment $W : V \times \mathbb{N} \to 2^V$ such that for each $u \in V$ and $t \in \mathbb{N}$, it holds that $W(u,t) \subseteq \{u\} \cup \{v : (u,v) \in E\}$ and that $|W(u,t)| = c_t$ for some function c of t. Intuitively, W gives the candidates that the walk goes to for the next step, and the random walk (G, v_0, W) on graph G proceeds as follows. It starts at v_0, and at step $t \in \mathbb{N}$, it goes from the current vertex v_{t-1} to a uniformly random vertex in $W(v_{t-1}, t)$. We say *a path* $(v_0, v_1, ..., v_T)$ *is generated by the random walk* if $v_t \in W(v_{t-1}, t)$ for all $t \in [T]$. Denote by $p(u, t_1, v, t_2)$ the probability that the random walk is at v after step t_2 under the condition that the walk is at u after step t_1. Let $p_t = \max_{u,v,t_1,t_2:\ t_2-t_1=t} p(u, t_1, v, t_2)$. For $(u, u') \in E$, let $q(u, u', t_1, v, t_2)$ be the probability that the walk is at v after step t_2, under the conditions that 1) the walk is at u after step t_1, and 2) the walk does not go to u' at step $t_1 + 1$. The following lemma on the relation of the two probabilities is obvious.

Lemma 19 *If* $|W(u, t_1 + 1)| > 1$*, then* $q(u, u', t_1, v, t_2) \le 2p(u, t_1, v, t_2)$.

Proof By considering the two cases of the step $t_1 + 1$ (going to u' or not), we have

$$p(u, t_1, v, t_2) = \frac{1}{|W(u, t_1 + 1)|} p(u', t_1 + 1, v, t_2) + \left(1 - \frac{1}{|W(u, t_1 + 1)|}\right) q(u, u', t_1, v, t_2). \tag{4.7}$$

Thus

$$q(u,u',t_1,v,t_2) \le p(u,t_1,v,t_2)/\left(1-\frac{1}{|W(u,t_1+1)|}\right) \le 2p(u,t_1,v,t_2). \tag{4.8}$$

□

Theorem 20 *Suppose G contains $G^w \times G^c$ (for two arbitrary graphs G^w and G^c) as a subgraph, and L is the length of the longest self-avoiding path in G^c. Let $T = \lfloor L/2 \rfloor$, then for the random walk (G^w, v_0^w, W) on G^w, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^T p_t}\right), \qquad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^T \sqrt{p_t}}\right). \tag{4.9}$$

Proof Without loss of generality, we assume $G = G^w \times G^c$, as Local Search on a subgraph is no harder than Local Search on the original graph. We shall construct a random walk on G by the random walk (G^w, v_0^w, W) on G^w and a simple one-way walk on G^c. Starting from some fixed vertex in G, the walk is proceeded by one step of walk in G^w followed by two steps of walk in G^c. (We perform *two* steps of walk in G^c mainly for some technical reasons, and this is where the factor of 2 in definition $T = \lfloor L/2 \rfloor$ comes from.) Precisely, fix a self-avoiding path $(z_{0,0}^c, z_{1,0}^c, z_{1,1}^c, z_{2,1}^c, z_{2,2}^c, ..., z_{T,T-1}^c, z_{T,T}^c)$ of length $2T$ in G^c. Let the set P contain all the paths $X = (x_0^w \otimes z_{0,0}^c, x_1^w \otimes z_{0,0}^c, x_1^w \otimes z_{1,0}^c, x_1^w \otimes z_{1,1}^c, ..., x_T^w \otimes z_{T-1,T-1}^c, x_T^w \otimes z_{T,T-1}^c, x_T^w \otimes z_{T,T}^c)$ in G such that $x_0^w = v_0^w$ and $(x_0^w, x_1^w, ..., x_T^w)$ is a path generated by the random walk (G^w, v_0^w, W). Define a problem $\textsc{Path}_P$: given a path $X \in P$, find the end point $x_T^w \otimes z_{T,T}^c$. To access X, we can ask whether $v \in set(X)$ for any vertex $v \in V$, and an oracle O will give us the Yes/No answer.[5] The following claim says that the $\textsc{Path}_P$ problem is not much harder than the Local Search problem.

Claim 21 $R_2(\textsc{Path}_P) \le 2RLS(G), \quad Q_2(\textsc{Path}_P) \le 2QLS(G).$

[5]Note that it is actually an oracle for the following function $g : \{0,1\}^n \to \{0,1\}$, with $g(x) = 1$ if and only if $x \in set(X)$. So strictly speaking, an input of $\textsc{Path}_P$ should be specified as $set(X)$ rather than X, because in general, it is possible that $X \neq Y$ but $set(X) = set(Y)$. For our problem, however, it is easy to check that for any $X, Y \in P$, it holds that $X = Y \Leftrightarrow set(X) = set(Y)$. Indeed, if $X \neq Y$, suppose the first diverging place is k, *i.e.* $x_{k-1}^w = y_{k-1}^w$, but $x_k^w \neq y_k^w$. Then Y will never pass $x_k^w \otimes z_{k,k-1}^c$ because the clock immediately ticks and the time always advances forward. (Or more rigorously, the only point that Y passes through $z_{k,k-1}^c$ is $y_k^w \otimes z_{k,k-1}^c$. Since $y_k^w \neq x_k^w$, $x_k^w \otimes z_{k,k-1}^c \notin set(Y)$.)

Proof Suppose we have an Q-query randomized or quantum algorithm $\mathcal{A}$ for Local Search, we shall give a $2Q$ corresponding algorithm $\mathcal{B}$ for $\textsc{Path}_P$. For any path $X \in P$, we define a function f_X essentially in the same way as Aaronson did in [2]: for each vertex $v \in G$, let

$$f_X(v) = \begin{cases} l_G(v, x_0^w \otimes z_{0,0}^c) + 3T & \text{if } v \notin set(X) \\ 3(T-k) & \text{if } v = x_k^w \otimes z_{k,k}^c \\ 3(T-k)-1 & \text{if } v = x_{k+1}^w \otimes z_{k,k}^c \neq x_k^w \otimes z_{k,k}^c \\ 3(T-k)-2 & \text{if } v = x_{k+1}^w \otimes z_{k+1,k}^c \end{cases}. \tag{4.10}$$

It is easy to verify that the only local minimum is $x_T^w \otimes z_{T,T}^c$.

Given an oracle O and an input X of the $\textsc{Path}$ problem, $\mathcal{B}$ simulates $\mathcal{A}$ to find the local minimum of f_X, which is also the end point of X. Whenever $\mathcal{A}$ needs to make a query on v to get $f_X(v)$, $\mathcal{B}$ asks O whether $v \in set(X)$. If $v \notin set(X)$, then $f_X(v) = l_G(v, x_0^w \otimes z_{0,0}^c) + 3T$; otherwise, $v = x^w \otimes z_{k+1,k}^c$ or $v = x^w \otimes z_{k,k}^c$ for some $x^w \in V^w$ and k. Note that k is known for any given vertex v. So if $v = x^w \otimes z_{k+1,k}^c$, then $x^w = x_{k+1}^w$ and thus $f_X(v) = 3(T-k)-2$. Now consider the case that $v = x^w \otimes z_{k,k}^c$. If $k = 0$, then let $f_X(v) = 3T$ if $v = x_0^w \otimes z_{0,0}^c$ and $f_X(v) = 3T - 1$ otherwise. If $k \geq 1$, then $\mathcal{B}$ asks O whether $x^w \otimes z_{k,k-1}^c \in set(X)$. If yes, then $v = x_k^w \otimes z_{k,k}^c$ and thus $f_X(v) = 3(T-k)$; if no, then $v = x_{k+1}^w \otimes z_{k,k}^c \neq x_k^w \otimes z_{k,k}^c$ and thus $f_X(v) = 3(T-k) - 1$. Therefore, at most 2 queries on O can simulate one query on f_X, so we have a $2Q$ algorithm for $\textsc{Path}_P$ in both randomized and quantum cases. $\square$

(Continue the proof of Theorem 20) By the claim, it is sufficient to prove lower bounds for $\textsc{Path}_P$. We define a relation R_P as follows.

$$R_P = \{(X,Y) : X \in P,\ Y \in P,\ X \text{ and } Y \text{ has different end points}\}. \tag{4.11}$$

For any pair $(X,Y) \in R_P$, where $X = (x_0^w \otimes z_{0,0}^c, x_1^w \otimes z_{0,0}^c, x_1^w \otimes z_{1,0}^c, x_1^w \otimes z_{1,1}^c, ..., x_T^w \otimes z_{T-1,T-1}^c, x_T^w \otimes z_{T,T-1}^c, x_T^w \otimes z_{T,T}^c)$ and $Y = (y_0^w \otimes z_{0,0}^c, y_1^w \otimes z_{0,0}^c, y_1^w \otimes z_{1,0}^c, y_1^w \otimes z_{1,1}^c, ..., y_T^w \otimes z_{T-1,T-1}^c, y_T^w \otimes z_{T,T-1}^c, y_T^w \otimes z_{T,T}^c)$, we write $X \wedge Y = k$ if $x_0^w = y_0^w$, ..., $x_{k-1}^w = y_{k-1}^w$ but $x_k^w \neq y_k^w$. Intuitively, $X \wedge Y = k$ if k is the place that the paths X and Y diverge for the first time. Note that if $X \wedge Y = k$, then $x_k^w, y_k^w \in W(x_{k-1}^w, k)$ and thus $|W(x_{k-1}^w, k)| \geq 2$. By Lemma 19, this implies that $q(x_{k-1}^w, x_k^w, k-1, v^w, j) \leq 2p_{j-k+1}$.

We choose the weight functions in Theorem 8 by letting

$$\begin{aligned} w(X,Y) &= 1/|\{Y' \in P : Y' \wedge X = k\}| && (4.12)\\ &= 1/|\{X' \in P : X' \wedge Y = k\}| && (4.13)\\ &= 1/[(c_k - 1)c_{k+1}...c_T]. && (4.14) \end{aligned}$$

To calculate $w_X = \sum_{Y':(X,Y') \in R_P} w(X,Y')$, we group those Y' that diverge from X at the same place k':

$$\begin{aligned} w_X &= \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y') \in R_P \\ X \wedge Y' = k'}} w(X,Y') && (4.15)\\ &= \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y') \in R_P \\ X \wedge Y' = k'}} \frac{1}{|\{Y' \in P : Y' \wedge X = k'\}|} && (4.16)\\ &= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(X,Y') \in R_P | Y' \wedge X = k'] && (4.17)\\ &= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(y')_T^w \neq x_T^w | Y' \wedge X = k'] && (4.18) \end{aligned}$$

Here equality (4.17) holds because all paths diverging from X firstly at k' have the same probability $1/[(c_{k'} - 1)c_{k'}...c_T]$. Also note that the probability in the last equality is nothing but $1 - q(x_{k'-1}^w, x_{k'}^w, k' - 1, x_T^w, T)$, which is at least $1 - 2p_{T-k'+1}$. So we have

$$w_X \geq T - 2\sum_{k'=1}^{T} p_{T-k'+1} = T - 2\sum_{t=1}^{T} p_t. \qquad (4.19)$$

And similarly, we have $w_Y \geq T - 2\sum_{t=1}^{T} p_t$ too.

Now we describe $u(X,Y,i)$ and $v(X,Y,i)$, where i is a point $x_{j+r}^w \otimes z_{j+s,j}^c \in set(X) - set(Y)$ or $y_{j+r}^w \otimes z_{j+s,j}^c \in set(Y) - set(X)$. Here $(r,s) \in \{(0,0),(1,0),(1,1)\}$, and $0 \leq j \leq j + r \leq T$. Let

$$u(X,Y,x_{j+r}^w \otimes z_{j+s,j}^c) = a_{k,j,r,s} w(X,Y), \quad u(X,Y,y_{j+r}^w \otimes z_{j+s,j}^c) = b_{k,j,r,s} w(X,Y), \quad (4.20)$$

$$v(X,Y,x_{j+r}^w \otimes z_{j+s,j}^c) = b_{k,j,r,s} w(X,Y), \quad v(X,Y,y_{j+r}^w \otimes z_{j+s,j}^c) = a_{k,j,r,s} w(X,Y), \quad (4.21)$$

where $a_{k,j,r,s}$ and $b_{k,j,r,s}$ will be given later (satisfying $a_{k,j,r,s}b_{k,j,r,s} = 1$, which makes u, v, w really a weight scheme). We shall calculate $u_{X,i}$ and $v_{Y,i}$ for $i = x^w_{j+r} \otimes z^c_{j+s,j} \in set(X) - set(Y)$; the other case $i = y^w_{j+r} \otimes z^c_{j+s,j}$ is symmetric. Note that if $x^w_{j+r} \otimes z^c_{j+s,j} \notin set(Y')$ and $X \wedge Y' = k'$, then $k' \le j + r$.

$$u_{X,x^w_{j+r}\otimes z^c_{j+s,j}} = \sum_{k'=1}^{j+r} \sum_{\substack{Y':(X,Y')\in R_P, X\wedge Y'=k' \\ x^w_{j+r}\otimes z^c_{j+s,j}\notin set(Y')}} a_{k',j,r,s} w(X,Y') \tag{4.22}$$

$$\le \sum_{k'=1}^{j+r} \sum_{Y':X\wedge Y'=k'} a_{k',j,r,s} w(X,Y') \tag{4.23}$$

$$= \sum_{k'=1}^{j+r} a_{k',j,r,s} \tag{4.24}$$

The computation for $v_{Y,x^w_{j+r}\otimes z^c_{j+s,j}}$ is a little more complicated. By definition,

$$v_{Y,x^w_{j+r}\otimes z^c_{j+s,j}} = \sum_{k'=1}^{j+r} \sum_{\substack{X':(X',Y)\in R_P,\ X'\wedge Y=k', \\ x^w_{j+r}\otimes z^c_{j+s,j}\in set(X')}} b_{k',j,r,s} w(X',Y) \tag{4.25}$$

$$\le \sum_{k'=1}^{j+r} \sum_{\substack{X':X'\wedge Y=k', \\ x^w_{j+r}\otimes z^c_{j+s,j}\in set(X')}} b_{k',j,r,s} w(X',Y) \tag{4.26}$$

$$= \sum_{k'=1}^{j+r} b_{k',j,r,s} \mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X')|X' \wedge Y = k'] \tag{4.27}$$

We can see that by adding the clock, the passing probability $\mathbf{Pr}_{X'}[x^w_{j+r}\otimes z^c_{j+s,j} \in set(X')|X' \wedge Y = k']$ is roughly the hitting probability $q(y^w_{k'-1}, y^w_{k'}, k' - 1, x^w_{j+r}, j) + q(y^w_{k'-1}, y^w_{k'}, k' - 1, x^w_{j+r}, j + 1)$ except for some corner cases. To be more precise, define

$$Bound_{k',j,r,s} = 2p_{j-k'+2} \cdot \lambda[s = 1 \text{ OR } j < T] + 2p_{j-k'+1} \cdot \lambda[s = 0 \text{ AND } (k' \le j \text{ OR } r = 0)] \tag{4.28}$$

where the Boolean function $\lambda[\phi] = 1$ if ϕ is true and 0 otherwise. Then

Claim 22 $\mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X')|X' \wedge Y = k'] \le Bound_{k',j,r,s}$.

Proof We study the probability $\mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X')|X' \wedge Y = k']$ case by case. If $s = 1$, then $r = 1$, and $x^w_{j+1} \otimes z^c_{j+1,j} \in set(X')$ if and only if $x^w_{j+1} = (x')^w_{j+1}$. So

$$\mathbf{Pr}_{X'}[x^w_{j+r}\otimes z^c_{j+s,j} \in set(X')|X'\wedge Y = k'] = q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+1}, j+1) \le 2p_{j-k'+2} \tag{4.29}$$

by Lemma 19. If $s = 0$, then $x^w_{j+r} \otimes z^c_{j,j} \in set(X')$ if and only if "$x^w_{j+r} = (x')^w_j$ or $x^w_{j+r} = (x')^w_{j+1}$". Also note that

$$\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_j | X' \wedge Y = k'] = q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j) \tag{4.30}$$

unless $k' = j+1$ and $r = 1$, in which case $\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_j | X' \wedge Y = k'] = 0$ because $x^w_{j+1} \otimes z^c_{j,j} \notin set(Y)$ but $(x')^w_j \otimes z^c_{j,j} = y^w_j \otimes z^c_{j,j} \in set(Y)$. The other probability

$$\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_{j+1} | X' \wedge Y = k'] = \begin{cases} q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j+1) & \text{if } j \le T-1, \\ 0 & \text{if } j = T. \end{cases} \tag{4.31}$$

Putting all cases together, we get the desired result. □

(Continue the proof of Theorem 20) The claim implies that

$$v_{Y, x^w_{j+r} \otimes z^c_{j+s,j}} \le \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}. \tag{4.32}$$

The symmetric case of $u(X,Y,i)$ and $v(X,Y,i)$ where i is a point $y^w_{j+r} \otimes z^c_{j+s,j} \in set(Y) - set(X)$ can be dealt with in the same way, yielding $u_{X, y^w_{j+r} \otimes z^c_{j+s,j}} \le \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}$ and $v_{Y, y^w_{j+r} \otimes z^c_{j+s,j}} \le \sum_{k'=1}^{j+r} a_{k',j,r,s}$.

By the definition of $Bound_{k',j,r,s}$, it holds for any (j,r,s) that

$$\sum_{k'=1}^{j+r} Bound_{k',j,r,s} \le 4 \sum_{t=1}^{T} p_t \qquad \text{and} \qquad \sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}} \le 4 \sum_{t=1}^{T} \sqrt{p_t}. \tag{4.33}$$

Now for the randomized lower bound, $a_{k',j,r,s} = b_{k',j,r,s} = 1$.

$$RLS(G) = \Omega\left(\min_{j,r,s} \max \left\{ \frac{T - 2\sum_{t=1}^{T} p_t}{j+r}, \frac{T - 2\sum_{t=1}^{T} p_t}{\sum_{k'=1}^{j+r} Bound_{k',j,r,s}} \right\} \right) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t} \right). \tag{4.34}$$

For the quantum lower bound, pick $a_{k',j,r,s} = \sqrt{Bound_{k',j,r,s}}$, and $b_{k',j,r,s} = 1/\sqrt{Bound_{k',j,r,s}}$. Then

$$QLS(G) = \Omega\left(\min_{j,r,s} \sqrt{ \frac{\left(T - 2\sum_{t=1}^{T} p_t\right)\left(T - 2\sum_{t=1}^{T} p_t\right)}{\left(\sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}}\right)\left(\sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}}\right)} } \right) = \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}} \right) \tag{4.35}$$

This completes the proof of Theorem 20. □

4.4 Applications to the two special graphs

In this section, we will apply Theorem 20 to the two special graphs. Note that in both cases, the probability p_t is not easy to upper bound. Also note that we need not only to pick the random walk, but also the way to decompose the graph.

4.4.1 Lower bounds for Local Search on the Boolean Hypercube

To apply Theorem 20 to $\{0,1\}^n$, we decompose the whole graph into the two parts $\{0,1\}^m$ and $\{0,1\}^{n-m}$, where m is to be decided later (and to be taken different values for randomized and quantum lower bounds). Pick the random walk $(\{0,1\}^m, v_0^w, W)$, where $v_0^w = 0^m \in \{0,1\}^m$ and $W(x,t) = \{x^{(i)} : i \in \{0, ..., m-1\}\}$ for each vertex $x = x_0...x_{m-1} \in \{0,1\}^m$ and each $t \in \mathbb{N}$. Finally, note that the longest self-avoiding path of the graph $\{0,1\}^{n-m}$ is a Hamilton path with length $L = 2^{n-m} - 1$.

The following bounds on p_t are rather loose for $10 < t \leq m^2$ but sufficient for our purpose. The proof of the lemma uses some techniques in generating functions and Fourier analysis.

Lemma 23 *For any $t \in \mathbb{N}$, we have*

$$p_t = \begin{cases} O(m^{-\lceil t/2 \rceil}) & \text{if } \ t \leq 10 \\ O(m^{-5}) & \text{if } \ 10 < t \leq m^2 \\ O(2^{-m}) & \text{if } \ t > m^2 \end{cases} \tag{4.36}$$

Proof Consider that we put t balls randomly into m bins one by one. The j-th ball goes into the i_j-th bin. Denote by n_i the total number of balls in the i-th bin. We write $n_i \equiv b_i$ if $b_i = n_i \bmod 2$. We say that $(i_1, ..., i_t)$ *generates the parity sequence* $(b_1, ..., b_m)$, or simply $(i_1, ..., i_t)$ *generates* $(b_1, ..., b_m)$, if $n_i \equiv b_i$ for all $i \in [m]$. For $b_1...b_m \in \{0,1\}^m$, denote by $p^{(t)}[b_1, ..., b_m]$ the probability that $n_i \equiv b_i$, $\forall i \in [m]$. Let $p^{(t)} = \max_{b_1,...,b_m} p^{(t)}[b_1, ..., b_m]$. It is easy to see that $p^{(t)} = p_t$ in Lemma 23, so it is enough to prove the same bounds in Lemma 23 for $p^{(t)}$.

