Quantum Query Complexity of Hypergraph Search Problems

by

Zhiying Yu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

© Zhiying Yu 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the study of quantum query complexity, it is natural to study the problems of finding triangles and spanning trees in a simple graph. Over the past decades, many techniques are developed for finding the upper and lower quantum query bounds of these graph problems. We can generalize these problems to detecting certain properties of higher rank hypergraphs and ask whether these techniques are still available. In this thesis, we will see that when the rank increase, complexity bounds still holds for some problems, although less effectively. For some other problems, their nontrivial complexity bounds vanish. Moreover, we will focused on using the generalized adversary and learning graph techniques for finding nontrivial quantum query bounds for different hypergraph search problems. The following results are presented.

- Discover a general quantum query lower bound for subhypergraph-closed properties and monotone properties over r-partite r-uniform hypergraphs.
- Provide tight quantum query bounds for the connectivity and acyclicity problems over r-uniform hypergraphs.
- Present a nontrivial learning graph algorithm for the 3-simplex finding problem.
- Formulate nested quantum walk in the adaptive learning context and use it to present a nontrivial quantum query algorithm for the 4-simplex finding problem.
- Present a natural relationship of lower bounds for simplex finding of different ranks.
- Use the learning graph formalization of tetrahedron certificate structure to find a nontrivial quantum query lower bound of the 3-simplex sum problem.

Acknowledgments

I would like to thank my supervisor Shalev Ben-David for the helpful discussions and support during my master's study, Professor Ben-David provided lots of feedback for my master's research and thesis. I also like to thank professors Ashwin Nayak, Richard Cleve, and David Gosset for introducing me to the beautiful theory of quantum computing and quantum complexity. I would also like to thank the examining committee who agreed to read and review my thesis.

Finally, I would like to thank my parents and friends for giving me the emotional support to pursue my research in quantum query complexity.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgments iv

List of Figures vii

List of Tables viii

1 Introduction 1

2 Preliminaries 4

2.1 Notations and Hypergraph Theory 4

2.2 Quantum Query Complexity 6

2.3 Quantum Counting and Quantum Walk 8

2.3.1 Quantum Counting 9

2.3.2 MNRS Quantum Walk 9

2.4 Quantum Query Lower Bound Techniques 12

2.4.1 Polynomial Bound 12

2.4.2 Positive and Generalized Adversary Bound 13

2.5 Learning Graph 16

2.5.1 Non-adaptive Learning Graph 16
2.5.2 Adaptive Learning Graph 18

3 Hypergraph Properties **21**
3.1 Subhypergraph-closed Properties 21
3.2 Monotone Hypergraph Properties 22
3.3 Hypergraph Connectivity and Acyclicity 26

4 Upper Bound on Finding n-simplices **32**
4.1 Tetrahedron Finding with Vertex Quantum Walk 33
4.2 Leanrning Graph for Nested Quantum Walk 38
4.2.1 Configuration Package 39
4.2.2 Alpha Symmetric Stage 41
4.2.3 Main Learning Graph Construction 42
4.3 4-Simplex Finding with Nested Quantum Walk 45

5 Lower Bound of Certificate Structure **51**
5.1 Rank Reduction 51
5.2 Certificate Structure 54
5.2.1 r-simplex Certificate Structure 55
5.2.2 Dual of a Learning Graph for Certificate Structures 56
5.3 Tetrahedron Sum Problem 58

6 Conclusion and Open Problems **64**

References **66**

APPENDICES **71**

A Proof of Lemmas in Section 4.3 **72**

List of Figures

2.1 Learning graph of an OR of Boolean functions and learning graph of a MNRS quantum walk. 20

3.1 Special 3-edges over a 3-partition of vertices. 26

3.2 Diagram of a 3-hypergraph instance of the Ambainis' adversary lower bound for the **3-HG Connectivity** problem. 28

5.1 Example of quantum query lower bound by rank reduction. 53

List of Tables

4.1 Parameters and update complexities of nested quantum walk learning graph for 4-simplex finding. 50

Chapter 1

Introduction

In the age of quantum computing, the study of quantum complexity theory allows us to discover the power and limitations of quantum algorithms. Unlike classical complexity theory, the time and space efficiency of quantum algorithms are hard to analyze. However, many quantum algorithms, including the famous order finding and Grover search algorithm, focus on quantum query resources.