We start with several simple observations. First, we assume that t and $\sum_{i=1}^{m} b_i$ have the same parity, because otherwise the probability is 0 and the lemma holds trivially. Second, by the symmetry, any permutation of $b_1, ..., b_m$ does not change $p^{(t)}[(b_1, ..., b_m)]$. Third, $p^{(t)}[(b_1, ..., b_m)]$ decreases if we replace two 1's in $b_1, ..., b_m$ by two 0's. Precisely, if we have two b_i's being 1, say $b_1 = b_2 = 1$, then $p^{(t)}[(b_1, ..., b_m)] < p^{(t)}[(0, 0, b_3, ..., b_m)]$. In fact, note that

$$p^{(t)}[(b_1, ..., b_m)] = \frac{1}{m^t} \sum_{\substack{n_1+...+n_m=t \\ n_i \equiv b_i, i\in[m]}} \frac{t!}{n_1!...n_m!} \tag{4.37}$$

$$= \frac{1}{m^t} \sum_{\substack{n_3+...+n_m \le t \\ n_i \equiv b_i, i=3,...,m}} \left(\frac{t!}{(n_1+n_2)!n_3!...n_m!} \sum_{\substack{n_1+n_2=t-n_3-...-n_m \\ n_i \equiv b_i, i=1,2}} \frac{(n_1+n_2)!}{n_1!n_2!} \right) \tag{4.38}$$

where as usual, let $0! = 1$. If $n_3 + ... + n_m < t$, then

$$\sum_{\substack{n_1+n_2=t-n_3-...-n_m \\ n_i \equiv 1, i=1,2}} \frac{(n_1+n_2)!}{n_1!n_2!} = \sum_{\substack{n_1+n_2=t-n_3-...-n_m \\ n_i \equiv 0, i=1,2}} \frac{(n_1+n_2)!}{n_1!n_2!} \tag{4.39}$$

If $n_3 + ... + n_m = t$, then the only possible (n_1, n_2) is $(0, 0)$, so

$$\sum_{\substack{n_1+n_2=t-n_3-...-n_m \\ n_i \equiv 1, i=1,2}} \frac{(n_1+n_2)!}{n_1!n_2!} = 0, \qquad \sum_{\substack{n_1+n_2=t-n_3-...-n_m \\ n_i \equiv 0, i=1,2}} \frac{(n_1+n_2)!}{n_1!n_2!} = 1. \tag{4.40}$$

Thus $p^{(t)}[(1, 1, b_3, ..., b_m)] < p^{(t)}[(0, 0, b_3, ..., b_m)]$.

By these observations, it is sufficient to prove the lemma for the case $p^{(t)}[(0, ..., 0)]$ if t is even, and for the case $p^{(t)}[(1, 0, ..., 0)]$ if t is odd. Note that if t is even, then

$$p^{(t)}[(0, ..., 0)] = \sum_{i=1}^{m} \mathbf{Pr}[i_1 = i]\mathbf{Pr}[(i_2, ..., i_t) \text{ generates } (e_i)] \tag{4.41}$$

where e_i is the m-long vector with only coordinate i being 1 and all other coordinates being 0. By symmetry, $p^{(t-1)}[e_1] = ... = p^{(t-1)}[e_m]$, thus $p^{(t)}[(0, ...0)] = p^{(t-1)}[e_1] = p^{(t-1)}[1, 0, ..., 0]$. Therefore, it is enough to show the lemma for even t.

We now express $p^{(t)}[0, ..., 0]$ in two ways. One is to prove the first case ($t \le 10$) in the lemma, and the other is for the second case ($10 < t \le m^2$) and the third case ($t > m^2$) in the lemma.

To avoid confusion, we write the number m of bins explicitly as subscript: $p_m^{(t)}[b_1, ..., b_m]$. We consider which bin(s) the first two balls are put into.

$$p_m^{(t)}[0, ..., 0] = \mathbf{Pr}[i_1 = i_2] p_m^{(t-2)}[0, ..., 0] + \mathbf{Pr}[i_1 \neq i_2] p_m^{(t-2)}[1, 1, 0, ..., 0] \tag{4.42}$$

$$= \frac{1}{m} p_m^{(t-2)}[0, ..., 0] + \frac{m-1}{m} p_m^{(t-2)}[1, 1, 0, ..., 0] \tag{4.43}$$

To compute $p_m^{(t-2)}[1, 1, 0, ..., 0]$, we consider how to put $(t-2)$ balls in m bins. By the analysis of the third observation above, we know that

$$p_m^{(t-2)}[0, ..., 0] - p_m^{(t-2)}[1, 1, 0, ..., 0] \tag{4.44}$$

$$= \mathbf{Pr}[n_1 = n_2 = 0, n_3 \equiv 0, ..., n_m \equiv 0] \tag{4.45}$$

$$= \mathbf{Pr}[n_1 = n_2 = 0] \mathbf{Pr}[n_3 \equiv 0, ..., n_m \equiv 0 | n_1 = n_2 = 0] \tag{4.46}$$

$$= \left(\frac{m-2}{m}\right)^{t-2} p_{m-2}^{(t-2)}[0, ..., 0] \tag{4.47}$$

Therefore,

$$p_m^{(t)}[0, ..., 0] = \frac{1}{m} p_m^{(t-2)}[0, ..., 0] - \frac{m-1}{m} \left(\frac{m-2}{m}\right)^{t-2} p_{m-2}^{(t-2)}[0, ..., 0]. \tag{4.48}$$

Now using the above recursive formula and the base case $p_m^{(2)}[0, ..., 0] = 1/m$, it is easy (but tedious) to prove by calculations that $p_m^{(t)}[0, ..., 0] = ((t-1)!!/m^{\frac{t}{2}})(1 - o(1))$ for even $t \leq 10$. This proves the first case in the lemma.

For the remaining two cases, we shall use generating function and some technique inspired by Fourier analysis. Consider the generating function

$$(x_1 + ... + x_m)^t = \sum_{n_1 + ... + n_m = t} \binom{t}{n_1, ..., n_m} x_1^{n_1} ... x_m^{n_m}. \tag{4.49}$$

If $x_i \in \{-1, 1\}$, then $(x_1 + ... + x_m)^t = \sum_{n_1 + ... + n_m = t} \binom{t}{n_1, ..., n_m} (-1)^{|\{i : x_i = -1, n_i \equiv 1\}|}$. We sum it over all $x_1 ... x_m \in \{-1, 1\}^m$. Note that for those $(n_1, ..., n_m)$ that have some $n_{i_0} \equiv 1$, it holds due to the cancelation that $\sum_{x_1, ..., x_m \in \{-1,1\}} (-1)^{|\{i : x_i = -1, n_i \equiv 1\}|} = 0$. On the other hand, if all n_i's are even, then $\sum_{x_1, ..., x_m \in \{-1,1\}} (-1)^{|\{i : x_i = -1, n_i \equiv 1\}|} = 2^m$. Thus we have

$$\sum_{x_1, ..., x_m \in \{-1,1\}} (x_1 + ... + x_m)^t = 2^m \sum_{\substack{n_1 + ... + n_m = t \\ n_i \equiv 0, i \in [m]}} \binom{t}{n_1, ..., n_m}. \tag{4.50}$$

And therefore,

$$p^{(t)}[0,...,0] = \frac{1}{m^t} \sum_{\substack{n_1+...+n_m=t \\ n_i \equiv 0, i\in[m]}} \binom{t}{n_1,...,n_m} \tag{4.51}$$

$$= \frac{1}{2^m m^t} \sum_{x_1,...,x_m \in \{-1,1\}} (x_1 + ... + x_m)^t \tag{4.52}$$

$$= \frac{1}{2^m m^t} \sum_{i=0}^{m} \binom{m}{i} (m-2i)^t \tag{4.53}$$

$$= \frac{1}{2^m} \sum_{i=0}^{m} \binom{m}{i} \left(1 - \frac{2i}{m}\right)^t. \tag{4.54}$$

Note that t is even, so $p^{(t)}[0,...,0]$ decreases if t increases by 2, and this proves the second case of the lemma with the help of the first case. And if $t > m^2/2$, then

$$p^{(t)}[0,...,0] \leq \frac{1}{2^m}\left(2 + \left(1 - \frac{2}{m}\right)^t \sum_{i=1}^{m-1} \binom{m}{i}\right) < 2/2^m + e^{-m} = O(1/2^m) \tag{4.55}$$

This proves the third case of the lemma. □

Now it is very easy to prove Theorem 15 using this lemma. For the randomized lower bound, let $m = \lfloor (n + \log_2 n)/2 \rfloor$, then $T = \Theta(2^{n/2}/n^{1/2})$ and $\sum_{t=1}^{T} p_t = O(1/n)$. Thus $RLS(\{0,1\}^n) = \Omega(\sqrt{n}2^{n/2})$. For the quantum lower bound, let $m = \lfloor (2n + \log_2 n)/3 \rfloor$, then $T = \Theta(2^{n/3}/n^{1/3})$ and $\sum_{t=1}^{T} \sqrt{p_t} = O(1/\sqrt{n})$. Thus $QLS(\{0,1\}^n) = \Omega(2^{n/3}n^{1/6})$.

4.4.2 Lower bounds for Local Search on the constant dimensional grid

In this section we shall first prove a lower bound weaker than Theorem 16 in Section 4.4.2, and then improve it to Theorem 16 in Section 4.4.2 and Section 4.4.2.

A weaker family of lower bounds

To simplify notations, we let $n = N^{1/d}$. As in Section 4.4.1, we decompose the grid into two parts, $[n]^m$ and $[n]^{d-m}$. For each vertex $x = x_0...x_{m-1} \in [n]^m$ and each $i \in \{0,...,m-1\}$, define

$$x^{(i),-} = x_0...x_{i-1} \max\{x_i - 1, 1\} x_{i+1}...x_{m-1}, \tag{4.56}$$

$$x^{(i),+} = x_0...x_{i-1} \min\{x_i + 1, n\} x_{i+1}...x_{m-1}. \tag{4.57}$$

We perform the random walk $([n]^m, v_0^w, W)$ where $v_0^w = 00...0 \in [n]^m$ and

$$W(x,t) = \{x^{((t-1) \bmod m),+}, x^{((t-1) \bmod m),-}\}. \tag{4.58}$$

To analyze the probability p_t in Theorem 20, we first consider the following simpler "line walk". Suppose a particle is initially put at point $i \in \{1, ..., n\}$, and in each step the particle moves either to $\max\{1, i-1\}$ or to $\min\{n, i+1\}$, each with probability 1/2. Let $p_{ij}^{(t)}$ denote the probability that the particle starting from point i stops at point j after exact t steps of the walk. For $t \geq 1$, the following proposition gives a very good (actually tight) estimate on $\max_{ij} p_{ij}^{(t)}$.

Proposition 24 *For any $t \geq 1$,*

$$\max_{i,j} p_{ij}^{(t)} = \begin{cases} O(1/\sqrt{t}) & \text{if} \quad t \leq n^2 \\ O(1/n) & \text{if} \quad t > n^2. \end{cases} \tag{4.59}$$

Before the formal proof, let us briefly discuss the main difficulty and the idea to get around it. First note that since we care about the whole mixing process (*i.e.* before and after mixing), the standard eigenvalue gap does not immediately apply. Second, if there are not the two barriers (1 and n) then $p_{ij}^{(t)}$ is very easy to calculate: $p_{ij}^{(t)} = \binom{t}{t/2+(j-i)/2}$ if $j-i$ and t have the same parity, and 0 otherwise. However, since we now have the two barriers, it is hard to count the number of paths from i to j after exactly t steps. Fortunately, there is a basic *reflecting rule* as follows.

reflecting rule: In the line walk without barrier, the number of paths from $i > 0$ to $j > 0$ in exactly t steps touching or crossing the point 0 is equal to the number of paths from $-i$ to j in exactly t steps.

The proof of this rule is very easy. Suppose a random path touches the point 0 at t for the first time, then do a reflection of the first t steps of the path with respect to point 0. See Figure 4.1 for an illustration. It is not hard to see that this gives a 1-1 correspondence between the following two sets: 1) the set of paths from i to j after exactly t steps touching or crossing the point 0, and 2) the set of paths from $-i$ to j.

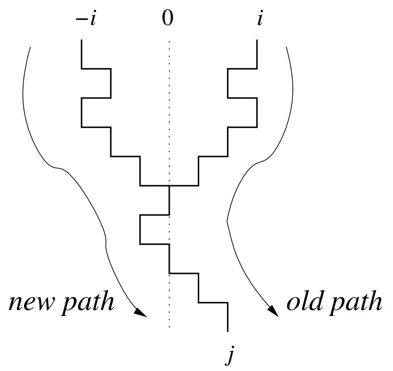

Figure 4.1: The proof of the reflecting rule.

Now let us consider the barrier setting. Note that a path may try to cross the two barriers in some pattern, for example, try to cross the left barrier (*i.e.* point 1) a times and then try to cross the right barrier (*i.e.* point n) b times. Imagine that we now remove the two barriers, then the path will touch (from right) but not cross the point $1-a$ and will touch (from left) but not cross the point $n+b-a$. To use the reflecting rule, we just need to further note the following simple fact:

$$\begin{aligned}&\{\text{paths touching but not crossing the point } 1-a\}\\ =\;&\{\text{paths touching or crossing the point } 1-a\}-\{\text{paths touching or crossing the point } -a\}.\end{aligned}$$

Following this idea, we will construct a series of 1-1 correspondences to reduce the problem step by step to the no-barrier case. The precise proof is as follows.

Proof We consider two settings. One is the line walk on n points $0,...,n-1$ with the two barriers 0 and $n-1$ [6]. Another is the same except that the barriers are removed, and we have infinite points in a line. For each t-bit binary string $x = x_1...x_t$, we use P_i^x and Q_i^x to denote the two paths that starting at i and walk according to x in the two settings. Precisely, at step s, Q_i^x goes left if $x_s = 0$ and goes right if $x_s = 1$. P_i^x goes in the same way except that it will stand still if the point is currently at left (or right) end and it still wants to go left (or right). If the end point of P_i^x is j, then we write $i \rightarrow_t^{P,x} j$. Let $X_{ij}^{(t),P}$ be the set of $x \in \{0,1\}^t$ such that $i \rightarrow_t^{P,x} j$, and put $n_{ij}^{(t),P} = |X_{ij}^{(t),P}|$. Then by definition,

[6]Here we let the n points be $0,...,n-1$ instead of $1,...,n$ just to make the later calculation cleaner.

$p_{ij}^{(t)} = n_{ij}^{(t),P}/2^t$. The notations $i \to_t^{Q,x} j$, $X_{ij}^{(t),Q}$ and $n_{ij}^{(t),Q}$ are similarly defined, with the corresponding P changed to Q. Note that $n_{ij}^{(t),Q} = \binom{t}{t/2+(j-i)/2}$ if $j-i$ and t have the same parity, and 0 otherwise. We now want to upper bound $n_{ij}^{(t),P}$ in terms of $n_{ij}^{(t),Q}$.

For a path P_i^x, if at some step it is at point 0 and wants to go left, we say it *attempts to pass the left barrier.* Similarly for the right barrier. We say a path is in the $\{a_s, b_s\}_{s=1}^l$ category if it first attempts to pass the left barrier for a_1 times, and then attempts to pass the right barrier for b_1 times, and so on. We call each round a stage s, which begins at the time that P_i^x attempts to pass the left barrier for the $(a_1 + ... + a_{s-1} + 1)$-th time, and ends right before the time that P_i^x attempts to pass the left barrier for the $(a_1 + ... + a_s + 1)$-th time. We also split each stage s into two halves, cutting at the time right before the path attempts to pass the right barrier for the $(b_1 + ... + b_{s-1} + 1)$-th time. Note that a_1 may be 0, which means that the path first attempts to pass the right barrier. Also b_l may be 0, which means the the last barrier the path attempts to pass is the left one. But all other a_i, b_i's are positive. Also note that in the case of $l = 0$, the path never attempts to pass either barrier. Now for any fixed $l > 0$, we consider those categories with $a_1 > 0$ and $b_l > 0$. Other cases can handled similarly. Partition $X_{ij}^{(t),P}$ as

$$X_{ij}^{(t),P} = \bigcup_{l,\ \{a_s,b_s\}_{s=1}^l} X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l] \tag{4.60}$$

where $X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$ contains those $x \in \{0,1\}^t$ such that P_i^x is in the category $\{a_s, b_s\}_{s=1}^l$. Put $n_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l] = |X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]|$, thus $n_{ij}^{(t),P} = \sum_l \sum_{\{a_s,b_s\}_{s=1}^l} n_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$.

Now consider the corresponding paths in $X_{ij}^{(t),Q}$. The following observation relates P_i^x and Q_i^x.

Observation 25 *For each $x \in X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$, the following three properties hold for any s.*

1. *In the first half of stage s, the path Q_i^x touches (from right) but does not cross the point $\alpha_s = \sum_{r=1}^{s-1}(b_r - a_r) - a_s$.*

2. *In the second half of stage s, the path Q_i^x touches (from left) but does not cross the point $\beta_s = n - 1 + \sum_{r=1}^{s}(b_r - a_r)$.*

3. The path Q_i^x ends at $\gamma = j + \sum_{s=1}^{l}(b_s - a_s)$.

We let $Y_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$ contain those $x \in \{0,1\}^t$ satisfying the three conditions in the above observation, and denote by $m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$ the size of the set $Y_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$. Thus the observation says $X_{ij}^{(t),P}[\{\alpha_s,\beta_s\}_{s=1}^{l}] \subseteq Y_{ij}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$, and therefore we have $n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^{l}] \le m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$. So it is enough to upper bound $m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$.

Now for each $x \in Y_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$, if we change the condition 1 in state $s = 1$ by allowing the path to cross the point α_1, and let $Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$ be the new set satisfying the new conditions, then $m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}] = |Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]| - |Z_{i\gamma}^{(t),Q}[\alpha_1 - 1,\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}]|$. In other words, the set of paths touches (from right) but does not cross α_1 is the set of paths touches or crosses α_1 minus the set of paths touches or crosses $\alpha_1 - 1$.

Now we calculate $|Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]|$ by the so-called reflection rule. Suppose the first time that Q_i^x touches α_1 is t_1. We reflect the first t_1 part of the path Q_i^x with respect to the point α_1. Precisely, let $y = (1-x_1)...(1-x_{t_1})x_{t_1+1}...x_t$, then the paths Q_i^x and $Q_{2\alpha_1-i}^{y}$ merge at time t_1. And it is easy to check that it is a 1-1 correspondence between $Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]$ and $Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}]$, Here $Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}]$ is the set of paths starting at $2\alpha_1 - i$, satisfying (a) the condition 2 at the first stage, (b) both conditions 1 and 2 at the rest $l-1$ stages, and (c) condition 3. So

$$|Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}]| = |Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}]| = m_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}] \tag{4.61}$$

$$= m_{-2a_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}] \tag{4.62}$$

$$= m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1 + a_1,\{\alpha_s + a_1,\beta_s + a_1\}_{s=2}^{l}] \tag{4.63}$$

where (4.62) is due to the fact that $\alpha_1 = -a_1$, and (4.63) is because that the number of the paths does not change if we move all the paths right by a_1. Similarly, we have

$$|Z_{i\gamma}^{(t),Q}[\alpha_1 - 1,\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}]| = m_{2\alpha_1-2-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}] \tag{4.64}$$

$$= m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1 + a_1,\{\alpha_s + a_1,\beta_s + a_1\}_{s=2}^{l}]. \tag{4.65}$$

Therefore,

$$
\begin{aligned}
n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^{l}] &\le m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^{l}] && (4.66)\\
&= m_{-2a_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}] - m_{-2a_1-2-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^{l}] && (4.67)\\
&= m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]\\
&\quad - m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]. && (4.68)
\end{aligned}
$$

Note that $\alpha_s+a_1 = b_1+\sum_{r=2}^{s-1}(b_r-a_r)-a_s$, $\beta_s+a_1 = n-1+b_1+\sum_{r=2}^{s}(b_r-a_r)$ and $\gamma+a_1 = j+b_1+\sum_{r=2}^{s}(b_r-a_r)$ are all functions of $(b_1,a_2,b_2,...,a_l,b_l)$, not of a_1 any more. Therefore,

$$
\begin{aligned}
&\sum_{a_1,b_1,...,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^{l}] && (4.69)\\
\le &\sum_{b_1,...,a_l,b_l>0}\sum_{a_1>0}(m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]\\
&\qquad - m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]) && (4.70)\\
= &\sum_{b_1,...,a_l,b_l>0}(m_{-1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]\\
&\qquad + m_{-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]) && (4.71)\\
\le &\sum_{b_1,...,a_l,b_l>0} 2\max_{h=1,2}\{m_{-h-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]\}. && (4.72)
\end{aligned}
$$

Now using the similar methods, *i.e.* reflecting with respect to points $(n-1+b_1)$ and $(n+b_1)$, moving the paths left by b_1, and finally collapsing the telescope, we can get

$$
\begin{aligned}
&\sum_{b_1,...,a_l,b_l>0} m_{-h-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}]\\
\le &\sum_{a_2,b_2,...,a_l,b_l>0} 2\max_{k=1,2}\{m_{2n+i+h-k+1,\gamma+a_1-b_1}^{(t),Q}[\{\alpha_s+a_1-b_1,\beta_s+a_1-b_1\}_{s=2}^{l}]\} && (4.73)
\end{aligned}
$$

and thus

$$
\begin{aligned}
&\sum_{a_1,b_1,...,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^{l}]\\
\le &\sum_{a_2,b_2,...,a_l,b_l>0} 4\max_{h=0,1,2}\{m_{2n+i+h,\gamma+a_1-b_1}^{(t),Q}[\{\alpha_s+a_1-b_1,\beta_s+a_1-b_1\}_{s=2}^{l}]\}. && (4.74)
\end{aligned}
$$

We continue this process, and finally we get

$$\sum_{a_1,b_1,\ldots,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^l] \leq 2^{2l} \max_{h=0,1,\ldots,2l} n^{(t),Q}_{2ln+i+h,\gamma+\sum_{s=1}^l(a_s-b_s)} \tag{4.75}$$

$$= 2^{2l} \max_{h=0,1,\ldots,2l} n^{(t),Q}_{2ln+i+h,j} \tag{4.76}$$

$$= 2^{2l} n^{(t),Q}_{2ln+i,j} \tag{4.77}$$

$$\leq 2^{2l}\binom{t}{\frac{t}{2}+\frac{j-i-2ln}{2}}. \tag{4.78}$$

Thus

$$\sum_{l>0}\sum_{a_1,b_1,\ldots,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^l] \tag{4.79}$$

$$\leq \sum_{l\geq 0} 2^{2(l+1)}\binom{t}{\frac{t}{2}+ln} \tag{4.80}$$

$$=4\binom{t}{t/2}+\sum_{l\geq 1} 2^{2(l+1)}\binom{t}{t/2+ln} \tag{4.81}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}\left(\binom{t}{t/2+ln}+\binom{t}{t/2+ln-1}+\ldots+\binom{t}{t/2+ln-n+1}\right) \tag{4.82}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}\left(\binom{t}{t}+\binom{t}{t-1}+\ldots+\binom{t}{t/2+ln-n+1}\right) \tag{4.83}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}2^t e^{-\frac{2(l-1)^2n^2}{3t}} \tag{4.84}$$

where $\binom{t}{t'}=0$ if $t'>t$. Here the first two inequalities are by the monotonicity of binomial coefficients, and the last inequality is by Chernoff's Bound. Now if $t\leq n^2$, then $\sum_{l\geq 1} 2^{2(l+1)}e^{-\frac{2(l-1)^2n^2}{3t}} \leq \sum_{l\geq 1} 2^{2(l+1)}e^{-\frac{2(l-1)^2}{3}} = O(1)$, so $\sum_{l>0}\sum_{a_1,b_1,\ldots,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^l] \leq O(\binom{t}{t/2}+2^t/n) = O(2^t/\sqrt{t})$. For other categories of $a_1=0$ or $b_l=0$, the same result can be proved similarly, and the $l=0$ is easy since $n_{ij}^{(t),Q}=O(2^t/\sqrt{t})$. Putting all things together, we see that $p_{ij}^{(t)}=O(1/\sqrt{t})$ if $t\leq n^2$. The other part, *i.e.* $p_{ij}^{(t)}=O(1/n)$ when $t>n^2$, can be easily derived from this and the fact that $\max_{ij} p_{ij}^{(t)}$ decreases as t increases. This completes our proof. $\square$

Now we use Proposition 24 to prove the weaker lower bounds for grids. Note that the random walk $([n]^m, v_0^w, W)$ is just a product of m line walks, *i.e.* cyclicly perform the line walk in the cyclic order of dimension $0, 1, ..., m-1$ (see Eq. (4.58)). Therefore, the p_t in the random walk $([n]^m, v_0^w, W)$ satisfies

$$p_t = \begin{cases} O(1/\sqrt{t^m}) & \text{if } t \leq n^2, \\ O(1/n^m) & \text{if } t > n^2. \end{cases} \tag{4.85}$$

Now for the randomized lower bounds, when $d > 4$ we pick $m = \lceil d/2 \rceil > 2$ and we get

$$RLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1) + n^{d-m}/n^m}\right) = \Omega(n^{\lfloor d/2 \rfloor}) = \begin{cases} \Omega(n^{\frac{d}{2}}) & \text{if } d \text{ is odd,} \\ \Omega(n^{\frac{d}{2}-\frac{1}{2}}) & \text{if } d \text{ is even.} \end{cases} \tag{4.86}$$

For $d = 4, 3, 2$, we let $m = 2, 2, 1$ respectively, and get $RLS([n]^4) = \Omega(n^2/(\log n + 1)) = \Omega(n^2/\log n)$, $RLS([n]^3) = \Omega(n/(\log n + 1/n)) = \Omega(n/\log n)$, and $RLS([n]^2) = \Omega(n/(\sqrt{n} + 1)) = \Omega(\sqrt{n})$.

For the quantum lower bounds, if $d > 6$, we let m be the integer closest to $2d/3$, thus $m > 4$. We get

$$QLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1) + n^{d-m}/n^{m/2}}\right) = \begin{cases} \Omega(N^{\frac{1}{3}}) & \text{if } d = 3d' \\ \Omega(N^{\frac{1}{3}-\frac{1}{3d}}) & \text{if } d = 3d'+1 \\ \Omega(N^{\frac{1}{3}-\frac{1}{6d}}) & \text{if } d = 3d'+2 \end{cases} \cdot \tag{4.87}$$

For $d = 6$, let $m = 4$ and we have $QLS([n]^6) = \Omega(n^2/\log n)$. For $d = 5, 4, 3$, we let $m = d-2$ and then $QLS([n]^d) = \Omega(n^2/(n^{2-(d-2)/2} + n^{2-(d-2)/2})) = \Omega(n^{d/2-1})$, which is $\Omega(n^{5/2}), \Omega(n^2), \Omega(n^{3/2})$, respectively. For $d = 2$, let $m = 1$ and $QLS([n]^2) = \Omega(\frac{n}{n^{3/4}}) = \Omega(n^{1/4})$.

Improvement

One weakness of the above proof is the integer constraint of the dimension m. We now show a way to get around the problem, allowing m to be any real number between 0 and

$d-1$. The idea is to partition the grid into many blocks, with different blocks representing different time slots, and the blocks are threaded into one very long block by many paths that are pairwise disjoint. Roughly speaking, we view $[n]^d$ as the product of d line graph $[n]$. For each of the first $d-1$ line graphs, we cut it into n^{1-r} parts evenly, each of size n^r. (Here $r = m/(d-1)$). Then $[n]^{d-1}$ is partitioned into $n^{(d-1)(1-r)}$ smaller grids, all isomorphic to $[n^r]^{d-1}$. Putting the last dimension back, we have $n^{(d-1)(1-r)}$ *blocks*, all isomorphic to $[n^r]^{d-1} \times [n]$. Now the random walk will begin in the first block, and within each block, there is just one step of random walk in $[n^r]^{d-1}$ followed by two steps of one-way walk in the last dimension space $[n]$. When the walk runs out of the clock $[n]$, the walk will move to the next block via a particular block-changing path. All block-changing paths are carefully designed to be disjoint, and they "thread" all the blocks to form a $[n^r]^{d-1} \times [L]$ grid, where $L = (n - 2n^r)n^{(1-r)(d-1)}$. ($L$ is not $n \cdot n^{(1-r)(d-1)}$ because we need $2n^r$ points for the block-changing paths.) Figure 4.2 is an illustration for the case of $d = 2$.

We now describe the partition and the walk precisely. For $x = x_0...x_{d-1}$ in $[n]^d$, let $x^{(k)=l} = x_0...x_{k-1}lx_{k+1}...x_{d-1}$, and $x^{(k)=(k)+i} = x_0...x_{k-1}(x_k + i)x_{k+1}...x_{d-1}$, where i satisfies $x_k + i \in [n]$. Recall that $x^{(i),-} = x^{(i)=\max\{x_i-1,1\}}$ and $x^{(i),+} = x^{(i)=\min\{x_i+1,n\}}$.

For any fixed constant $r \in (0,1)$, let $\alpha = \lfloor n^r \rfloor$, $\beta = \lfloor n^{1-r} \rfloor$ and $n' = \alpha\beta$. Note that $n' \geq (n^r - 1)(n^{1-r} - 1) = n - o(n)$. We now consider the slightly smaller grid $[n']^d$. Let V_1 be the set $[n']^{d-1} = \{x_0...x_{d-2} : x_i \in [n']\}$. We cut V_1 into β^{d-1} parts $\{x_0...x_{d-2} : (k_i - 1)\alpha < x_i \leq k_i\alpha\}_{k_0...k_{d-2} \in [\beta]^{d-1}}$, each of which is a small grid isomorphic to $[\alpha]^{d-1}$. We then refer to the set $\{x_0...x_{d-2}x_{d-1} : (k_i - 1)\alpha < x_i \leq k_i\alpha, i = 0, ..., d-2, \alpha < x_{d-1} \leq n' - \alpha\}$ as the "*block* $(k_0, ..., k_{d-2})$". Note that $(k_0, ..., k_{d-2})$ can be also viewed as a point in grid $[\beta]^{d-1}$, and there is a Hamilton path $HamPath_{\beta,d-1}$ in $[\beta]^{d-1}$, as defined in Section 2. We call the block $(k'_0, ..., k'_{d-2})$ *the next block* of the block $(k_0, ..., k_{d-2})$ if $(k'_0, ..., k'_{d-2})$, viewed as the point in $[\beta]^{d-1}$, is the next point of $(k_0, ..., k_{d-2})$ in $HamPath_{\beta,d-1}$. Note that by our definition of $HamPath_{\beta,d-1}$, we know that $\exists i \in \{0, ..., d-2\}$ such that $k'_i \in \{k_i + 1, k_i - 1\}$ and for all other $j \neq i$, $k'_j = k_j$. That is, adjacent blocks have only one coordinate to be different, and this difference is 1. We call the block $(k_0, ..., k_{d-2})$ *the last block* if $(k_0, ..., k_{d-2})$

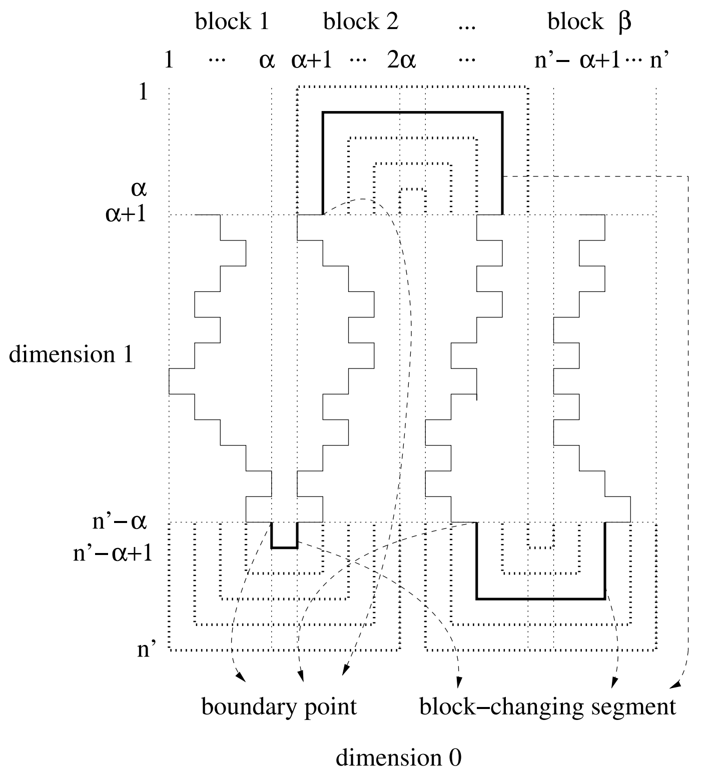

Figure 4.2: Illustration for changing a block in the 2-dimensional grid

is the last point in $HamPath_{\beta,d-1}$.

Now we define the random walk by describing how a particle may go from start to end. The path set is just all the possible paths the particle goes along. Intuitively, within one block, the last dimension $d-1$ serves as the clock space. So as before, we perform one step of line walk (in the dimension which is the circularly next dimension of the last one that the walk just goes in), followed by two steps of walk in the clock space. If we run out of clock, we say we reach *a boundary point* at the current block, and we move to the next block via a path segment called *block-changing segment.* In the following Algorithm 4.1, we specify how the particle may move during the whole random walk process, including going through block-changing segments. We always use $x_0 ... x_{d-1}$ to denote the current position of the particle, and assume $x_i = (k_i - 1)\alpha + y_i$, *i.e.* x is in the block $(k_0, ..., k_{d-2})$ with the offsets $(y_0, ..., y_{d-1})$. Thus the instruction $x_0 = x_0 + 1$, for example, means that the particle moves from $x_0 ... x_{d-1}$ to $(x_0 + 1)x_1 ... x_{d-1}$.

```
Algorithm 4.1: specify the random walk in a real number dimensional grid.
  1. Initially $x_0 = ... = x_{d-2} = 0$, $x_{d-1} = \alpha + 1$, $k_0 = ... = k_{d-2} = 1$.
  2. for $t = 1$ to $(n' - 2\alpha)\beta^{d-1}$,
       Let $t' = \lfloor \frac{t-1}{n'-2\alpha} \rfloor$, $i = (t-1) \bmod (d-1)$
       do either $x_i = \max\{x_i - 1, (k_i - 1)\alpha + 1\}$ or $x_i = \min\{x_i + 1, k_i\alpha\}$ randomly
       if $t \neq k(n' - 2\alpha)$ for some positive integer $k$,
         do $x_{d-1} = x_{d-1} + (-1)^{t'}$ twice
       else   (the particle is now at a boundary point)
         if the particle is not in the last block
         (Suppose the current block changes to the next block by increasing $k_j$ by $b \in \{-1, 1\}$)
           do $x_{d-1} = x_{d-1} + (-1)^{t'}$ for $(\alpha + 1 - y_j)$ times
           do $x_j = x_j + b$ for $2(\alpha + 1 - y_j) - 1$ times
           do $x_{d-1} = x_{d-1} + (-1)^{t'+1}$ for $(\alpha + 1 - y_j)$ times
           $k_j = k_j + b$
         else
           The particle stops and the random walk ends
```

It is easy to verify that every boundary point has one unique block-changing segment, and different block-changing segments do not intersect. Also note that we do not let the clock tick when we are moving from one block to another. Thus the block-changing segments thread all the blocks to form a $[\alpha]^{d-1} \times [L]$ grid, where $L = (n' - 2\alpha)\beta^{d-1}$. Actually, for our lower bound purpose, we can think of the random walk as performed in the product graph $[\alpha]^{d-1} \times [L]$. We will make this clearer below.

What we care about is, as before, the probability that the random walk starting from a point $x = x_0 ... x_{d-1}$ passes another point $x' = x'_0 ... x'_{d-1}$. Note that for any point x (including those on the block-changing segments), there is only one time t when the walk may hit x, and this t is determined by x itself. Similarly we use t' to denote the time when the path passes x'. Denote the probability that the random walk starting from x passes

x' by $\mathbf{Pr}[x \to x']$. As before suppose $x_i = (k_i - 1)\alpha + y_i$ and $x'_i = (k'_i - 1)\alpha + y'_i$ for $i \in \{0, ..., d-2\}$.

We first consider the case that one of the two points, say x' is on a block-changing segment. Since different block-changing segments never intersect, a path passes x' if and only if the path passes the boundary point x'' at the beginning of the block-changing segment that x' is in. Also note that the time that the path passes x'' is also t' because the time does not elapse on the block-changing segment. So we have that $\mathbf{Pr}[x \to x'] = \mathbf{Pr}[x \to x'']$, and it is enough to consider the case that both x and x' are not in clock-changing segments.

Now suppose both x and x' are not in clock-changing segments. In general, x and x' may be not in the same block , so going from x to x' needs to change blocks. Recall that to change from the block $(k_0, ..., k_{d-2})$ to the next one, only one k_i changes by increasing or decreasing by 1. Suppose that to go to x' from x, we change blocks for c times, by changing $k_{i_1}, k_{i_2}, ..., k_{i_c}$ in turn. Let $n_j = |\{s \in [c] : i_s = j\}|$. Note that to get to x' from x after $t' - t$ steps, the coordinate j needs to be x'_j after $t' - t$ steps for each coordinate $j \in \{0, ..., d-2\}$. It is not hard to see that if a block-changing needs to change k_j by increasing $b \in \{-1, 1\}$, then among all the offsets y_i's, only the y_j gets changed, and the change is a reflection within the block. That is, suppose x_j is $(k_j - 1)\alpha + y_j$ before the block-changing, then x_j changes to $(k_j + b - 1)\alpha + (\alpha + 1 - y_j)$ after the block-changing. So if $c = 1$, then $\mathbf{Pr}[x \to x']$ is equal to the probability that a random walk in $[\alpha]^{d-1}$ starting from $y_0 ... y_{d-2}$ hits $y''_0 ... y''_{d-2}$ after exactly $t' - t$ steps, where $y''_j = y'_j$ if $j \neq i_1$ and $y''_{i_1} = \alpha + 1 - y'_{i_1}$. For general c, $\mathbf{Pr}[x \to x']$ is equal to the probability that a random walk in $[\alpha]^{d-1}$ starting from $y_0 ... y_{d-2}$ hits $y''_0 ... y''_{d-2}$ after exactly $t' - t$ steps, where $y''_j = y'_j$ if n_j is even and $y''_j = \alpha + 1 - y'_j$ if n_j is odd. Note that this probability has nothing to do with the block-changing; it is just the same as we have a clock space $[(n' - 2\alpha)\beta^{d-1}]$ to record the random walk on $[\alpha]^{d-1}$. Thus we can use Proposition 24 to upper bound this probability and just think of the graph as $[n^r]^{d-1} \times [L]$ and use Theorem 20, with $G^w = [n^r]^{d-1}$ and $G^c = [L]$.

Now we have $T = \lfloor L/2 \rfloor$ and $p_t = O(1/\sqrt{t^{d-1}})$ for $t \leq n^{2r}$ and $p_t = O(1/n^{r(d-1)})$ for

$t > n^{2r}$. So for randomized lower bounds, if $d \geq 4$, then let $r = d/(2d-2)$ and we get

$$RLS([n]^d) = \Omega\left(n^{1+(1-r)(d-1)} \Big/ \left(\sum_{t=1}^{n^{d/(d-1)}} \frac{1}{\sqrt{t^{d-1}}} + \frac{n^{1+(1-r)(d-1)}}{n^{r(d-1)}}\right)\right) = \Omega\left(n^{d/2}\right). \quad (4.88)$$

If $d = 3$, let $r = 3/4 - \log\log n/(4\log n)$, and we get $RLS([n]^3) = \Omega((n^3/\log n)^{1/2})$. For $d = 2$, let $r = 2/3$ and we get $RLS([n]^2) = \Omega(n^{2/3})$.

For the quantum lower bounds, if $d \geq 6$, then let $r = 2d/(3d-3)$ and we get

$$QLS([n]^d) = \Omega\left(n^{1+(1-r)(d-1)} \Big/ \left(\sum_{t=1}^{n^{d/(d-1)}} \frac{1}{t^{(d-1)/4}} + \frac{n^{1+(1-r)(d-1)}}{n^{r(d-1)/2}}\right)\right) = \Omega(n^{d/3}). \quad (4.89)$$

If $d = 5$, then let $r = 5/6 - \log\log n/(6\log n)$ and $QLS([n]^5) = \Omega((n^5/\log n)^{1/3})$. For $2 \leq d \leq 4$, we let $r = d/(d+1)$, then $QLS([n]^d) = \Omega(n^{d/2-d/(d+1)})$, which is $\Omega(n^{1/3})$, $\Omega(n^{3/4})$, $\Omega(n^{6/5})$ for $d = 2, 3, 4$, respectively.

Further improvement on 2-dimensional grid $[n]^2$

Some other random walk may be used to further improve the lower bound on low dimension grid cases. Here is one way to improve $QLS([n]^2)$ from $\Omega(n^{1/3})$ to $\Omega(n^{4/5})$. We cut the graph $[n]^2$ into $n^{2/5}$ smaller grids, each of size $n^{4/5} \times n^{4/5}$. Without loss of generality, assume both $n^{1/5}$ and $n^{4/5}$ are integers, and further assume $n^{1/5} = 3 \bmod 4$; otherwise we can consider a slightly smaller grid by the simple trick as at the beginning of Section 4.4.2. We shall use a random walk similar to Aaronson's in [2] as follows in each block, and change blocks after each step. Thus different blocks record different time.

For any time $t \in [n^{1/5}(n^{1/5}-1)]$, suppose $t = 2rn^{1/5} + t'$ where $r \in \{0, 1, ..., (n^{1/5}-3)/2\}$ and $t' \in \{1, 2, ..., 2n^{1/5}\}$. Let

$$u = \begin{cases} 0 & \text{if } t' \equiv 0, 1 \pmod 4 \\ n^{4/5} & \text{if } t' \equiv 2, 3 \pmod 4. \end{cases} \quad (4.90)$$

Let $block(t)$ as the small grid

$$\begin{cases} \{(x = (\lceil t'/2 \rceil - 1)n^{4/5} + x', y = 2rn^{4/5} + u + y') : x', y' \in [n^{4/5}]\} & \text{if } r \text{ is even} \\ \{(x = (n^{1/5} - \lceil t'/2 \rceil)n^{4/5} + x', y = 2rn^{4/5} + u + y') : x', y' \in [n^{4/5}]\} & \text{if } r \text{ is odd.} \end{cases} \quad (4.91)$$

The (x', y') is called the *offset* of (x, y). Now define the random walk as in the following Algorithm 4.2 and as depicted in Figure 4.3.

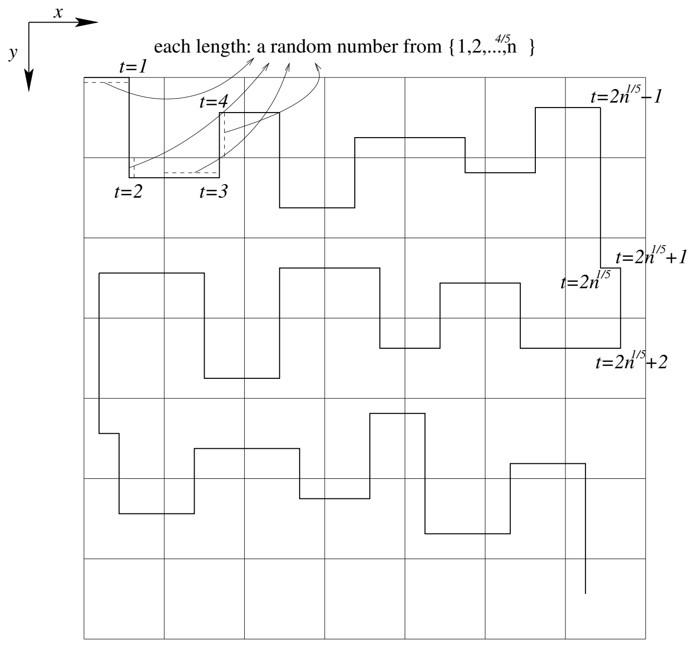

Figure 4.3: A different random walk in the 2 dimensional-grid

We then follow the same track as in the proof of Theorem 20. To get a reduction from Local Search on $[n]^2$ to the $Path_P$ problem, we define the function

$$f_X(v) = \begin{cases} l_{[n]^2}(v - (1,1)) & \text{if } v \notin set(X) \\ -2n^{4/5}(t-1) - (-1)^r x'_v + (-1)^{\lceil t'/2 \rceil} y'_v & \text{if } v \in set(X) \cap block(t). \end{cases} \tag{4.92}$$

Intuitively, the function value decreases along the path as before. But the decrement is not always by 1: each block has its fixed value setting. If for example the path passes through the block toward right and down (as in the first block), then the value $-x' - y'$ is used within the block. In this way, we do not need to know the length of the path segment from top to v to calculate each $f_X(v)$.