A query to an oracle function $\mathcal{O}$ refers to one evaluation $\mathcal{O}(x)$ given some input x. However, quantum algorithms may query functions in superposition $\mathcal{O} : |x\rangle |0\rangle \mapsto |x\rangle |f(x)\rangle$. This property makes the number of quantum queries used for a function a popular complexity measure. Moreover, the simplicity of this measure allows researchers to find many interesting lower bound results, which can be used to further lower bound time complexities of quantum algorithms.

The class of problems of detecting graph properties in a simple graph was studied extensively for their quantum query complexity. Given a simple graph G with n vertices V, the indicator function of edges $\mathcal{O}_G : V^2 \to \{0, 1\}$ where $\mathcal{O}_G(v_1, v_2) = 1$ if and only if $(v_1, v_2) \in E(G)$ is treated as an oracle. For this set of problems, the challenge is to find the algorithm that decides if G has a certain property using as few quantum queries to $\mathcal{O}_G$ as possible. Usually, deciding that G has the property involves searching for a particular subgraph of G. The famous triangle finding problem is an example.

Triangle Finding Problem (Triangle):

> Given a simple graph $G = (V, E)$ with n vertices and m edges. Suppose the edge set E is given as the oracle input as a subset of $\mathcal{S}_2(V)$. Find three distinct vertices a, b, c such that $ab, ac, bc \in E$ or decide that no such vertices exist.

It is natural to extend this class of problems to hypergraphs. Assume G is an r-uniform hypergraph, whose hyperedges are size r subsets of vertices in G. The oracle $\mathcal{O}_G$ becomes the indicator of hyperedges of G. In this case, we can ask whether G contains a certain subhypergraph similar to the triangle finding problem.

Nowadays, many quantum algorithms are available for solving the triangle finding problem such as quantum walk and learning graph. The current best quantum query upper bound for this problem is $O(n^{1.25})$ [19]. On the other hand, there are many lower bound techniques for quantum query complexity such as lower bound by reduction, the polynomial method, and the different variants of adversary methods [3] [7] [5] [21] [40]. Unfortunately, these lower bound techniques don't provide much for the query complexity of the triangle finding problem, whose best lower bound today is the trivial $\Omega(n)$. Quantum query complexity is less well-studied for searching problems in hypergraphs. For example, as we will see in chapter 4, an r-simplex extends the concept of triangle to r-uniform hypergraph. Yet, to our knowledge, for $r \geq 4$, even the best quantum query algorithm for finding r-simplex is achieved by trivial Grover search. However, we can ask whether existing upper and lower bound techniques for solving graph problems transfer to solving hypergraph problems.

The primary focus of this thesis is to explore the algorithmic and lower bound approaches that nontrivially bounds the quantum query complexity of some natural hypergraph problems. We will investigate the general lower bound of subhypergraph-closed, and monotone hypergraph properties. We will also study search problems such as hypergraph connectivity, hypergraph acyclicity, r-simplex finding, and r-simplex sum, all of which naturally extend a corresponding graph problem. Moreover, despite no nontrivial lower bound is known for r-simplex finding, there is nontrivial relationship of lower bound among the class of r-simplex finding problems. Our research aims to expand what we know about the capabilities of quantum algorithms and try to encourage further quantum research for solving hypergraph problems.

This thesis is organized as follows. In chapter 2, we introduce the background knowledge and technical results in hypergraph theory and quantum complexity theory needed for the later sections.

In chapter 3, we generalize the concept of subgraph-closed property and monotone property from graphs to hypergraphs and investigate whether there is a general lower bound on the class of problems with a specific hypergraph property. Then, we present the hypergraph connectivity and acyclicity problems. We will show that these two example problems are simple enough to possess a tight quantum query complexity bound.