What we care about is still, as in Equality (4.27), the probability that the path X' passes another point x on X, under the condition that $X' \wedge Y = k'$. It is not hard to see that this probability is $\Theta(1)$ in general if x is in $block(k')$, and $\Theta(1/n^{4/5})$ otherwise (*i.e.*

Algorithm 4.2: A different random walk in the 2 dimensional grid.

```
1. Initially $(x, y) = (1, 1)$
2. for $t = 1, 2, ..., n^{1/5}(n^{1/5} - 1)$    (Suppose the current point is $(x, y)$ with offset $(x', y')$)
3.    if $t'$ is odd
4.       pick a random $x'' \in [n^{1/5}]$, move horizontally to the point in $block(t)$ with the offset $(x'', y')$
5.    else
6.       if $t' = 2n^{1/5}$ then c = 1 else c = 0
7.       pick a random $y'' \in [n^{1/5}]$, move vertically to the point in $block(t + c)$ with the offset $(x', y'')$
```

when x is in $block(t)$ for some $t > k'$). Thus by $L = \Theta(n^{2/5})$ we have

$$QLS([n]^2) = \Omega\left(n^{2/5}/\left(1 + n^{2/5}/\sqrt{n^{4/5}}\right)\right) = \Omega(n^{2/5}) \tag{4.93}$$

This completes the proof of Theorem 16.

Note that this random walk suffers from the fact that the "passing probability" is now $n^{4/5}$ times the "hitting probability". So for general d, we can get $RLS([n]^d) = \Omega(n^{d/(d+1)})$ and $QLS([n]^d) = \Omega(n^{d/(2d+1)})$, which only gives better results for QLS on the 2-dimensional grid.

4.5 New algorithms for Local Search on general graphs

In [7, 2], a randomized and a quantum algorithm for Local Search on general graphs are given as follows. Pick k random samplings over all the vertices, and find a vertex v in them with the minimum f-value.[7] Then roughly speaking, v is the N/k-minimum vertex over all the N vertices in G. Now we follow a *decreasing path* as follows. Find a neighbor of v with the minimum f-value, and continue this minimum-value-neighbor search process until

[7]For the minimum f-value finding procedure, The randomized algorithm in [7] just queries all these vertices and find the minimum, while the quantum algorithm in [2] uses the algorithm by Durr and Hoyer [32] based on Grover search [34] to get a quadratic speedup.

getting to a local minimum. Since v is the N/k-minimum vertex, any decreasing path from it has length no more than N/k. Thus we need $k + \delta N/k$ queries in the randomized case and $\sqrt{k} + \sqrt{\delta}N/k$ queries in the quantum case, and optimizing k achieves the performance of the algorithms mentioned in Section 4.1. We can see that the algorithms actually fall into the generic algorithm category (see Section 1), with the initial point picked as the best one over some random samples.

In this section, we give new randomized and quantum algorithms, which work better than this simple "random sampling + steepest descent" method when the graph expands slowly. Here the idea is that after finding the minimum vertex v of the k sampled points, we know that v is (roughly) the N/k-minimum vertex. Therefore, there must be a local optimum within the smaller range $\{u : d_G(u, v) \leq N/k\}$. So instead of following the decreasing path of v, we start over by doing the local search within this smaller range, and this procedure can be done recursively.

While this idea sounds simple and effective, there is one caveat here: a local minimum u in the smaller range may be not a local minimum in the original larger graph G, because u may have more neighbors in G. To deal with this difficulty, we will actually solve a stronger version of the local search problem: on the graph G, given a function $f : V \to \mathbb{R}$ and a vertex v, find a local optimum u *s.t.* $f(u) \leq f(v)$. Note that such u must exist; any decreasing path from v leads to a valid u. Also note that this problem is harder than the original local search problem: any algorithm for this new problem is an algorithm for the original one. A key property of this new problem is that it allows a recursion: given a small range S and a vertex v in it, suppose that any vertex w on the boundary of S is worse than v (*i.e.* $f(w) > f(v)$), then any local optimum u in S satisfying $f(u) \leq f(v)$ is also a local optimum in a larger range $S' \supseteq S$.

Now we describe the algorithm precisely as in Algorithm 4.3, with some notations as follows. For $G = (V, E)$, a given function $f : V \to \mathbb{N}$, a vertex $v \in V$ and a set $S \subseteq V$, let $n(v, S) = |\{u \in S : f(u) < f(v)\}|$. The boundary $B(S)$ of the set $S \subseteq V$ is defined by $B(S) = \{u \in S : \exists v \in V - S \textit{ suchthat } (u, v) \in E\}$. In particular, $B(V) = \emptyset$. A decreasing path from a vertex $v \in V$ is a sequence of vertices $v_0, v_1, ..., v_k$ such that $v_0 = v$, v_k is a local minimum and $f(v_{i+1}) = \min_{v:(v_i,v)\in E} f(v) < f(v_i)$ for $i = 0, ..., k-1$. We write $f(u) \leq f(S)$ if $f(u) \leq f(v)$ for all $v \in S$. In particular, it always holds that $f(u) \leq f(\emptyset)$. Suppose $d = \max_{u,v\in V} l(u,v)$ is the diameter of the graph, and $\delta = \max_{v\in V} |\{u : (u,v) \in E\}|$ is the max degree of the graph. In the following algorithm, the asymptotical numbers at the end of some command lines are the numbers of randomized or quantum queries needed for the step. For those commands without any number, no query is needed.

Define $c(k) = \max_{v\in V} |\{u : l(u,v) \leq k\}|$. Clearly, the expanding speed of a graph is upper bounded by $c(k)$. The following theorem says that the algorithm is efficient if $c(k)$ is small.

Theorem 26 *The algorithm outputs a local minimum with probability at least 1/2. The randomized algorithm uses* $O\left(\sum_{i=0}^{I-1} \frac{c(m_i)}{m_i} \log\log d\right)$ *queries in expectation, and the quantum algorithm uses* $O\left(\sum_{i=0}^{I-1} \sqrt{\frac{c(m_i)}{m_i}} (\log\log d)^{1.5}\right)$ *queries in expectation.*

In case that $c(k) = O(k^\alpha)$ *for some* $\alpha \geq 1$ *(*α *may be a function of* n*) and* $k = 1, ..., d$*, the expected number of queries that the randomized algorithm uses is* $O\left(\frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} \log\log d\right)$ *if* $\alpha > 1$ *and* $O(\log d \log\log d)$ *if* $\alpha = 1$*. The expected number of queries that the quantum algorithm use is* $O\left(\frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} (\log\log d)^{1.5}\right)$ *if* $\alpha > 1$ *and* $O(\log d \log logd)$ *if* $\alpha = 1$*.*

Several comments before proving the theorem:

1. $\lim_{\alpha\to 1} \frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} = \lim_{\alpha\to 1} \frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} = \log_2 d$

2. If $\alpha - 1 \geq \epsilon$ for some constant $\epsilon > 0$, then $\frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} = \Theta(d^{\alpha-1})$ and $\frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} = \Theta(d^{(\alpha-1)/2})$.

   If further the bound $c(k) = O(k^\alpha)$ is tight in the sense that $N = c(d) = \Theta(d^\alpha)$, then $RLS(G) = O\left(\frac{N}{d} \log\log d\right)$ and $QLS(G) = O\left(\sqrt{\frac{N}{d}} (\log\log d)^{1.5}\right)$.

3. For 2-dimensional grid, $d = \Theta(n)$ and $\alpha = 2$. Thus Theorem 18 follows immediately.

Proof We shall prove the theorem for the quantum algorithm. The analysis of the

Algorithm 4.3: New randomized and quantum algorithm for Local Search on a general graph $G = (V, E)$.

```
1. $m_0 = d$, $U_0 = V$;
2. $i = 0$;
3. while ($|m_i| > 10$) do
    (a) Randomly pick (with replacement) $\lceil \frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1} \rceil$ vertices from $U_i$, where $\epsilon_1 = 1/(10 \log_2 d)$;
    (b) Search the sampled vertices for one $v_i$ with the minimal $f$ value.
        - Randomized algorithm: query all the sampled vertices and get $v_i$. — $O\left(\frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1}\right)$
        - Quantum algorithm: use Durr and Hoyer's algorithm [32] with the error probability at most $\epsilon_2 = 1/(10 \log_2 d)$. — $O\left(\sqrt{\frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1}} \log \frac{1}{\epsilon_2}\right)$
    (c) if $i = 0$, then $u_{i+1} = v_i$;
        else if $f(u_i) \le f(v_i)$, then $u_{i+1} = u_i$;
            else $u_{i+1} = v_i$;
    (d) for $j = 1, 2, ...$
        i. Randomly pick $m_{ij} \in M_i = \{m : m_i/8 \le m \le m_i/2,\ |W(m)| \le 10|U_i|/m_i\}$, where $W(m) = \{w \in U_i : l(w, u_{i+1}) = m\}$. Let $W_{ij} = W(m_{ij})$.
        ii. Test whether $f(u_{i+1}) \le f(W_{ij})$
            - Randomized algorithm: query all vertices in $W_{ij}$. — $O(|W_{ij}|)$
            - Quantum algorithm: use Durr and Hoyer's algorithm [32] on $W_{ij}$ with the error probability at most $\epsilon_3 = 1/(200 \log_2 d)$. — $O\left(\sqrt{|W_{ij}|} \log \frac{1}{\epsilon_3}\right)$
        iii. If the answer is Yes, jump out of this for loop and go to Step 3e.
    (e) $J_i = j$, $m_{i+1} = m_{ij}$, $W_i = W_{ij}$, $U_{i+1} = \{u \in U_i : l(u, u_{i+1}) \le m_{i+1}\}$;
    (f) $i = i + 1$;
4. $I = i$;
5. Follow a decreasing path of $u_I$ to find a local minimum.
    - Randomized algorithm: in each step, query all the neighbors — $O(\delta)$
    - Quantum algorithm: in each step, use Durr and Hoyer's algorithm with the error probability at most 1/100 — $O(\sqrt{\delta})$
```

randomized algorithm is almost the same (and actually simpler). We say W_i is *good* if $f(u_{i+1}) \leq f(W_i)$. We shall first prove the following claim; the theorem then follows easily.

Claim 27 *For each $i = 0, 1, ..., I-1$, the following three statements hold.*

1. *$n(u_{i+1}, U_{i+1}) \leq n(u_{i+1}, U_i) \leq m_i/8 \leq m_{i+1}$ with probability $1 - \epsilon_1 - \epsilon_2$.*

2. *If $n(u_{i+1}, U_i) \leq m_i/8$, then W_i is good with probability $1 - \epsilon_3 J_i$, and $\mathbf{E}[J_i] \leq 2$.* [8]

3. *If $W_0, ..., W_i$ are all good, then $f(u_{i+1}) \leq f(B(U_{i+1}))$, and $u_{i+1} \notin B(U_{i+1})$.*

Proof 1: In Step 3a - 3c, denote by S the set of the $\lceil \frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1} \rceil$ sampled vertices in Step 3a. Let $a = \min_{u \in S} f(u)$, then $|\{v \in U_i : f(v) < a\}| \leq m_i/8$ with probability at least $1 - \epsilon_1$. The v_i found in Step 3b achieves the minimum in the definition of a with probability at least $1 - \epsilon_2$. Put the two things together, we have $n(v_i, U_i) \leq m_i/8$ with probability at least $1 - \epsilon_1 - \epsilon_2$. Since $f(u_{i+1}) \leq f(v_i)$ (by Step 3c), $U_{i+1} \subseteq U_i$ (by Step 3e) and $m_{i+1} \geq m_i/8$ (by Step 3(d)i), we have $n(u_{i+1}, U_{i+1}) \leq n(u_{i+1}, U_i) \leq n(v_i, U_i) \leq m_i/8 \leq m_{i+1}$ with probability at least $1 - \epsilon_1 - \epsilon_2$.

2: We say an m_{ij} is *good* if the corresponding W_{ij} is good, *i.e.* $f(u_{i+1}) \leq f(W_{ij})$. Note that for any $m_{ij} \in [m_i]$, we have $W_{ij} \subseteq U_i$, and also have $W_{ij} \cap W_{ij'} = \emptyset$ if $m_{ij} \neq m_{ij'}$. Therefore, if $n(u_{i+1}, U_i) \leq m_i/8$, then at most $m_i/8$ distinct m_{ij}'s in $[m_i]$ are *not* good. Also note that the number of distinct m_{ij}'s such that $|W(m_{ij})| > 10|U_i|/m_i$ is less than $m_i/10$. Therefore, $|M_i| \geq (\frac{3}{8} - \frac{1}{10})m_i > m_i/4$. So if $n(u_{i+1}, U_i) \leq m_i/8$, a random m_{ij} in M_i is good with probability at least $1/2$, and thus $\mathbf{E}[J_i] \leq 2$. Also the probability that all the Grover searches in Step 3(d)ii are correct is at least $1 - J_i \epsilon_3$.

3: We shall first prove $B(U_{i+1}) \subseteq B(U_i) \cup W_i$. In fact, any $s \in B(U_{i+1})$ satisfies that $s \in U_{i+1}$ and that $\exists t \in V - U_{i+1}$ such that $l(s,t) = 1$. Recall that $U_{i+1} \subseteq U_i$, so if $t \in V - U_i$, then $s \in B(U_i)$ by definition. Otherwise $t \in U_i - U_{i+1}$, and thus $t \in U_i$ and $l(t, u_{i+1}) > m_{i+1}$

[8]Since J_i is a random variable, the meaning of "W_i is good with probability $1 - \epsilon_3 J_i$" is that for each fixed $j = 1, 2, ...$, we have that W_i is good with probability $1 - \epsilon_3 j$ under the condition that $J_i = j$. Similar language is also used in the later part of the proof. Finally we will upper bound the probability of these random variables being large, and in the case that they are small, the error probability is small. This implies that the total error probability is small.

by the definition of U_{i+1}. Noting that $l(s, u_{i+1}) \leq m_{i+1}$ since $s \in U_{i+1}$, and that $l(s,t) = 1$, we have $l(s, u_{i+1}) = m_{i+1}$, which means $s \in W_i$. Thus for all $s \in B(U_{i+1})$, either $s \in B(U_i)$ or $s \in W_i$ holds, which implies $B(U_{i+1}) \subseteq B(U_i) \cup W_i$.

Applying the result recursively, we have $B(U_{i+1}) \subseteq B(U_0) \cup W_0 \cup ... \cup W_i = W_0 \cup ... \cup W_i$. Since we have $f(u_{i+1}) \leq f(u_i) \leq ... \leq f(u_1)$ (by Step 3c) and $f(u_{k+1}) \leq f(W_k)$ (for $k = 0, ..., i$) by the assumption that all W_k's are good, we know that $f(u_{i+1}) \leq f(W_0 \cup ... \cup W_i)$, which implies $f(u_{i+1}) \leq f(B(U_{i+1}))$.

For the other goal, $u_{i+1} \notin B(U_{i+1})$, it is sufficient to prove $u_{i+1} \notin B(U_i)$ and $u_{i+1} \notin W_i$. The latter is easy to see by the definition of W_i. For the former, we can actually prove $u_{k+1} \notin B(U_k)$ for all $k = 0, ..., i$ by induction on k. The base case of $k = 0$ is trivial because $B(U_0) = \emptyset$. Now suppose $u_k \notin B(U_{k-1})$. There are two cases of u_{k+1} by Step 3c. If $f(u_k) \leq f(v_k)$, then $u_{k+1} = u_k \notin B(U_{k-1})$ by induction. Again by the definition of W_{k-1} we know that $u_k \notin W_{k-1}$ and thus $u_{k+1} = u_k \notin B(U_k)$. The other case is $f(u_k) > f(v_k)$, then $u_{k+1} = v_k$, and therefore $f(u_{k+1}) = f(v_k) < f(u_k) \leq f(B(U_k))$ (by the first part in 3), which implies that $u_{k+1} \notin B(U_k)$. $\square$

(Continue the proof of Theorem 26) Now by the claim, we know that with probability at least $1 - I(\epsilon_1 + \epsilon_2) - \sum_{i=0}^{I-1} J_i \epsilon_3$, we will have

$$n(u_I, U_I) \leq m_I, \qquad f(u_I) \leq f(B(U_I)), \qquad u_I \notin B(U_I). \tag{4.94}$$

Note that the correctness of the algorithms follows from these three items. Actually, by the last two items, we know that any decreasing path from u_I is contained in U_I. Otherwise suppose $(u_I^0, u_I^1, ..., u_I^T)$ is a decreasing path from u_I (so $u_I^0 = u_I$), and the first vertex out of U_I is u_I^t, then $u_I^{t-1} \in B(U_I)$. Since $u_I^0 \notin B(U_I)$, we have $t-1 > 0$ and thus $f(u_I^{t-1}) < f(u_I)$, contradicting to $f(u_I) \leq f(B(U_I))$. Now together with the first item, we know that any decreasing path from u_I is no more than m_I long. Thus Step 5 will find a local minimum by following a decreasing path.

The error probability of the algorithm is $I(\epsilon_1 + \epsilon_2) + J\epsilon_3 + 10/100$, where $J = \sum_{i=0}^{I-1} J_i$. Since $\mathbf{E}[J] \leq 2I$, we know by Markov inequality that $J < 20I$ with probability at least

9/10. Since $\epsilon_1 = \epsilon_2 = 1/(10\log_2 d)$ and $\epsilon_3 = 1/(200\log_2 d)$, and noting that $I \le \log_2 d$ because $m_0 = d$ and $m_{i+1} \le \lceil m_i/2 \rceil$, the total error probability is less than 1/2.

We now consider the number of queries used in the i-th iteration. Note from Step 1 and Step 3e that $|U_i| \le c(m_i)$ for $i = 0, 1, ..., I-1$. So Step 3b uses

$$O\left(\sqrt{\frac{8|U_i|}{m_i}\log\log d}\log\log d\right) = O\left(\sqrt{\frac{c(m_i)}{m_i}}(\log\log d)^{1.5}\right) \tag{4.95}$$

queries. Also note from Step 3(d)i that $|W_{ij}| \le 10|U_i|/m_i$, so the number of queries used in Step 3d is $O(\sum_{j=1}^{J_i}\sqrt{c(m_i)/m_i}\log\log d)$, which has the expectation of $O(\sqrt{c(m_i)/m_i}\log\log d)$. Finally, Step 5 uses $O(\sqrt{\delta})$ queries. Note that $\delta = c(1) = O(c(m_I)/m_I)$ where m_I is a constant integer in the range $[6, 10]$. Altogether, the total expected number of queries used is

$$O\left(\left(\sum_{i=0}^{\log_2 d-1}\sqrt{c(m_i)/m_i}\right)(\log\log d)^{1.5}\right). \tag{4.96}$$

If $c(k) = O(k^\alpha)$ for some $\alpha \ge 1$ and $k = 1, ..., d$, then

$$\sum_{i=0}^{\log_2 d-1}\sqrt{\frac{c(m_i)}{m_i}} = \sum_{i=0}^{\log_2 d-1} m_i^{(\alpha-1)/2} = \sum_{i=0}^{\log_2 d-1}(d/2^i)^{(\alpha-1)/2} = \frac{d^\beta - 1}{1-2^{-\beta}} \tag{4.97}$$

where $\beta = (\alpha-1)/2$. This completes the proof for the quantum algorithm, except that in the case of $\alpha = 1$ we only have a quantum upper bound of $O(\log d(\log\log d)^{1.5})$. But note that the randomized algorithm uses $O(\log d \log\log d)$ queries (because of the saving at error probability controls). So if $\alpha = 1$, the quantum algorithm just uses the randomized one. $\square$

4.6 Open problems: remaining gaps

We list those grids on which the query complexities of Local Search still have gaps.

$d =$	2	3
old RLS	$[\Omega(1), O(n)]$	$[\tilde{\Omega}(\sqrt{n}), O(n^{\frac{3}{2}})]$
new RLS	$[\Omega(n^{\frac{2}{3}}), O(n)]$	$[\Omega(n^{\frac{3}{2}}/(\log n)^{\frac{1}{2}}), O(n^{\frac{3}{2}})]$
remaining gap	$n^{\frac{1}{3}}$ $(= N^{\frac{1}{6}})$	$(\log n)^{\frac{1}{2}}$ $(= (\log N)^{\frac{1}{2}})$

$d =$	2	3	4	5
old QLS	$[\Omega(n^{\frac{1}{4}}), O(n^{\frac{2}{3}})]$	$[\tilde{\Omega}(n^{\frac{1}{4}}), O(n)]$	$[\tilde{\Omega}(n^{\frac{1}{2}}), O(n^{\frac{4}{3}})]$	$[\tilde{\Omega}(n^{\frac{3}{4}}), O(n^{\frac{5}{3}})]$
new QLS	$[\Omega(n^{\frac{2}{5}}), O(n^{\frac{1}{2}})]$	$[\Omega(n^{\frac{3}{4}}), O(n)]$	$[\Omega(n^{\frac{6}{5}}), O(n^{\frac{4}{3}})]$	$[\Omega(n^{\frac{5}{3}}/(\log n)^{\frac{1}{3}}), O(n^{\frac{5}{3}})]$
remaining gap	$n^{\frac{1}{10}}(= N^{\frac{1}{20}})$	$n^{\frac{1}{4}}(= N^{\frac{1}{12}})$	$n^{\frac{2}{15}}(= N^{\frac{1}{30}})$	$(\log n)^{\frac{1}{3}}\ (= (\log N)^{\frac{1}{3}})$

Here $N = n^d$ is the number of the vertices in the grid.

Chapter 5

Quantum Query Complexities for a class of functions: weakly symmetric functions

The results in this chapter are from a work jointly done with Sun and Yao [71].

5.1 Background of Monotone Graph Properties and Aanderaa-Rosenberg Conjecture

In classical decision trees, considerable attention has been paid on the effect of symmetry on computational complexity. The most famous example is the confirmation of the Aanderra-Rosenberg Conjecture that all non-constant monotone graph properties on n vertices have (deterministic) decision tree complexity $\Theta(n^2)$ [60]. There are however many intriguing questions that remain open, in classical as well as quantum decision trees. The purpose of this chapter is to fill some of these gaps in the area of quantum complexity.

Let Γ be a group of permutations of $1, 2, \cdots, N$. Let $f : \{0,1\}^N \rightarrow \{0,1\}$ be any Boolean function *invariant* under Γ, i.e., $f(x_1, x_2, \cdots, x_N) = f(x_{\sigma(1)}, x_{\sigma(2)}, \cdots, x_{\sigma(N)})$ for all $\sigma \in \Gamma$. We are interested in how low the complexity can be for such functions. In the case of classical decision trees, let D_Γ be the minimum deterministic complexity for any non-constant Boolean function invariant under Γ, and let $D_\Gamma^{(M)}$ be the minimum deterministic complexity for any non-constant *monotone* Boolean function invariant under Γ. For quantum decision trees, let Q_Γ and $Q_\Gamma^{(M)}$ be the corresponding minimum complexities (allowing a small two-sided error probability).

Much is known about D_Γ and $D_\Gamma^{(M)}$ when Γ is a transitive group. See Section 5.4 for the definition of transitive groups. In particular, the Aanderra-Rosenberg Conjecture mentioned above can be phrased as $D_\Gamma^{(M)} = \Theta(N)$, where $N = \binom{n}{2}$ and Γ is the permutation group on the N edges of an n-vertex graph induced by the relabeling of the n vertices. It also has been proved that, for the same Γ, $D_\Gamma = \Theta(N^{1/2})$. We remark that it was not easy to construct a graph property with $O(N^{1/2})$ complexity, and the first non-trivial such example found was the property of being Scorpion graphs (see Section 5.2 for definition). Historically, it was actually first conjectured that $D_\Gamma = \Theta(N)$ for Γ corresponding to directed or undirected graph properties, but it was soon discovered that there are counterexamples in both cases: there is a directed graph property called Sink and an undirected graph property called Scorpion, both having $D(f) = O(\sqrt{N})$. Since neither of these two functions is monotone, it was then conjectured that $D_\Gamma^{(M)} = \Theta(N)$ for directed or undirected graph properties, which was even conjectured to be true for general transitive group Γ. The conjecture also went to the randomized case: is it true that $D_\Gamma^{(M)} = \Theta(N)$ for any transitive group Γ?

5.1.1 Is monotonicity still necessary in the quantum version of the conjecture?

Less is known about Q_Γ and $Q_\Gamma^{(M)}$. For the case when Γ is the set of all permutations on $1, 2, \cdots, N$, it is known that $Q_\Gamma = \Theta(\sqrt{N})$ and $Q_\Gamma^{(M)} = \Theta(\sqrt{N})$ [20]. It is generally believed that, for Γ corresponding to graph properties, $Q_\Gamma^{(M)} = \Theta(N^{1/2})$. Clearly, $O(N^{1/2})$ is an upper bound, and so far the best lower bound (by Santha and Yao (unpublished)) is $\Omega(N^{1/3})$. There do not seem to be published results on Q_Γ, and at first glance, it appears that Q_Γ may even be $\Theta(N^{1/2})$. The main result of this paper is to pin down Q_Γ.

Theorem 28 *For the* Γ *corresponding to graph properties,* $Q_\Gamma = \tilde{\Theta}(N^{1/4})$.

It is of interest to note that this is strictly below the quantum complexity of any non-constant *monotone* graph property. This means that the monotonicity condition is still important for the quantum case. The graph property used to establish the upper bound in Theorem 28 is again the Scorpion graph property. It is perhaps a natural choice, since the Scorpion graph property has low classical complexity, but the proof is nontrivial.

We have also obtained results for several other Γ. Let Γ_{directed} be the group of permutations of $1, 2, \cdots, N$ corresponding to the directed graph properties on n vertices (where $N = n(n-1)$). Let Γ_{circular} be the group of all cyclic shifts of $1, 2, \cdots, N$. A circular function is one whose value is invariant to any circular shift of the input indices.

Theorem 29 $Q_{\Gamma_{directed}} = \tilde{\Theta}(N^{1/4})$.

Theorem 30 $Q_{\Gamma_{circular}} = \tilde{\Theta}(N^{1/4})$.

The upper bound for Theorem 30 is established by a specially constructed Boolean function. In contrast, the upper bounds for Theorems 28 and 29 are achieved by some well-known Boolean functions.

The lower bound part of all the above 3 theorems can actually be generalized to the general transitive permutation group case.

Theorem 31 *For any transitive group* Γ*,* $Q_\Gamma = \Omega(N^{1/4})$.

5.2 Quantum algorithms for Scorpion

To prove the upper bound part of Theorems 28 and 29, we give $\tilde{O}(N^{1/4})$ algorithms for a graph property Scorpion and a directed graph property Sink, two problems classically interesting because their classical decision tree complexity is $O(n)$ [16, 17]. Note that $N = \binom{n}{2}$ for graph properties and $N = n(n-1)$ for directed graph properties, where n is the number of vertices. (We remark that instead of Sink, one can also use a directed graph version of Scorpion.) The definitions of Scorpion graphs and Sink graphs are as follows.

Definition 7 *An n-vertex undirected graph G is a Scorpion if there are three special vertices called Body, Tail and Sting, whose degrees are $n-2$, 2 and 1, respectively. Furthermore, the only vertex that Body does not connect to is Sting, and the only neighbor of Sting is Tail. (See Figure 1) We call any vertex other than these three special ones a Foot. Between any pair of Feet, there may or may not be an edge.*

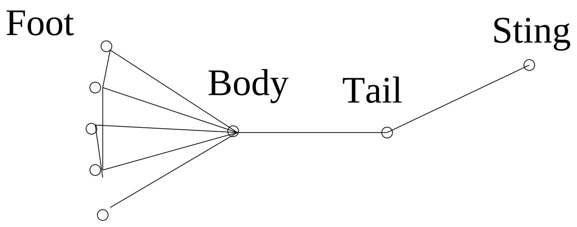

Figure 5.1: A Scorpion graph

An n-vertex directed graph G is a Sink if there is a special vertex with out-degree 0 and in-degree $n-1$. (See Figure 2)

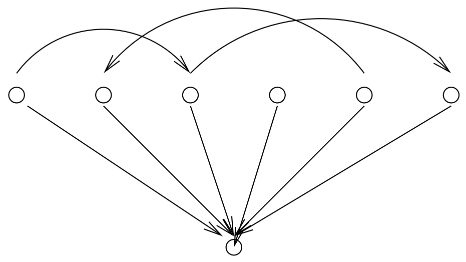

Figure 5.2: A Sink graph

Both the properties have classical decision tree complexity of $\Theta(n)$. The algorithm for Sink is as follows. From an arbitrary vertex v, do a depth first search. The first time we cannot find a new vertex, we get a candidate sink vertex. Check whether this candidate has out-degree 0 and in-degree $n-1$. Output "Sink" if yes and "not Sink" otherwise. The algorithm for Scorpion is a little more complicated, and we refer to [16, 17] for details of algorithms for these two properties.

We give quantum algorithms for Scorpion. Similar algorithms work for Sink too, and are omitted here. We first make some basic observations. The first one is about Grover search [34] of bounded error input, by Hoyer, Mosca and de Wolf [38].

Lemma 32 *(Hoyer, Mosca, de Wolf [38]) Given* n *elements, some of which may be marked, we wish to decide whether there is a marked element and locate one if any exists. Suppose that for any given element, one can decide with error probability less than* $1/3$ *whether it is marked, by a* q*-quantum-query subroutine. Then we can decide whether there exists a marked element (and output one if there is any) using an quantum algorithm with expected time and query complexity* $E = O(\sqrt{n/m}q)$*, where* m *is the unknown number of marked elements.*

The second lemma considers to find multiple marked elements rather than just one. Suppose we do not know the number m of marked elements, and are now required to find k marked ones. We use the following algorithm.

```
1. for $i = 0$ to $k - 1$
2.    for $j = 0$ to $\log(k/\epsilon)$
3.       run the algorithm in Lemma 32 for $2E_i$ steps,
4.       if we find a marked element during the time
5.          unmark the element (to avoid finding the same one in later rounds)
5.          go to line 1 with $i$ incremented
6.    output $m = i$ and halt.
```

In line 3, the E_i is the expected number of steps of the algorithm in Lemma 32 in the i-th iteration, and $E_i = O(q\sqrt{\frac{n}{m-i}})$.

We give a brief analysis of this algorithm. If $k \leq m$, then in the i-th outer iteration, the expected time to find a marked element is $\sum_{t=1}^{log(k/\epsilon)} 2tE_i/2^t < 4E_i = O(kq\sqrt{\frac{n}{m-i}})$ and the error probability is $2^{-t} = \epsilon/k$. So the expected value of the whole running time is

$$\begin{aligned}
& O(q(\sqrt{n/m} + \sqrt{n/(m-1)} + ... + \sqrt{n/(m-k+1)})) && (5.1)\\
= \;& O(q\sqrt{n} \cdot 2(\sqrt{m} - \sqrt{m-k})) && (5.2)\\
= \;& O(q\sqrt{n}k/\sqrt{m}) = O(kq\sqrt{n/m}) && (5.3)\\
= \;& O(q\sqrt{kn}), && (5.4)
\end{aligned}$$

and the whole error probability is less than ϵ. If $k > m$, then for the first m marked elements the analysis is the same as above and the expected time complexity is $O(q\sqrt{mn})$. After we find m marked elements in the $i = 0, ..., m-1$ iterations, we shall not find any marked element in the $i = m$ iteration, thus output m and halt by executing line 6. This iteration needs $O(q\sqrt{n}\log(k/\epsilon))$ time and the error probability is ϵ/k. Thus the whole time complexity in the $k > m$ case is $O(q\sqrt{mn} + q\sqrt{n}\log(k/\epsilon))$ and the error probability is less than ϵ. In summary, we have the following lemma.

Lemma 33 *The above algorithm outputs k marked elements if $k \leq m$, and outputs m and the m marked elements when $k > m$. In both cases the algorithm is correct with probability $1 - \epsilon$, and the time/query complexity is $O(q\sqrt{\min\{k, m\}n} + q\sqrt{n}\log(k/\epsilon))$.*

The third lemma is a basic observation that if we find a candidate for Body, Tail or Sting, then we can easily decide whether the given graph is a Scorpion or not.

Lemma 34 *Given a graph G and a vertex v, we can check whether G is a Scorpion graph with v being Body, Tail or Sting with error probability ϵ by $O(\sqrt{n}\log(1/\epsilon))$ queries.*

Proof We first check whether the degree of v is 1, 2, or n-2 by $O(\sqrt{n}\log(2/\epsilon))$ queries.

- Case 1: $deg(v) = 1$. We use Grover search to find the only neighbor u of v. Check that $deg(u) = 2$ and then find the other neighbor w of u. Finally, check that $deg(w) = n-2$. Output YES and w, u, v as Body, Tail and Sting and then halt if and only if G passes all these checks, and NO otherwise.

- Case 2: $deg(v) = n - 2$. We use Grover search to find the only vertex u which is not adjacent with v. Check $deg(u) = 1$ and then the rest part is similar with Case 1.
- Case 3: $deg(v) = 2$. Find the neighbors u, w of v. Check that $deg(u) = 1$ or $deg(u) = n - 2$. Then proceed as in Case 1 or Case 2.

In all three cases we should use Grover search for $\Theta(\log(1/\epsilon))$ times to let the error probability decrease to $\epsilon/2$, and thus the whole error probability is less than ϵ. $\Box$

The last lemma is the key idea of the algorithms that follow. Basically, they uses random sampling to find a vertex with low degree.

Definition 8 *For any vertex v in an undirected graph $G = (V, E)$ and any set $U \subseteq V$, define the set of neighbors of v in U as $N_U(v) = \{u : (u, v) \in E, u \in U\}$. We write $N(v)$ for $N_V(v)$.*

Lemma 35 *In a graph $G = (V, E)$, if we pick a random set T of $2\frac{u}{d}\log n$ vertices from a set U of u vertices, then we have*

$$\mathbf{Pr}[\forall v \in V,\ if\ N(v) \cap T = \phi,\ then\ |N_U(v)| \leq d] > 1 - 1/n.$$

This lemma says that, intuitively, if we pick a random subset T of U and T does not hit any neighbor of v (in U), then the degree of v in U is small with high probability.

Proof We have

$$\begin{aligned}
& \mathbf{Pr}[\forall v \in V,\ if\ N(v) \cap T = \phi,\ then\ |N_U(v)| \leq d] && (5.5)\\
= \ & 1 - \mathbf{Pr}[\exists v \in V,\ s.t.\ N(v) \cap T = \phi,\ and\ |N_U(v)| > d] && (5.6)\\
\geq \ & 1 - n \cdot \mathbf{Pr}[N(v) \cap T = \phi,\ and\ |N_U(v)| > d] && (5.7)\\
\geq \ & 1 - n \cdot \mathbf{Pr}[N(v) \cap T = \phi \mid |N_U(v)| > d] && (5.8)\\
> \ & 1 - n \cdot (\frac{u-d}{u})^{|T|} && (5.9)\\
> \ & 1 - n \cdot e^{-2\log n} && (5.10)\\
= \ & 1 - 1/n && (5.11)
\end{aligned}$$

as claimed. $\Box$

5.2.1 An $\tilde{O}(n^{3/4})$ algorithm

In this section, we give an $O(n^{3/4}\log^{1/4} n)$ algorithm. The basic idea is to find a low degree vertex by random sampling, and then search among the neighbors of this vertex for the body. The algorithm is as in Algorithm 5.1 box. The quantity following each step is the query complexity of that step.

Algorithm 5.1
Input: $G = (V, E)$.
Output: Tell whether G is a Scorpion, and if yes, output the Body, Tail and Sting.[a]

1. Let $d = \sqrt{n \log n}$. Pick a set U of $2\frac{n}{d}\log n$ random vertices. — 0
2. Use the algorithm in Lemma 32 to find a vertex v such that $N_U(v) = \emptyset$, and the error probability is less than 1/10. If we do not find one, then output "not Scorpion" and halt. — $O(\sqrt{n}\sqrt{\frac{n}{d}\log n}) = O(\frac{n}{\sqrt{d}}\sqrt{\log n})$
3. Use Lemma 34 to check whether G is a Scorpion with v being Body, Tail or Sting (and halt if it is). Make the error probability less than 1/10. — $O(\sqrt{n})$
4. Find all but up to d neighbors of v by Lemma 33 with error probability less than 1/10. If we find more than d neighbors of v, output FAILURE and halt. — $O(\sqrt{nd})$
5. Search among these neighbors of v for a vertex u with degree $n-2$ by Lemma 32. If we do not find u, then output "not Scorpion"; otherwise, use Lemma 34 to check whether G is a Scorpion with u being Body and output the result. — $O(\sqrt{nd})$

[a]There appear "FAILURE" outputs in the algorithms. That is to make the algorithms more clear: "FAILURE" is something unexpected and the overall probability of "FAILURE" is no more than a small constant.

The correctness and complexity of Algorithm 5.1 is given by the following theorem.

Theorem 36 *Given a graph G, Algorithm 5.1 decides whether G is a Scorpion graph and, if yes, output the Body, Tail and Sting. The time and query complexity of Algorithm 5.1 is $O(n^{3/4}\log^{1/4} n)$.*

Proof If G is a Scorpion, then there are at least two low degree vertices: Tail and Sting. So we can find a vertex v in Step 2 with probability at least 9/10, and we know by Lemma 35 that $deg(v) \leq d$ with probability at least $1 - 1/n$. Then if v is Tail or Sting we will find that G is a Scorpion by Lemma 34 in Step 3. Otherwise, v is a low degree Foot. So Body must be one of v's neighbors, all of which have already been found in Step 4. Thus with probability 9/10, Body is found in Step 5 and finally it uses Lemma 34 to output the correct answer. The total error probability is less than $3/10 + 1/n \leq 1/3$.

If G is not a Scorpion, then error can only be made in Steps 3 or Step 5, where we use Lemma 34 twice each of which has error probability less than 1/10.

Finally, The query complexity of the algorithm is $O(n^{3/4}\log^{1/4} n)$ by letting $d = \sqrt{n \log n}$.

□

5.2.2 Improvement

Now we give an improved algorithm, which is of $\tilde{O}(\sqrt{n})$ complexity. The key idea is that in Step 4 in Algorithm 5.1, instead of getting all neighbors of v, we again pick some random samples from these neighbors. We then find another vertex u which connects to none of these sample neighbors, then by Lemma 35, we know $N_{N(v)}(u)$ is small — in other words, v and u share few common neighbors. But note that Body still connects to both v and u if they are feet. So we can search among these common neighbors for Body. Directly following this idea gives an $O(n^{2/3})$ algorithm. To make full use of it, we apply it for about $\log n$ rounds as in Algorithm 5.2.

The following theorem actually shows Theorem 28.

Theorem 37 *Given a graph G, Algorithm 5.2 decides whether G is a Scorpion with error probability less than $1/3$ and, if yes, outputs the body, tail and sting. The time and query complexities of Algorithm 5.2 are $O(\sqrt{n}\log^2 n)$.*

Proof The proof is similar to the one for Algorithm 5.1. Note that by Lemma 35, after the i^{th} iteration of 1(b), with high probability (at least $1-1/n$) the number of common neighbors of $v_1, ..., v_{i+1}$ is no more than d_{i+1}. And after the total Step 1, with high probability (at least $7/10 - m/n$) the number of common neighbors of $v_1, v_2, ..., v_m$ (or $v_1, ..., v_i$ if we get Step 2 by jumping out of the i^{th} iteration of Step 1) is no more than d_m (or d_i, respectively).

Algorithm 5.2
Input: $G = (V, E)$.
Output: tell whether the G is a Scorpion, and if yes, output the Body, Tail and Sting.
Parameters: $m = \frac{1}{2}(\log n - \log\log n) - 1$, $d_i = (i-1)!n/m^i$ for $i = 1, ..., m$.

```
1. for $i = 0$ to $m - 1$ do
   (a) If $i = 0$, randomly pick a set $T_0$ of $k_0 = 2\frac{n}{d_1}\log n$ vertices from $V$.   — 0
       Else, randomly pick a set $T_i$ of $k_i = 2\frac{d_i}{d_{i+1}}\log n$ vertices from $\cap_{j=1}^{i} N(v_j)$ using
       Lemma 33 and making the error probability less than $\frac{1}{10m}$. If we cannot find so
       many vertices, jump out of this for loop and go to Step 2.   —
       $O(\sqrt{k_i in} + \sqrt{in}\log(k_i \cdot 10m)) = O(\sqrt{k_i in})$
   (b) Find a point $v_{i+1}$ such that $v_{i+1}$ connects to none of points in $T_i$ using Lemma
       32 and making the error probability less than $\frac{1}{10m}$. If no $v_{i+1}$ is found, output
       "not Scorpion" and halt.   —$O(\sqrt{k_i n}\log m)$
   (c) Use Lemma 34 to check whether $G$ is a Scorpion with $v_{i+1}$ being Body, Tail or
       Sting (and halt if yes), making the error probability less than $\frac{1}{10m}$.   —
       $O(\sqrt{n}\log m)$
2. If we get here by jumping out of the for loop at the $i^{th}$ iteration, then find all common
   neighbors of $v_1, ..., v_i$ using Lemma 33 with error probability less than 1/20.   —
   $O(\sqrt{k_i ni})$
   Otherwise, find all but no more than $d_m$ common neighbors of $v_1, ..., v_m$ using Lemma
   33 with error probability less than 1/20.   — $O(\sqrt{d_m nm})$
3. Use Lemma 32 to search among these common neighbors for a vertex $u$ with degree
   $n - 2$. If we do not find $u$, then output "not Scorpion"; otherwise, use Lemma 34 to
   check whether $G$ is a Scorpion and output the result.   — $\max\{O(\sqrt{d_m n}), O(\sqrt{k_i n})\}$
```

So if G is a Scorpion, then Body will be found finally in Step 3 with probability at least 6/10, if Tail or Sting is not found in Step 1(c) luckily (which makes the algorithm succeed even earlier). On the other hand, if G is not a Scorpion, then similar arguments as in the proof of Theorem 36 yield the small constant error probability result.

For the complexity, let us first calculate the cost of Step 1. Without loss of generality, suppose that all m iterations of Step 1 are done before we get to Step 2. The first iteration of Step 1 is of cost $O(\sqrt{k_0 n}\log m) = O(\frac{n}{\sqrt{d_1}}\sqrt{\log n}\log m)$ and the cost of the i^{th} iteration is $O(\sqrt{k_i in} + \sqrt{k_i n}\log m)$. Note that

$$\begin{aligned} k_i &= 2(d_i/d_{i+1})\log n && (5.12)\\ &= 2((i-1)!n/m^i)/(i!n/m^{i+1})\log n && (5.13)\\ &= \frac{2m}{i}\log n, && (5.14)\end{aligned}$$

thus by noting that $\sum_{i=1}^{m-1}\sqrt{k_i in} \geq \sum_{i=1}^{m-1}\sqrt{k_i n}\log m$, we have the cost of Step 1 as

$$\begin{aligned} & O\left(\sqrt{k_0 n\log m} + \sum_{i=1}^{m-1}(\sqrt{k_i in} + \sqrt{k_i n}\log m)\right) && (5.15)\\ = \;& O\left(\sqrt{k_0 n\log m} + \sum_{i=1}^{m-1}(\sqrt{k_i in})\right) && (5.16)\\ = \;& O\left(\sqrt{2mn\log n\log m} + \sum_{i=1}^{m-1}\sqrt{2mn\log n}\right) && (5.17)\\ = \;& O\left(m\sqrt{mn\log n}\right) && (5.18)\\ = \;& O\left(\sqrt{n}\log^2 n\right) && (5.19)\end{aligned}$$

since $m = \frac{1}{2}(\log n - \log\log n) - 1$.

Now we consider the cost of Step 2 and 3, which is $O(\sqrt{k_i ni} + \sqrt{d_m mn}) = O(\sqrt{n}\log n)$

because

$$\begin{aligned} d_m &= \frac{(m-1)!n}{m^m} && (5.20)\\ &= O\left(\frac{\sqrt{m}(m/e)^{m-1}n}{m^m}\right) && (5.21)\\ &= O\left(\frac{n}{\sqrt{m}}(1/e)^{m-1}\right) && (5.22)\\ &= O\left(\frac{n}{\sqrt{\log n}}\frac{1}{n/\log n}\right) && (5.23)\\ &= O\left(\sqrt{\log n}\right) && (5.24)\end{aligned}$$

As a result, the time and query complexities of Algorithm 5.2 are $O(\sqrt{n}\log^2 n)$. $\square$

We add some remarks about the Sink problem to end this section. The same algorithm can be applied to solve Sink. Now instead of finding a vertex with low degree, we are trying to find one with low out-degree. Here the sink point plays both the Sting role (by having no out-degree) and the Body role (by having $n-1$ in-degree).

5.3 One other symmetry group: circular functions

To show the upper bound part of Theorem 30, it is sufficient to construct a circular function f whose quantum query complexity is $\tilde{O}(N^{1/4})$.

Definition 9 *A function $f:\{0,1\}^N \to \{0,1\}$ is a circular function if any circular permutation of input indices does not change the function value. In other words, $f(x') = f(x)$ for any x and x' with $x' = x_{k+1}x_{k+2}...x_n x_1...x_k$ for some $1 \le k < n$.*

Now we give a particular circular function f. Basically, it is a variant of Sink. Here we only give the definition when $N = n^2$ for some integer n. The general case can be shown using similar algorithms and theorems.

Definition 10 *Let the function $f:\{0,1\}^N \to \{0,1\}$ be a Boolean function of N Boolean variables where $N = n^2$. For each input $x = x_1x_2...x_N$ we write it as an $n \times n$ matrix with row and column indices ranging over $\{0,1,...,n-1\}$, and the (i,j)-entry being x_{in+j+1}. We* *denote this matrix by M_x, or M if x is clear from the context. We use $M(i,j)$ to denote the (i,j) entry of the matrix M.*

We denote by $+_n$ and $-_n$ the addition and subtraction mod n, respectively. For example, $(n-1)+_n 1 = 0$ and $0 -_n 1 = n-1$.

Let $f(x) = 1$ if and only if M_x is of the following (i,j)-form for some $i, j \in \{0, 1, ..., n-1\}$: row i contains all 0 entries; in row $(i -_n 1)$, all the entries with column index greater than j are 0; in row $(i +_n 1)$, all the entries with column index less than j are 0; at last, all the entries in column j except $M_x(i,j)$ are 1.

$$\begin{array}{cccccccc}
 & 0 & ... & j-1 & j & j+1 & ... & n-1 \\
 & & & & & & & \\
0 & & & & 1 & & & \\
\vdots & & & & \vdots & & & \\
i-1 & & & & 1 & 0 & ... & 0 \\
i & 0 & ... & 0 & 0 & 0 & ... & 0 \\
i+1 & 0 & ... & 0 & 1 & & & \\
\vdots & & & & \vdots & & & \\
n-1 & & & & 1 & & &
\end{array} \tag{5.25}$$

Sometimes we say M is of $(i,*)$-form if M is of (i,j)-form for some j; we also say M is of $(*,j)$-form if M is of (i,j)-form for some i. The following facts are obvious, where the $D(f) = O(\sqrt{N})$ part can be shown by using an algorithm similar to the one for Sink described in Section 5.2.

Lemma 38 *The function f is a circular function, and $D(f) = O(\sqrt{N})$.*

Another key property of f is that if $f(x) = 1$, then any row except row i has at least one entry being 1, and exactly one column — column j — intersects all rows but row i with a 1-entry.

Before describing the algorithm, we construct subroutines analogous to those in Lemma 34.

Lemma 39 *Given x and row index i , we can decide whether M_x is in $(i,*)$-form with high probability by using $O(\sqrt{n})$ queries; symmetrically, given x and column index j , we can decide whether M_x is in $(*,j)$-form with high probability by using $O(\sqrt{n})$ queries.*

Proof An algorithm for the row case:

1. Check whether $M(i,1) = ... = M(i,n) = 0$ and if not, return NO.
2. Find j such that $M(i +_n 1, 1) = ... = M(i +_n 1, j-1) = 0$ and $M(i +_n 1, j) = 1$. If no j is found, return NO.
3. Check whether $M(i -_n 1, j+1) = ... = M(i -_n 1, n) = 0$ and $M(1,j) = ... = M(i-1,j) = M(i+1,j) = ... = M(n,j) = 1$. Return YES if so and NO otherwise.

An algorithm for the column case:

1. Check that only one entry is 0 in column j and assume $M(i,j) = 0$ if so. If not, return NO.

2. Check whether $M(i -_n 1, j+1) = ... = M(i -_n 1, n) = M(i +_n 1, 1) = ... = M(i +_n 1, j-1) = 0$ and all entries in row i are also 0's. Return YES if so and NO otherwise. □

Now we give an $O(\sqrt{n}\log^2 n)$ algorithm for f in the box Algorithm 5.3.

The following theorem validates Theorem 30 in Section 5.1. We omit the proof because it is almost the same as that for Algorithm 5.2.

Theorem 40 *Algorithm 5.3 decides f with high probability and the time and query complexity is $O(\sqrt{n}\log^2 n)$.*

We give some brief remarks on the case of N not being a perfect square to end this section. Let n to be the maximal integer such that $n^2 \leq N$. Again we write $x_1...x_N$ in the matrix form similar as in Definition 10, now with n columns and $n+1$ or $n+2$ rows, but

```
Algorithm 5.3
Input: $x$
Output: $f(x)$
Parameters: $m = \frac{1}{2}(\log n - \log\log n) - 1$, $d_i = (i-1)!n/m^i$ for $i = 1, ..., m$.

1. for $i = 0$ to $m - 1$ do
   (a) If $i = 0$, randomly pick a set $T_0$ of $k_0 = 2\frac{n}{d_1}\log n$ column indices $\{c_1, ..., c_{k_0}\}$. — 0
       Else, randomly pick a set $T_i$ of $k_i = 2\frac{d_i}{d_{i+1}}\log n$ column indices $\{c_1, ..., c_{k_i}\}$ such
       that $\forall c \in T_i$, $M(r_s, c) = 1$ for all $s \in [i]$ using Lemma 33 and making the error
       probability less than $\frac{1}{10m}$. If we cannot find so many columns, then jump out
       from this for loop and go to Step 2.                                  — $O(\sqrt{k_i i n})$
   (b) Find a row index $r_{i+1}$ such that $M(r_{i+1}, c_1) = ... = M(r_{i+1}, c_{k_i}) = 0$ using 32 and
       making the error probability less than $\frac{1}{10m}$. If no $r_{i+1}$ is found, output $f(x) = 0$
       and halt.                                                             — $O(\sqrt{k_i n})$
   (c) Check whether $M_x$ is in $(r_{i+1}, *)$-form by Lemma 39 with error probability less
       than $\frac{1}{10m}$. If YES, output $f(x) = 1$ and halt.                      — $O(\sqrt{n})$
2. If we get here by jumping out of the for loop at the $i^{th}$ iteration, then use Lemma
   33 to get all the columns $c$ such that $M(r_1, c) = ... = M(r_{i+1}, c) = 1$ with error
   probability less than $\frac{1}{20}$.                                           — $O(\sqrt{k_i n i})$
   Otherwise, get all but no more than $d_m$ columns $c$ such that $M(r_1, c) = ... =
   M(r_m, c) = 1$ using Lemma 33 with error probability less than $\frac{1}{20}$.     — $O(\sqrt{d_m n m})$
3. Search among these columns for a column $c$ such that $M_x$ is of $(*, c)$-form by the
   algorithm in Lemma 39 with error probability less than $\frac{1}{20}$. Output $f(x) = 1$ if we
   succeed and $f(x) = 0$ otherwise.                    — $\max\{O(\sqrt{d_m n}), O(\sqrt{k_i n})\}$
```

the last row may be not complete. Suppose the last entry in the last row is in the j_0-th column. We define $f(x) = 1$ if, for some circular permutation σ, $M_{\sigma(x)}$ is in the form of (5.25) with $j \leq j_0$. It can be shown that all the lemmas and theorem hold for this case.

5.4 Quantum lower bounds for all weakly symmetric functions

Before we prove Theorem 31, we remark that the lower bound part of Theorem 28 is easy to obtain by existing results. Turan showed that, for any graph property f, $s(f) \geq n/4$ [72] and Beals *et al.* showed $Q_2(f) \geq \sqrt{bs(f)}/4$ [20], which together imply $Q_2(f) \geq \sqrt{n}/8$ because of the trivial fact $bs(f) \geq s(f)$.

In this section we prove Theorem 31 using the quantum adversary method.

Definition 11 *A permutation group Γ on the set $\{1, 2, ..., N\}$ is transitive if, for any $i, j \in \{1, 2, ..., N\}$, there is a permutation $\sigma \in \Gamma$ such that $\sigma(i) = j$.*

A function invariant under a transitive permutation group is also called a *weakly symmetric function.* A simple but useful fact about any transitive group Γ is the following lemma in [60]. We denote by $w(x)$ the number of 1's in x, and by $\sigma(x)$ the string $x_{\sigma(1)}x_{\sigma(2)}...x_{\sigma(N)}$.

Lemma 41 *(Rivest and Vuillemin, [60]) If Γ is transitive, then for any $x \in \{0,1\}^N$ and any $i \in \{1, 2, ..., N\}$*

$$w(x) \cdot |\{\sigma(x) : \sigma \in \Gamma\}| = N \cdot |\{\sigma(x) : \sigma \in \Gamma, \sigma(x)_i = 1\}|. \tag{5.26}$$

For completeness, we first show the following classical folklore result.

Proposition 42 *For any N-ary function f invariant to a transitive group Γ, we have $C_0(f)C_1(f) \geq N$ and thus $D(f) \geq C(f) \geq \sqrt{N}$.*

Proof Let A and B be a 1 and 0-certificate with size $C_1(f)$ and $C_0(f)$, respectively. Let $\Gamma(B) = \{\sigma(B) : \sigma \in \Gamma\}$. Then for any $B' \in \Gamma(B)$, B' is also a 0-certificate assignment. So

$A \cap B' \neq \emptyset$, and thus

$$\sum_{B' \in \Gamma(B)} |A \cap B'| \geq |\Gamma(B)|. \tag{5.27}$$

By Lemma 41 we know that

$$|B| \cdot |\Gamma(B)| = N \cdot |\sigma(B) : \sigma \in \Gamma, i \in \sigma(B)|. \tag{5.28}$$

Therefore,

$$\begin{aligned} \sum_{B' \in \Gamma(B)} |A \cap B'| &= |A| \cdot |\sigma(B) : \sigma \in \Gamma, i \in \sigma(B)| && (5.29) \\ &= |A| \cdot \frac{|B| \cdot |\Gamma(B)|}{N}. && (5.30) \end{aligned}$$

Combining these two equalities, we have $|A| \cdot |B| \geq N$, *i.e.* $C_1(f)C_0(f) \geq N$. So $D(f) \geq C(f) \geq \sqrt{N}$. $\square$

Now the proof of Theorem 31 is as follows. We denote $\mathbf{0} = 00...0$. For any x, let $x^{(S)}$ be the string obtained from x by flipping all the x_i that $i \in S$.

Proof (of Theorem 31) Let f be a nontrivial function invariant under a transitive permutation group Γ. Without loss of generality, we assume that $f(\mathbf{0}) = 0$. Let B be a minimal subset such that $f(\mathbf{0}^{(B)}) = 1$, *i.e.* for any $B' \subsetneq B$, we have $f(\mathbf{0}^{(B')}) = 0$. Thus flipping any x_i where $i \in B$ changes the value of $f(\mathbf{0}^{(B)})$, therefore $bs(f) \geq |B|$.

Now we use the Theorem 4 to show that $Q_2(f) = \Omega(\sqrt{n/|B|})$. Let $X = \{\mathbf{0}\}$, $Y = \{\sigma(\mathbf{0}^{(B)}) : \sigma \in \Gamma\}$ and $R = X \times Y$. Then

$$m = \max_x |\{y : (x,y) \in R\}| = |Y|, \tag{5.31}$$

$$m' = \max_y |\{x : (x,y) \in R\}| = 1. \tag{5.32}$$

And

$$\begin{aligned} l &= \max_{x,i} |\{y : (x,y) \in R, x_i \neq y_i\}| && (5.33) \\ &= |\{\sigma(\mathbf{0}^{(B)}) : \sigma \in \Gamma, \sigma(\mathbf{0}^{(B)})_i = 1\}|, && (5.34) \end{aligned}$$

$$l' = \max_{y,i} |\{x : (x,y) \in R, x_i \neq y_i\}| = 1. \tag{5.35}$$

Thus by Theorem 4 and the above lemma, we have

$$Q_2(f) = \Omega(\sqrt{\frac{mm'}{ll'}}) = \Omega\left(\sqrt{\frac{|Y| \cdot 1}{\frac{|B||Y|}{N} \cdot 1}}\right) \tag{5.36}$$

$$= \Omega(\sqrt{N/|B|}) = \Omega(\sqrt{N/bs(f)}). \tag{5.37}$$

On the other side, we know $Q_2(f) = \Omega(\sqrt{bs(f)})$, so $Q_2(f) = \Omega(N^{1/4})$. $\square$

Chapter 6

Quantum Query Complexity *vs.* Other Complexity Measures

6.1 The quantum query complexity and the influence

For Boolean function $f : \{0,1\}^n \to \{0,1\}$, let μ_p be the distribution on $\{0,1\}^n$ such that $\mu_p(x) = p^{|x|}q^{n-|x|}$, where $q = 1 - p$ and $|x|$ is the number of 1's in x. In other words, it is the distribution that we pick each x_i independently being 1 with probability p. Define the influence of the i-th variable to be $\mathrm{inf}_i(f, p) = \mathbf{Pr}_{x\sim\mu_p}[f(x) \neq f(x^i)]$, where x^i is obtained from x by flipping the i-th variable of x. The total influence is the summation: $\mathrm{inf}(f, p) = \sum_i \mathrm{inf}_i(f, p)$.

Recently O'Donnell, Saks, Schramm, and Servedio [54] showed that

$$R(f) \geq \frac{\mathbf{Var}[f]}{pq \max_i \mathrm{inf}_i(f, p)}, \tag{6.1}$$

for any Boolean function f and any p. Another recent result by O'Donnell and Servedio [55] is

$$R(f) \geq pq\, \mathrm{inf}(f, p)^2, \tag{6.2}$$

for any *monotone* Boolean function f and any p. Using these two, one gets $R(f) \geq n^{4/3}/p_c^{1/3}$ for any monotone function f that is invariant to any permutation from a transitive group, where p_c is a critical probability, *i.e.* $\mathbf{E}_{x\sim\mu_{p_c}}[f(x)] = 1/2$. This is the first proof of $\Omega(n^{4/3})$ lower bound for all the monotone graph properties without using the graph packing technique as used in all previous work by Yao [76], King [42], Hajnal [35], and Chakrabarti and Khot [24].

It is natural to conjecture

$$Q_2(f) \geq \Omega\left(\sqrt{\frac{\mathbf{Var}[f]}{pq \max_i \inf_i(f,p)}}\right) \tag{6.3}$$

for any Boolean function f and

$$Q_2(f) \geq \Omega(\sqrt{pq}\inf(f,p)) \tag{6.4}$$

for any monotone Boolean function f. Shi showed (6.4) for $p = 1/2$ and any Boolean function [64].

In this note we prove (6.4) for any f (not necessarily monotone), generalizing Shi's result [64]. We will give two proofs; the first one uses the polynomial methods, but it only shows the case of monotone function, and it has a log factor loss. However, it implies that the lower bounds actually holds not only for the quantum query complexity but also for the approximate degree.

The second proof uses the quantum adversary method, and proves exactly the inequality (6.4) in a simple way. Unlike the result (6.2) and the first proof mentioned in the last paragraph, the condition of monotonicity is not needed the second proof. This implies that the quantum lower bound of $\Omega(n^{2/3})$ actually hold not only for monotone transitive functions, but for all balanced transitive functions, unless the widely believed conjecture that the quantum and randomized query complexities cannot have a super-quadratic gap is not true.

6.1.1 Proof by the polynomial method

Theorem 43 *For monotone f and any $p \in (0,1)$,*

$$Q_2(f) = \Omega\left(\frac{\sqrt{pq}\inf(f,p)}{\log n}\right) \tag{6.5}$$

where $q = 1 - p$.

Proof By simple probability amplification, we know that $Q_2(f)\log n \geq Q_{1/2n}(f)$, where $Q_{1/2n}(f)$ is defined like $Q_2(f)$, but with further requirement that the error is no more than $1/2n$ for any input. Denote by $\widetilde{deg}_{1/2n}(f)$ a lowest degree of the polynomial approximating f with error no more than $1/2n$ for any input. Since $Q_{1/2n}(f) = \Omega(\widetilde{deg}_{1/2n}(f))$, it is enough to show that $\widetilde{deg}_{1/2n}(f) = \Omega(\sqrt{pq}\inf(f,p))$.

We will use Bernstein's Inequality, which says that any polynomial $r(t)$ with degree d and $\|r\|_{[-1,1]} = 1$ has $d \geq \sqrt{1-t^2}|r'(t)|$, $\forall t \in (-1,1)$. Denote by f_ϵ the best polynomial to approximate f up to ϵ, and let $\phi_p(f_\epsilon) = \mathbf{E}_{x\sim\mu_p}[f_\epsilon(x)]$. We will use Bernstein's Inequality for $\phi_p(f_\epsilon)$. By some simple scaling $t = 2p-1$, we know that $\sqrt{1-t^2} = 2\sqrt{pq}$). So it is enough to lower bound $\frac{d\phi_p(f_\epsilon)}{dp}$ by $\inf(f,p)$.

Let $\vec{p} = (p_1, ..., p_n)$ and $\phi_{\vec{p}}(f_\epsilon) = \mathbf{E}_{x\sim\mu_p}[f_\epsilon(x)]$. Then $\frac{d\phi_p(f_\epsilon)}{dp} = \sum_i \frac{\partial\phi_{\vec{p}}(f_\epsilon)}{\partial p_i}|_{p_i=p}$ by chain law. We use $x_{[n]-i}$ to denote $x_1...x_{i-1}x_{i+1}...x_n$, and use $x_{[n]-i}\circ b$ to denote $x_1...x_{i-1}bx_{i+1}...x_n$ for $b \in \{0,1\}$. Fix $\epsilon = 1/2n$, then

$$\frac{\partial\phi_{\vec{p}}(f_\epsilon)}{\partial p_i} = (\partial\sum_x \mu_{\vec{p}}(x)f_\epsilon(x))/\partial p_i \tag{6.6}$$

$$= (\partial\sum_{x:x_i=1}\mu_{\vec{p}}(x_{[n]-i})p_i f_\epsilon(x) + \sum_{x:x_i=0}\mu_{\vec{p}}(x_{[n]-i})(1-p_i)f_\epsilon(x))/\partial p_i \tag{6.7}$$

$$= \sum_{x:x_i=1}\mu_{\vec{p}}(x_{[n]-i})f_\epsilon(x) - \sum_{x:x_i=0}\mu_{\vec{p}}(x_{[n]-i})f_\epsilon(x) \tag{6.8}$$

$$= \sum_{x_{[n]-i}}\mu_{\vec{p}}(x_{[n]-i})(f_\epsilon(x_{[n]-i}\circ 1) - f_\epsilon(x_{[n]-i}\circ 0)) \tag{6.9}$$

$$= \sum_{x_{[n]-i}\nsim i}\mu_{\vec{p}}(x_{[n]-i})\left[f_\epsilon(x_{[n]-i}\circ 1) - f_\epsilon(x_{[n]-i}\circ 0)\right] \tag{6.10}$$

$$+ \sum_{x_{[n]-i}\sim i}\mu_{\vec{p}}(x_{[n]-i})\left[(f_\epsilon(x_{[n]-i}\circ 1) - f_\epsilon(x_{[n]-i}\circ 0)\right] \tag{6.11}$$

where $x_{[n]-i} \sim i$ means $x_{[n]-i}$ is sensitive at i, *i.e.* $f(x_{[n]-i}\circ 1) \neq f(x_{[n]-i}\circ 0)$. Since f_ϵ approximates f, we know that for $x_{[n]-i} \nsim i$, $-2\epsilon \leq f_\epsilon(x_{[n]-i}\circ 1) - f_\epsilon(x_{[n]-i}\circ 0) \leq 2\epsilon$. And for $x_{[n]-i} \sim i$, we have $f(x_{[n]-i}\circ 1) = 1$ and $f(x_{[n]-i}\circ 0) = 0$ because f is monotone, and thus $f_\epsilon(x_{[n]-i} \circ 1) - f_\epsilon(x_{[n]-i} \circ 0) \geq 1 - 2\epsilon$. Therefore,

$$\frac{\partial \phi_{\vec{p}}(f_\epsilon)}{\partial p_i} \geq \sum_{x_{[n]-i} \not\sim i} \mu_{\vec{p}}(x_{[n]-i})(-2\epsilon) + \sum_{x_{[n]-i} \sim i} \mu_{\vec{p}}(x_{[n]-i})(1-2\epsilon). \tag{6.12}$$

Note that $\sum_{x_{[n]-i} \sim i} \mu_{\vec{p}}(x_{[n]-i})|_{\vec{p}=(p,\ldots,p)}$ is nothing but $\inf_i(f,p)$, so

$$\frac{\partial \phi_{\vec{p}}(f_\epsilon)}{\partial p_i}|_{\vec{p}=(p,\ldots,p)} \geq (1 - \inf_i(f,p))(-2\epsilon) + \inf_i(f,p)(1-2\epsilon) = \inf_i(f,p) - 2\epsilon. \tag{6.13}$$

Now

$$\frac{d\phi_p(f_\epsilon)}{dp} = \sum_i \frac{\partial \phi_{\vec{p}}(f_\epsilon)}{\partial p_i}|_{\vec{p}=(p,\ldots,p)} \geq \sum_i (\inf(f,p) - 2\epsilon) = \inf(f,p) - 2n\epsilon = \inf(f,p) - 1. \tag{6.14}$$

□

6.1.2 Proof by the quantum adversary method

Basically, the quantum adversary method first picks a relation, *i.e.* a set of 0 and 1 input pairs, and assign a weight to each pair. It turns out that the initial total weight decreases by a constant fraction after the computation. So by upper bounding the average progress of one query, we can give a lower bound of the number of queries.

Theorem 44 *We have for any Boolean function f and any $p \in (0,1)$ that*

$$Q_2(f) \geq \Omega(\sqrt{pq}\inf(f,p)) \tag{6.15}$$

Proof Consider the set $\{(x, x^{(i)}) : f(x) \neq f(x^{(i)})\}$. Put weight $w(x, x^{(i)}) = p(x)$. Use $x \sim i$ to denote that $f(x) \neq f(x^i)$. Let $|\psi_x^{(k)}\rangle$ be the state after exactly k queries, and let $\delta_k = \sum_{x,i:x\sim i} p(x)|\langle\psi_x^{(k)}|\psi_{x^{(i)}}^{(k)}\rangle|$. Then

$$\delta_0 = \sum_{x,i:x\sim i} p(x) \cdot |\langle\psi_x^{(0)}|\psi_{x^{(i)}}^{(0)}\rangle| = \sum_i \mathbf{Pr}_x[x \sim i] = \sum_i \inf_i(f,p) = \inf(f,p). \tag{6.16}$$

and $\delta_T = \sum_{x,i:x\sim i} p(x) \cdot |\langle\psi_x^{(T)}|\psi_{x^{(i)}}^{(T)}\rangle| \leq \epsilon \inf(f,p)$. The standard analysis of oracle operation tells us that the average progress $|\delta_k - \delta_{k+1}| = \sum_{x,i:x\sim i} p(x) 2 \left|\langle P_i\psi_x^{(k)}|P_i\psi_{x^{(i)}}^{(k)}\rangle\right|$, where P_i is the projector onto the subspace spanned by index i, *i.e.* $P_i(\sum_{j,z}\alpha_{j,z}|j,z\rangle) = \sum_z \alpha_{i,z}|i,z\rangle$. Therefore,

$$|\delta_k - \delta_{k+1}| \le 2 \sum_{x,i:x\sim i} p(x) \left\| P_i|\psi_x^{(k)}\rangle \right\| \cdot \left\| P_i|\psi_{x^{(i)}}^{(k)}\rangle \right\| \tag{6.17}$$

$$\le 2\sqrt{\left(\sum_{x,i:x\sim i} p(x) \left\| P_i|\psi_x^{(k)}\rangle \right\|^2\right)\left(\sum_{x,i:x\sim i} p(x) \left\| P_i|\psi_{x^{(i)}}^{(k)}\rangle \right\|^2\right)} \tag{6.18}$$

where both steps are by Cauchy-Schwartz Inequality. Furthermore, let $y = x^{(i)}$ and the above quantity is equal to

$$= 2\sqrt{\left(\sum_x p(x) \sum_{i:x\sim i} \left\| P_i|\psi_x^{(k)}\rangle \right\|^2\right)\left(\sum_{y,i:y\sim i} p(y^{(i)}) \left\| P_i|\psi_y^{(k)}\rangle \right\|^2\right)} \tag{6.19}$$

$$\le 2\sqrt{\sum_{y,i:y\sim i} p(y^{(i)}) \left\| P_i|\psi_y^{(k)}\rangle \right\|^2} \tag{6.20}$$

$$= 2\sqrt{\sum_{y,i:y\sim i, y_i=1} p(y)\frac{q}{p} \left\| P_i|\psi_y^{(k)}\rangle \right\|^2 + \sum_{y,i:y\sim i, y_i=0} p(y)\frac{p}{q} \left\| P_i|\psi_y^{(k)}\rangle \right\|^2} \tag{6.21}$$

$$\le 2\sqrt{\frac{q}{p} + \frac{p}{q}} \tag{6.22}$$

$$\le \frac{2}{\sqrt{pq}} \tag{6.23}$$

So $T \ge \Omega(\sqrt{pq}\ \inf(f,p))$. $\square$

6.2 The quantum query complexity and the quantum communication complexity

Recall that tuple search is to find a tuple satisfying some pre-defined relation. We will now study the scenario that the components in the desired tuple are distributed. For simplicity, let us just examine the case of binary relation. Suppose that Alice has input $x_1, ..., x_n$ and Bob has $y_1, ..., y_n$. Alice can access her input $x_1, ..., x_n$ only by quantum queries, and she cannot access Bob's input $y_1, ..., y_n$. Symmetric rules apply to Bob. They want to search for the unique pair of (x_i, y_j) in some given relation R, by using communications.[1] In other words, the model is the same as the one used to study quantum communication complexity, except that the two parties access their respective inputs by quantum queries and we care about the number of queries they make. So there are two natural resources to consider: the number of queries, and the number of qubits in the communication. The former is about quantum query complexity as studied above, and the latter is about quantum communication complexity, introduced by Yao [77] as a generalization of the classical communication complexity (also introduced by Yao [75]; see [45] as an excellent textbook) and extensively studied since then (see [26] for a survey). As far as we know, all previous work considers one of these two problems. [2] For example, Ambainis [9] and the first part of this thesis consider the quantum query complexity; Buhrman, Cleve and Wigderson [21] show an $O(\sqrt{n}\log n)$ upper bound of quantum communication complexity of Disj, later improved by Hoyer and de Wolf to $O(\sqrt{n}c^{\log^* n})$ [37] and finally to $O(\sqrt{n})$ by Aaronson and Ambainis [4], matching the $\Omega(\sqrt{n})$ lower bound shown by Razborov [58]. Since query and communication are both well-studied resources, it is natural to study both of them simultaneously, and see how they interact with each other.

We can use a protocol similar to the one shown by Buhrman, Cleve and Wigderson [21], but it makes $\Theta(n)$ queries, which is higher than the optimal $\Theta(n^{2/3})$ value. We can also have a protocol achieving the optimal quantum query complexity, but the number of communication qubits is asymptotically more than the optimal $\widetilde{\Theta}(\sqrt{n})$ value. So there seems to be a tradeoff between the quantum query computation and the quantum communication. This paper gives one tradeoff result as follows. For a protocol P computing function f, denote by $q(P)$ the number of quantum queries and by $c(P)$ the number of communication qubits.

Theorem 45 *Let $f =$ Unique 2-Subset Finding. For any given $q_0 \in (n^{2/3}, n)$, there*

[1]If the R is the Equality relation, then the problem is related to Disj, a well-studied function. But we should note that Disj is to decide whether two subsets of an n-element set intersect, while here the distributed search problem is to decide whether two n-element sets intersect.

[2]Some papers study yet other resources. For example, paper [40] gives a lower bound of the tradeoff between communication complexity and round complexity.

exists a protocol P with $q(P) = q_0$ and $c(P) = O(\frac{n^2 \log n}{q_0^{3/2}})$.

In other words, we have a family of protocols with $q(P)^{3/2} \cdot c(P) = O(n^2 \log n)$. This implies that we can pay more for quantum queries to save on quantum communication, and vice versa.

In this section we prove Theorem 45 by giving a family of protocols achieving the tradeoff result. Note that in Algorithm 3.3, both the preparation of the initial state $|\psi_{start}\rangle$ in Step 1 and the Quantum Walks in Step 2(b) can be done distributively. So it naturally induces a communication protocol as follows.

Protocol 6.1: for distributed Unique 2-Subset Finding
Input: $x_1, ..., x_N \in [M]$. $J_1, J_2 \subseteq [N]$, $|J_1| = m, |J_2| = n$. $R \subseteq [M] \times [M]$ such that there is at most one $(x_{j_1}, x_{j_2}) \in R$ with $j_1 \in J_1$, $j_2 \in J_2$ and $j_1 \neq j_2$.
Output: The unique pair (j_1, j_2) if it exists; otherwise reject.

1. Alice sets up her initial state $|\psi_a\rangle = \frac{1}{\sqrt{\binom{n}{r_1}(n-r_1)}} \sum_{S_1 \subseteq J_1, |S_1| = r_1, i_1 \in J_1 - S_1} |S_1, x_{S_1}, i_1\rangle$ in her register R_a
   Bob sets up his initial state $|\psi_b\rangle = \frac{1}{\sqrt{\binom{n}{r_2}(n-r_2)}} \sum_{S_2 \subseteq J_2, |S_2| = r_2, i_2 \in J_2 - S_2} |S_2, x_{S_2}, i_2\rangle$ in his register R_b
2. Do $\Theta(\frac{n}{\sqrt{r_1 r_2}})$ times
   (a) Bob sends R_b (*i.e.* all his qubits) to Alice.
   (b) Alice checks whether $(j_1, j_2) \in S_1 \times S_2$. If yes, do the following phase flip: $|S_1, x_{S_1}, i_1, S_2, x_{S_2}, i_2\rangle \to -|S_1, x_{S_1}, i_1, S_2, x_{S_2}, i_2\rangle$.
   (c) Alice sends R_b back to Bob.
   (d) Alice does $\lceil \frac{\pi}{4}\sqrt{r_1} \rceil$ times **Quantum Walk** on S_1 in J_1.
   Bob does $\lceil \frac{\pi}{8}\sqrt{r_2} \rceil$ times **Quantum Walk** on S_2 in J_2.
3. Bob does the measurement and outputs the corresponding result.

The correctness of the protocol is obvious because it is essentially the same as Algorithm 3.3. We now analyze the complexity. The number of queries is the same as that of Algorithm 3.3, *i.e.* $q(P) = \Theta(r_1 + r_2 + \frac{n}{\sqrt{r_1 r_2}}(\sqrt{r_1} + \sqrt{r_2})) = \Theta(r_1 + r_2 + n(1/\sqrt{r_1} + 1/\sqrt{r_2}))$. The number of communication qubits of this protocol is $c(P) = \Theta(\frac{n}{\sqrt{r_1 r_2}} r_2 \log n) = \Theta(\sqrt{\frac{r_2}{r_1}} n \log n)$. If $t = r_1/r_2 \geq 1$, then $q(P) = \Theta(r_1 + n/\sqrt{r_2}) = \Theta(t r_2 + n/\sqrt{r_2}) \geq \Theta(t^{1/3} n^{2/3})$, and the equality is achieved when $r_2 = (n/t)^{2/3}$. So for any given $q_0 \in (n^{2/3}, n)$, let $r_1 = q_0$ and $r_2 = n^2/q_0^2$, then $q(P) = \Theta(q_0)$ and $c(P) = \Theta(\frac{n^2 \log n}{q_0^{3/2}})$.

Chapter 7

On the Power of the Quantum Adversary Method

So far, we have used the quantum adversary method to show many quantum lower bounds. These include those for specific problems, such as the three graph properties in Section 3.1 and the Local Search problems in Chapter 4, for a class of functions as a whole, such as the transitive functions in Section 5.4, and in terms of other complexity measures, such as influence in Section 6.1. The quantum adversary method has also been used on many other problems (see [39] for a survey). Some applications are not hard, while some involve many other techniques, like random walk in the proof of Local Search.

Given the usefulness of the method, it is interesting to answer the following several questions:

1. Is it tight? That is, can we always use the quantum adversary method to prove tight lower bounds?

2. Is there any limitation on the method?

3. The polynomial method is also used to prove some lower bounds. Are the two methods comparable in power?

In this chapter, we will answer all these questions. We will show a number of limitations in terms of certificate complexity, a well-studied complexity measure. These make it very easy to judge whether it is possible to use the method to achieve some lower bounds. Also, these give negative answers to the question 1 and 3.

The results in this chapter are from paper [78].

Related work: In [70], Szegedy independently proved the same limit $\sqrt{N \cdot C_-(f)}$ for partial functions (as in Theorem 46), but it is for the quantum adversary method in [14]. Szegedy then independently proved the same limit $\sqrt{C_0(f)C_1(f)}$ for total functions (as in Theorem 51). Also, after a preliminary version of the paper [78] posted on arXiv (quant-ph/0311060), Laplante and Magniez posted the report quant-ph/0311189 (a preliminary version of [46]), giving the limit of $\sqrt{N \cdot C_-(f)}$ by a totally different argument (by Kolmogorov complexity).

7.1 A limitation of the quantum adversary method for general partial functions

In this section, we show a limitation of the quantum adversary method (in its version Theorem 6) for general partial Boolean functions. Since all the versions of the quantum adversary methods are equivalent in power [69], this implies the same limitation for all the versions of the method.

The limitation is in terms of certificate complexity, a well-studied complexity measure. Recall that $C_-(f) = \min\{C_0(f), C_1(f)\}$

Theorem 46 *For any N-ary Boolean function f, we cannot use the quantum adversary method Theorem 6 to get a lower bound better than $\sqrt{N \cdot C_-(f)}$.*

Proof Actually we prove a stronger result: for any (X, Y, R, u, v, w) as in Theorem 6,

$$\min_{(x,y)\in R, i\in[N]} \frac{w_x w_y}{u_{x,i} v_{y,i}} \leq NC_-(f). \tag{7.1}$$

Without loss of generality, we assume that $C_-(f) = C_0(f)$, and $X \subseteq f^{-1}(0)$ and $Y \subseteq f^{-1}(1)$. We can actually further assume that $R = X \times Y$, because otherwise we just let

$R' = X \times Y$, and set new weight functions as follows.

$$u'(x,y,i) = \begin{cases} u(x,y,i) & (x,y) \in R \\ 0 & otherwise, \end{cases} \qquad v'(x,y,i) = \begin{cases} v(x,y,i) & (x,y) \in R \\ 0 & otherwise, \end{cases}$$

$$w'(x,y) = \begin{cases} w(x,y) & (x,y) \in R \\ 0 & otherwise. \end{cases}$$

Then it is easy to see that it satisfies Definition 6, so it is also a weight scheme. And for these new weight functions, we have $u'_{x,i} = \sum_{y:(x,y)\in R', x_i \neq y_i} u'(x,y,i) = \sum_{y:(x,y)\in R, x_i \neq y_i} u(x,y,i) = u_{x,i}$ and similarly $v'_{y,i} = v_{y,i}$ and $w'_x = w_x, w'_y = w_y$.[1] It follows that $\frac{w_x w_y}{u_{x,i} v_{y,i}} = \frac{w'_x w'_y}{u'_{x,i} v'_{y,i}}$, thus we can use (X', Y', R', u', v', w') to derive the same lower bound as we use (X, Y, R, u, v, w).

So we now suppose $R = X \times Y$ and prove that $\exists x \in X, y \in Y, i \in [N]$, such that

$$w_x w_y \leq N \cdot C_0(f)\ u_{x,i} v_{y,i}, \tag{7.2}$$

Suppose the claim is not true. Then for all $x \in X, y \in Y, i \in [N]$, we have

$$w_x w_y > N \cdot C_0(f)\ u_{x,i} v_{y,i}. \tag{7.3}$$

We first fix i for the moment. And for each $x \in X$, we fix a smallest certificate set CS_x of f on x. Clearly $|CS_x| \leq C_0(f)$. We sum (7.3) over $\{x \in X : i \in CS_x\}$ and $\{y \in Y\}$. Then we get

$$\left(\sum_{x \in X:\ i \in CS_x} w_x \right) \left(\sum_{y \in Y} w_y \right) > N \cdot C_0(f) \left(\sum_{x \in X:\ i \in CS_x} u_{x,i} \right) \left(\sum_{y \in Y} v_{y,i} \right). \tag{7.4}$$

Note that $\sum_{y \in Y} w_y = \sum_{x \in X, y \in Y} w(x,y) = \sum_{x \in X} w_x$, and that $\sum_{y \in Y} v_{y,i} = \sum_{x \in X, y \in Y: x_i \neq y_i} v(x,y,i) =$

[1]Note that the function values of u', v', w' are zero when $(x,y) \neq R$, which does not conform to the definition of a weight scheme. But actually Theorem 3 also holds for $u \geq 0, v \geq 0, w \geq 0$ as long as $u_{x,i}, v_{y,i}, w_x, w_y$ are all strictly positive for any x, y, i. This can be seen from the proof of Alb_4 in Section 4.

$\sum_{x\in X} v_{x,i}$ where $v_{x,i} = \sum_{y\in Y: x_i\neq y_i} v(x,y,i)$. Inequality (7.4) now becomes

$$\begin{aligned}
\left(\sum_{x\in X:\ i\in CS_x} w_x\right)\left(\sum_{x\in X} w_x\right) &> N\cdot C_0(f)\left(\sum_{x\in X:\ i\in CS_x} u_{x,i}\right)\left(\sum_{x\in X} v_{x,i}\right) && (7.5)\\
&\geq N\cdot C_0(f)\left(\sum_{x\in X:\ i\in CS_x} u_{x,i}\right)\left(\sum_{x\in X:\ i\in CS_x} v_{x,i}\right) && (7.6)\\
&\geq N\cdot C_0(f)\left(\sum_{x\in X:\ i\in CS_x} \sqrt{u_{x,i}v_{x,i}}\right)^2 && (7.7)
\end{aligned}$$

by the Cauchy-Schwartz Inequality. We further note that

$$\begin{aligned}
u_{x,i}v_{x,i} &= \left(\sum_{y\in Y: x_i\neq y_i} u(x,y,i)\right)\left(\sum_{y\in Y: x_i\neq y_i} v(x,y,i)\right) && (7.8)\\
&\geq \left(\sum_{y\in Y: x_i\neq y_i} \sqrt{u(x,y,i)v(x,y,i)}\right)^2 && (7.9)\\
&\geq \left(\sum_{y\in Y: x_i\neq y_i} w(x,y)\right)^2 && (7.10)\\
&= w_{x,i}^2 && (7.11)
\end{aligned}$$

where we define $w_{x,i} = \sum_{y\in Y: x_i\neq y_i} w(x,y)$. Thus

$$\left(\sum_{x\in X:\ i\in CS_x} w_x\right)\left(\sum_{x\in X} w_x\right) > N\cdot C_0(f)\left(\sum_{x\in X:\ i\in CS_x} w_{x,i}\right)^2. \qquad (7.12)$$

Now we sum (7.12) over $i = 1, ..., N$, and note that

$$\sum_i \sum_{x\in X:\ i\in CS_x} w_x = \sum_{x\in X}\sum_{i: i\in CS_x} w_x \leq C_0(f)\sum_{x\in X} w_x \qquad (7.13)$$

because $|CS_x| \leq C_0(f)$ for each x. We have

$$\left(\sum_{x\in X} w_x\right)^2 > N\sum_{i=1}^{N}\left(\sum_{x\in X:\ i\in CS_x} w_{x,i}\right)^2. \qquad (7.14)$$

By the convexity of squaring (or by the Cauchy-Schwartz Inequality),

$$N(a_1^2 + ... + a_N^2) \geq (a_1 + ... + a_N)^2, \qquad (7.15)$$

we have

$$\left(\sum_{x\in X} w_x\right)^2 > \left(\sum_{x\in X, i\in[N]:\ i\in CS_x} w_{x,i}\right)^2 \tag{7.16}$$

$$= \left(\sum_{x\in X, i\in[N], y\in Y:\ i\in CS_x, x_i\neq y_i} w(x,y)\right)^2 \tag{7.17}$$

$$= \left(\sum_{x\in X, y\in Y}\ \sum_{i\in[N]:\ i\in CS_x, x_i\neq y_i} w(x,y)\right)^2 . \tag{7.18}$$

But by the definition of certificate, we know that for any x and y there is at least one index $i \in CS_x$ such that $x_i \neq y_i$. Therefore, we derive an inequality

$$\left(\sum_{x\in X} w_x\right)^2 > \left(\sum_{x\in X, y\in Y} w(x,y)\right)^2 = \left(\sum_{x\in X} w_x\right)^2 \tag{7.19}$$

which is a contradiction, as desired. □

We add some comments about this upper bound. First, this bound looks weak at first glance because the $\sqrt{N}$ factor seems too large. But in fact it is necessary. Consider the problem of Invert A Permutation[10] [2] , where $C_0(f) = C_1(f) = 1$ but even the using Theorem 5 gives lower bound $\Omega(\sqrt{N})$ [10].

Second, the quantum query complexity of Element Distinctness is known to be $\Theta(N^{2/3})$. The lower bound part is obtained by Aaronson and Shi by the polynomial method [5]; the upper bound part is obtained by Ambainis [12]. Observe that $C_1(f) = 2$ thus $\sqrt{NC_1(f)} = \Theta(N)$, we derive the following interesting corollary from the above theorem.

Corollary 47 *The quantum adversary method is not tight.*

In [9] Ambainis gave a function whose approximate degree is small but the quantum adversary method was used to show a large lower bound. Now the Element Distinctness shows the other direction, and thus we have

[2]The original problem is not a Boolean function, but we can define a Boolean-valued version of it. Instead of finding the position i with $x_i = 1$, we are to decide whether i is odd or even. The original proof of the $\Omega(\sqrt{N})$ lower bound still holds.

Corollary 48 *The quantum adversary method and the polynomial method is incomparable in power.*

We make some remarks on the quantity $\sqrt{N \cdot C_-(f)}$ to end this section. A function f is symmetric if $f(x_1...x_N) = f(x_{\sigma(1)}...x_{\sigma(n)})$ for any input x and any permutation σ on $[N]$. In [20], Beals *et al.* prove that $Q_2(f) = \Theta(\sqrt{N(N-\Gamma(f))})$ by using Paturi's result $\widetilde{deg}(f) = \Theta(\sqrt{N(N-\Gamma(f))})$ [57], where $\Gamma(f) = \min\{|2k-n+1| : f_k \neq k_{k+1}, 0 \leq k \leq n-1\}$. It is not hard to show that $\Gamma(f) = N - \Theta(C_-(f))$ for symmetric function f. Thus we know that both $\widetilde{deg}(f)$ and $Q_2(f)$ are $\Theta(\sqrt{N \cdot C_-(f)})$ for symmetric function f.

7.2 Limitations of the quantum adversary method for total functions

It turns out that if the function is total, then the upper bound can be further tightened. We introduce a new measure which basically characterizes the size of intersection of a 0 and 1-certificate sets.

Definition 12 *For any function f, if there is a certificate set assignment $CS : \{0,1\}^N \rightarrow 2^{[N]}$ such that for any inputs x, y with $f(x) \neq f(y)$, $|CS_x \cap CS_y| \leq k$, then k is called a candidate certificate intersection complexity of f. The minimal candidate certificate intersection complexity of f is called the certificate intersection complexity of f, denoted by $CI(f)$. In other words,*

$$CI(f) = \min_{CS} \max_{x,y:f(x)\neq f(y)} |CS_x \cap CS_y|. \tag{7.20}$$

Now we give the following theorem which improves Theorem 46 for total functions. Note that $CI(f) \leq C_-(f)$ by the definition of $CI(f)$.

Theorem 49 *The quantum adversary method Theorem 6 cannot prove a lower bound better than $\Omega(\sqrt{N \cdot CI(f)})$, for any N-ary total Boolean function f.*

Proof Again, we prove the stronger result that for any (X, Y, R, u, v, w) in Theorem 6,

$$\min_{(x,y)\in R, i\in[N]} \frac{w_x w_y}{u_{x,i} v_{y,i}} \leq N \cdot CI(f). \tag{7.21}$$

As in the proof for Theorem 46, we assume without loss of generality that $R = X \times Y$ and that for all $x \in X, y \in Y$, we have

$$w_x w_y > N \cdot CI(f)\ u_{x,i} v_{y,i}. \tag{7.22}$$

We shall show a contradiction as follows. Fix i and sum (7.22) over $\{x \in X : i \in CS_x\}$ and $\{y \in Y : i \in CS_y\}$, we get

$$\begin{aligned}
& \sum_{x\in X, y\in Y:\ i\in CS_x\cap CS_y} w_x w_y & (7.23)\\
> \ & N \cdot CI(f) \left(\sum_{x\in X:\ i\in CS_x} u_{x,i} \right) \left(\sum_{y\in Y:\ i\in CS_y} v_{y,i} \right) & (7.24)\\
= \ & N \cdot CI(f) \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x, x_i\neq y_i}} u(x,y,i) \right) \cdot \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_y, x_i\neq y_i}} v(x,y,i) \right) & (7.25)\\
\geq \ & N \cdot CI(f) \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} u(x,y,i) \right) \cdot \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} v(x,y,i) \right) & (7.26)\\
\geq \ & N \cdot CI(f) \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} \sqrt{u(x,y,i) v(x,y,i)} \right)^2 & (7.27)\\
\geq \ & N \cdot CI(f) \left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} w(x,y) \right)^2. & (7.28)
\end{aligned}$$

Now summing over $i = 1, ..., N$, we have

$$\begin{aligned}
&\sum_{\substack{x\in X, y\in Y, i\in[N]:\\ i\in CS_x\cap CS_y}} w_x w_y && (7.29)\\
&> N\cdot CI(f)\sum_{i=1}^{N}\left(\sum_{\substack{x\in X, y\in Y:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} w(x,y)\right)^2 && (7.30)\\
&\geq CI(f)\left(\sum_{\substack{x\in X, y\in Y, i\in[N]:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} w(x,y)\right)^2. && (7.31)
\end{aligned}$$

Note that for a total function f, if $f(x) \neq f(y)$, then there is at least one position $i \in CS_x \cap CS_y$ such that $x_i \neq y_i$.[3] Thus

$$\sum_{\substack{x\in X, y\in Y, i\in[N]:\\ i\in CS_x\cap CS_y, x_i\neq y_i}} w(x,y) \geq \sum_{x\in X, y\in Y} w(x,y). \tag{7.32}$$

On the other hand, by the definition of $CI(f)$, we have

$$\sum_{\substack{x\in X, y\in Y, i\in[N]:\\ i\in CS_x\cap CS_y}} w_x w_y \leq CI(f)\sum_{x\in X, y\in Y} w_x w_y = CI(f)\left(\sum_{x\in X, y\in Y} w(x,y)\right)^2. \tag{7.33}$$

Therefore we get a contradiction

$$CI(f)\left(\sum_{x\in X, y\in Y} w(x,y)\right)^2 > CI(f)\left(\sum_{x\in X, y\in Y} w(x,y)\right)^2, \tag{7.34}$$

as desired. □

And-Or Tree is a famous problem in both classical and quantum computation. In the problem, there is a complete binary tree with height $2n$. Any node in an odd level

[3]This is not true for partial functions, because it is possible that $CS_x \cap CS_y = \emptyset$, which may be the case when the domain of f does not contain any string that agrees both with x on CS_x and with y on CS_y. For example, in the Permutation Inversion problem, the input $x \in [N]^N$ is a permutation of $[N]$, and we are required to find $i \in [N]$ such that $x_i = 1$. It is clear that for each input x, the certificate set CS_x is a singleton whose only member is the position of 1. For this problem, if $f(x) \neq f(y)$, then $CS_x \cap CS_y = \emptyset$. Note that in the proof of Theorem 46, what we argue is that for any x and y satisfying $f(x) \neq f(y)$, there is at least one position $i \in CS_x$ such that $x_i \neq y_i$. This is true even for partial function because otherwise $f(y) = f(x)$ according to the definition of CS_x.

is labeled with AND and any node in an even level is labeled with OR. The $N = 4^n$ leaves are the input variables, and the value of the function is the value that we get at the root, with value of each internal node calculated from the values of its two children in the common AND/OR interpretation. The classical randomized decision tree complexity for AND-OR TREE is known to be $\Theta((\frac{1+\sqrt{33}}{4})^n) = \Theta(N^{0.753\ldots})$ by Saks and Wigderson in [61] and Santha in [62]. The best known quantum lower bound is $\Omega(\sqrt{N})$ by Barnum and Saks in [13] and best known quantum upper bound is the same as the best classical randomized one. Note that $C_-(\text{AND-OR TREE}) = 2^n = \sqrt{N}$ and thus $\sqrt{NC_-(f)} = N^{3/4}$. So if we only use Theorem 46, it seems that we still have a chance to improve the known $\Omega(\sqrt{N})$ lower bound by Theorem 6. But now by Theorem 49, we know that actually that is impossible.

Corollary 50 *The quantum adversary method Theorem 6 cannot prove a lower bound better than $\Omega(\sqrt{N})$ for* AND-OR TREE.

Proof It is sufficient to prove that there is a certificate assignment CS such that $|CS_x \cap CS_y| = 1$ for any $f(x) \neq f(y)$. In fact, by a simple induction, we can prove that the standard certificate assignment satisfies this property. The base case is trivial. For the induction step, we note that for an AND connection of two subtrees, the 0-certificate set of the new larger tree can be chosen as any one of the two 0-certificate sets of the two subtrees, and the 1-certificate set of the new larger tree can be chosen as the union of the two 1-certificate sets of the two subtrees. As a result, the intersection of the two new certificate sets is not enlarged. The OR connection of two subtrees is analyzed in the same way. Thus the intersection of the final 0- and 1-certificate sets is of size 1. $\square$

We can tighten the $\sqrt{N \cdot C_-(f)}$ upper bound in another way and get the following result which also implies Corollary 50.

Theorem 51 *The quantum adversary method Theorem 6 cannot prove a lower bound better than $\Omega(\sqrt{C_0(f)C_1(f)})$, for any total Boolean function f.*

Proof For any (X, Y, R, u, v, w) in Theorem 3, we assume without loss of generality that $X \subseteq f^{-1}(0), Y \subseteq f^{-1}(1)$ and $R = X \times Y$. We are to prove $\exists x, y, i, j$ such that $w_x w_y \leq$

$C_0(f)C_1(f)u_{x,i}v_{y,j}$. Suppose this is not true, *i.e.* for all $x \in X, y \in Y, i, j \in [N]$,

$$w_x w_y > C_0(f)C_1(f)u_{x,i}v_{y,j}. \tag{7.35}$$

First fix x, y and sum over $i \in CS_x$ and $j \in CS_y$. Since $|CS_x| \leq C_0(f), |CS_y| \leq C_1(f)$, we have

$$w_x w_y > \sum_{i \in CS_x} u_{x,i} \sum_{j \in CS_y} v_{y,j}. \tag{7.36}$$

Now we sum over $x \in X$ and $y \in Y$,

$$\left(\sum_{x \in X} w_x\right)\left(\sum_{y \in Y} w_y\right) \tag{7.37}$$

$$> \left(\sum_{x \in X, i \in CS_x} u_{x,i}\right)\left(\sum_{y \in Y, j \in CS_y} v_{y,j}\right) \tag{7.38}$$

$$= \left(\sum_{\substack{x \in X, y \in Y, i \in [N]: \\ x_i \neq y_i, i \in CS_x}} u(x,y,i)\right) \cdot \left(\sum_{\substack{x \in X, y \in Y, j \in [N]: \\ x_j \neq y_j, j \in CS_y}} v(x,y,j)\right). \tag{7.39}$$

Since f is total, there is at least one $i_0 \in CS_x \cap CS_y$ such that $x_{i_0} \neq y_{i_0}$. Thus

$$\left(\sum_{x \in X} w_x\right)\left(\sum_{y \in Y} w_y\right) > \left(\sum_{\substack{x \in X, \\ y \in Y}} u(x,y,i_0)\right)\left(\sum_{\substack{x \in X, \\ y \in Y}} v(x,y,i_0)\right) \tag{7.40}$$

$$\geq \left(\sum_{x \in X, y \in Y} \sqrt{u(x,y,i_0)v(x,y,i_0)}\right)^2 \tag{7.41}$$

$$\geq \left(\sum_{x \in X, y \in Y} w(x,y)\right)^2 \tag{7.42}$$

$$= \left(\sum_{x \in X} w_x\right)\left(\sum_{y \in Y} w_y\right) \tag{7.43}$$

which is a contradiction. □

Finally, we remark that even these two improved upper bounds are not always tight. For example, Sun, Yao and Zhang prove [71] that graph property SCORPION, directed graph property SINK and a circular function all have $Q_2(f) = \tilde{\Theta}(\sqrt{n})$, but both $\sqrt{C_0(f)C_1(f)}$ and $\sqrt{N \cdot CI(f)}$ are $\Theta(n)$.

References

[1] K. Aardal, S. Hoesel, J.K. Lenstra, L. Stougie. A Decade of Combinatorial Optimization. CWI Tracts 122, pp. 5-14, 1997.

[2] S. Aaronson. Lower Bounds for Local Search by Quantum Arguments. Proceedings of the thirty-sixth Annual ACM Symposium on Theory of Computing, pp. 465-474, 2004.

[3] S. Aaronson. Limitations of Quantum Advice and One-Way Communication, Theory of Computing 1:1-28, 2005.

[4] S. Aaronson and A. Ambainis. Quantum search of spatial regions. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 200-209, 2003.

[5] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. Journal of the ACM, 51(4), pp. 595-605, 2004.

[6] E. Aarts and J. Lenstra, Local Search in Combinatorial Optimization, John Wiley & Sons, Inc. New York, NY, USA, 1997.

[7] D. Aldous. Minimization Algorithms and Random Walk on the d-Cube. Annals of Probability, 11(2), pp. 403-413, 1983.

[8] I. Althöfer and K. Koschnich. On the Deterministic Complexity of Searching Local Maxima. Discrete Applied Mathematics 43, pp. 111-113, 1993.

[9] A. Ambainis. Polynomial Degree vs. Quantum Query Complexity. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 230-239, 2003.

[10] A. Ambainis. Quantum Lower Bounds by Quantum Arguments, Journal of Computer and System Sciences, 64, pp. 750-767, 2002.

[11] A. Ambainis. Quantum lower bounds for collision and element distinctness with small range. Theory of Computing, 1(3), 2005.

[12] A. Ambainis. Quantum Walk Algorithm for Element Distinctness. Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 22-31, 2004.

[13] H. Barnum and M. Saks. A lower bound on the quantum query complexity of read-once functions. Journal of Computer and Systems Sciences, 69(2):244258, 2004.

[14] H. Barnum, M. Saks, M. Szegedy. Quantum Query Complexity and Semidefinite Programming. Proceedings of the 18th Annual IEEE Conference on Computational Complexity, pp. 179-193, 2003.

[15] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, Strengths and weaknesses of quantum computing, SIAM Journal on Computing 26, pp. 15101523, 1997.

[16] M. R. Best, P. van Emde Boas and H. W. Lenstra, Jr. A sharpened version of the Aanderaa-Rosenberg conjecture, Report ZW 30/74, Mathematish Centrum, Amsterdam, 1974.

[17] B. Bollobás and S. E. Eldridge, Packing of graphs and applications to computational complexity, J. of Combinatorial Theory Ser. B 25, 105-124.

[18] Berzina, Dubrovsky, Freivalds, Lace and Scegulnaja, Quantum query complexity for some graph problems. Proceedings of the 30th conference on Current Trends in Theory and Practice of Computer Science, pp. 140-150, 2004.

[19] G. Brassard, P. Hoyer, A. Tapp. Quantum Counting. Proceedings of 25th International Colloquium on Automata, Languages, and Programming, LNCS 1443, 820-831, 1998.

[20] R. Beals, H. Buhrman, R. Cleve, M.Mosca, R. de Wolf. Quantum Lower Bounds by Polynomials. Journal of ACM, 48, pp. 778-797, 2001.

[21] H. Buhrman, R. Cleve, A. Wigderson. Quantum vs. classical communication and computation. Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 63-68, 1998

[22] H. Buhrman, C. Drr, M. Heiligman, P. Hyer, F. Magniez, M. Santha, R. de Wolf. Quantum algorithms for element distinctness, SIAM Journal on Computing, 34(6), pp. 1324-1330, 2005.

[23] H. Buhrman and R. de Wolf. Complexity Measures and Decision Tree Complexity: a survey. Theoretical Computer Science, Volume 288, Issue 1, pp. 21-43, 2002.

[24] A. Chakrabarti and S. Khot. Improved lower bounds on the randomized complexity of graph properties. Proceedings of the 28th International Colloquium on Automata, Lanaguages, and Programming, pp. 285-296, 2001.

[25] A. Childs and J. Eisenberg. Quantum algorithms for subset finding. Quantum Information and Computation 5, 593 (2005)

[26] R. de Wolf. Quantum communication and complexity. Theoretical Computer Science, 287(1), pp. 337-353, 2002.

[27] D. Deutsch. Quantum theory, the Church-Turing principle and the Universal Quantum Computer. Proceedings of the Royal Society of London: Series A - Mathematical and Physical Sciences A, 400(1818), pp. 97-117, 1985.

[28] D. Deutsch. Quantum Computational Networks. Proceedings of the Royal Society of London: Series A - Mathematical and Physical Sciences A, 425(1868), pp. 73-90, 1989.

[29] D. Deutsch and R. Josza. Rapid solution of problems by quantum computation, Proceedings of the Royal Society of London: Series A - Mathematical and Physical Sciences A, 439(1907), pp. 553-558, 1992.

[30] R. DeVore and G. Lorentz. Constructive approximation. Spring-Verlag, Berlin, 1993.

[31] C. Durr, M. Heiligman, P. Hoyer, M. Mhalla. Quantum Query Complexity of some Graph Problems. Proceedings of the 31st International Colloquium on Automata, Lanaguages, and Programming, pp. 481-493, 2004.

[32] C. Durr and P. Hoyer. A Quantum Algorithm for Finding the Minimum, quant-ph/9607014, 1996.

[33] R. Feynman. Simulating physics with computers. International Journal of Theoreteical Physics 21, pp. 467 - 488, 1982.

[34] L. Grover. A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, pp. 212-219, 1996.

[35] P. Hajnal, An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties, Combinatorica, 11, pp. 131-143, 1991.

[36] S. Hallgren. Polynomial-Time Quantum Algorithms for Pell's Equation and the Principal Ideal Problem. Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 653 658, 2002.

[37] P. Hoyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality. Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science, pp. 299-310, 2002.

[38] P. Hoyer, M. Mosca, R. de Wolf: Quantum Search on Bounded-Error Inputs. Proceedings of 30th International Colloquium on Automata, Languages, and Programming, LNCS 2719, 291-299, 2003.

[39] P. Hoyer and R. Spalek. Lower Bounds on Quantum Query Complexity, Bulletin of the European Association for Theoretical Computer Science 87, pp. 78-103, 2005.

[40] R. Jain. J. Radhakrishnan, P. Sen. A lower bound for bounded round quantum communication complexity of set disjointness. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 220 - 229, 2003

[41] D. Johnson, C. Papadimitriou, and M. Yannakakis. How Easy is Local Search, Journal of Computer and System Sciences 37, pp. 429448, 1988.

[42] V. King, Lower bounds on the complexity of graph properties, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pp. 468-476, 1988.

[43] A. Kitaev, A. Shen, M. Vyalyi Classical and Quantum Computation, American Mathematical Society, 2002.

[44] G. Kuperberg, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, quant-ph/0302112

[45] E. Kushilevitz and N. Nisan, Communication Complexity, Cambridge University Press, 1997.

[46] S. Laplante and F. Magniez. Lower Bounds for Randomized and Quantum Query Complexity Using Kolmogorov Arguments. Proceedings of the 19th Annual IEEE Conference on Computational Complexity, pp. 294-304, 2004.

[47] D. Llewellyn and C. Tovey. Dividing and Conquering the Square. Discrete Applied Mathematics 43, pp. 131-153, 1993.

[48] D. Llewellyn, C. Tovey, M. Trick. Local Optimization on Graphs. Discrete Aplied Mathematics 23, pp. 157-178, 1989. Erratum: 46, pp. 93-94, 1993.

[49] C. Lomont. The Hidden Subgroup Problem - Review and Open Problems. quant-ph/0411037

[50] F. Magniez, M. Santha, M. Szegedy. Quantum algorithms for the Triangle problem. Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, pp. 1109-1117, 2005.

[51] N. Megiddo and C. Papadimitriou. On Total Functions, Existence Theorems, and Computational Complexity. Theoretical Computer Science 81, pp. 317324, 1991.

[52] M. Nielsen and I. Chuang. Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[53] N. Nisan, CREW PRAMS and decision trees, SIAM Journal on computing, 20, 6, 999-1007, 1991. Earlier version in STOC'89.

[54] R. O'Donnell, M. Saks, O. Schramm, R. Servedio. Every decision tree has an influential variable. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science*, pp. 31-39, 2005.

[55] R. O'Donnell and R. Servedio. Learning monotone functions from random examples in polynomial time. *Manuscript*, 2005.

[56] J. Orlin, A. Punnen, A. Schulz. Approximate Local Search in Combinatorial Optimization. SIAM Journal on Computing, 33(5), pp. 12011214, 2004.

[57] R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions, Proceedings of the 24th Annual ACM Symposium on the Theory of Computing, 468-474, 1992.

[58] A. Razborov. Quantum communication complexity of symmetric predicates. Izvestiya: Mathematics, 67(1), pp. 145-159, 2003

[59] O. Regev. Quantum computation and lattice problems. SIAM Journal on Computing 33(3), pp. 738-760, 2004.

[60] R. L. Rivest and J. Vuillemin, On recognizing graph properties from adjacency matrices, Theoretical Computer Science 3: 371 - 384, 1976.

[61] M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. Proceedings of the 27th Annual Symposium on Foundations of Computer Science, pp. 29-38, 1986.

[62] M. Santha. On the Monte Carlo Boolean decision tree complexity of read-once formulae. Proceedings of the 6th Annual Structure in Complexity Theory Conference, pp. 180-187, 1991.

[63] M. Santha and M. Szegedy. Quantum and Classical Query Complexities of Local Search Are Polynomially Related. Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 494-501, 2004.

[64] Y. Shi. Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of boolean variables. *Information Processing Letters*, 75(1-2), pp. 79-83, 2000.

[65] Y. Shi and S. Zhang, The Quantum Query Complexity, the Approximate Degree and the Total Influence, manuscript, 2005.

[66] P. Shor. Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26, pp. 1484-1509, 1997.

[67] P. Shor. Progress in quantum algorithms. Quantum Information Processing 3, pp. 5-13, 2004.

[68] D. Simon. On the Power of Quantum Computation. Proceedings of the 35th Annual Symposium on the Foundations of Computer Science, pp. 116-123, 1994.

[69] R. Spalek and M. Szegedy. All Quantum Adversary Methods Are Equivalent. Proceedings of 32nd International Colloquium on Automata, Languages and Programming, pp. 1299-1311, LNCS 3580, Lisboa, Portugal, 2005.

[70] M. Szegedy. On the quantum query complexity of detecting triangles in graphs. quant-ph/0310107, 2003.

[71] X. Sun, A. Yao, S. Zhang, Graph properties and circular functions: how low can quantum query complexity go? Proceedings of the19th Annual IEEE Conference on Computational Complexity, pp. 286-293, 2004.

[72] G. Turan, The critical complexity of graph properties, Information Processing Letters, 18, 151-153, 1984

[73] W. van Dam, S. Hallgren and L. Ip, Quantum algorithms for some hidden shift problems. Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms, pp. 489-498, 2003.

[74] Y. Verhoeven, Enhanced Algorithms for Local Search. Information Processing Letters 97, pp. 171-176, 2006.

[75] A. Yao, Some Complexity Questions Related to Distributive Computing, Proceedings of the 11th Annual ACM Symposium on Theory of Computing, 209-213, 1979.

[76] A. Yao, Lower bounds to randomized algorithms for graph properties, Proceedings of the 28th Annual Symposium on Foundations of Computer Science, pp. 393-400, 1987.

[77] A. Yao, Quantum circuit complexity, Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, pp. 352-361, 1993

[78] S. Zhang. On the Power of Ambainis Lower Bounds. Theoretical Computer Science, 339(2-3), pp. 241-256, 2005.

[79] S. Zhang, Promised and Distributed Quantum Search. Proceedings of the 11th International Computing and Combinatorics Conference, pp. 430-439, 2005.

[80] S. Zhang, New upper and lower bounds for randomized and quantum Local Search, the 38th ACM Symposium on Theory of Computing 2006, to appear.