In chapter 4, we are interested in finding quantum query algorithms that search for an r-simplex in an r-uniform hypergraph. Here, we will see that existing quantum walk

algorithms for finding triangles cannot naturally generalize to higher rank. But, existing algorithmic techniques still allow us to build a nontrivial quantum query algorithm for 3-simplex(tetrahedron) finding. With a more complicated nested quantum walk structure, we show that the 4-simplex finding problem has a nontrivial quantum query upper bound.

In chapter 5, we search for lower bound results on simplex finding, and investigate variants of simplex finding problems in hypergraphs. Specifically, we build a randomized reduction from the r-simplex finding problem to the $(r+1)$-simplex finding problem, which results in a nontrivial relationship among simplex finding problems of different ranks. Later, we present the 3-simplex sum problem and use the learning graph of a tetrahedron certificate structure to show that, unlike the 3-simplex finding problem, this problem is understood to have a nontrivial quantum query lower bound.

Finally, in chapter 6, we state some conclusions and open problems. The appendix contains some technical proofs of lemmas in chapter 4.

Chapter 2

Preliminaries

Before we examine the class of hypergraph search problems, we need to introduce basic hypergraph notations and the necessary results from quantum complexity theory. Some of the most commonly used techniques for complexity analysis will be presented here, but most technical results in this chapter will be stated without proof.

2.1 Notations and Hypergraph Theory

Here, we introduce the basic notation and theory of hypergraphs. A hypergraph G consists of a set of vertices V and a set of hyperedges E, where every hyperedge $e \in E$ is a subset of V. We call a hyperedge e with k elements a k-edge. The integer k is the *size* of e. For the problems presented in this paper, we assume that the hypergraphs are simple, meaning that there are no parallel hyperedges and no hyperedges of size 0, 1.

We use n to denote the size of V. The *rank* r of a hypergraph G is the size of the largest hyperedge in E. For this paper, we will always assume that the rank r is upper bounded by a constant. Furthermore, if every hyperedge of G has size r, we call G an r-uniform hypergraph or an r-hypergraph in short.

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and let $[n, m]$ denote $\{n, n+1, \ldots, m\}$. Let $P_r^n = n!/(n-r)!$ denotes the permutation. Let $\mathcal{S}_r(V) = \mathcal{S}(V, r)$ denote the set of subsets of size r in V. To denote an element in $\mathcal{S}_r(V)$ conveniently, we often omit the curly bracket of a set. For example, we write a potential hyperedge $\{u, v, w\} \in \mathcal{S}_3(V)$ simply as uvw.

The weight of $w(x)$ of $x \in \{0,1\}^N$ is the number of 1s in x. Given G a hypergraph and subsets $A, B \subseteq V$, we use G_A to denote the restriction of G to A (i.e. the subgraph

of G induced by A) and we use $G_{A_1,\ldots,A_r}$ to denote the r-partite hypergraph obtained from taking the restriction of G to the r-partition $A_1, \ldots, A_r$.

Observe that a graph is a 2-uniform hypergraph, so some of the graph terminologies generalize to hypergraphs. Given a hypergraph $G = (V, E)$. We say $v, w \in V$ are *adjacent* if $v \neq w$ and there is a hyperedge $e \in E$ such that $\{v, w\} \subseteq e$, two hyperedges $e_1, e_2 \in E$ are *adjacent* if $e_1 \cap e_2 \neq \emptyset$. We say that $v \in V$ is *incident* to $e \in E$ if $v \in e$. The *degree* of a vertex $v \in V$ is the number of hyperedges incident to it. With the above definition, the concept of isomorphism and incidence matrix natually extends the corresponding graph definitions.

For $v, w \in V$, a (v, w)-*walk* in G is a sequence of alternating vertices and hyperedges $v_0 e_1 v_1 e_2 \ldots e_k v_k$ where $v_0 = v, v_k = w$ and v_{i-1} is adjacent to v_i via e_i for every $i \in [k]$. We call k is the *length* of the walk. If the vertices $v_0, v_1, \ldots, v_k$ and the hyperedges $e_1, \ldots, e_k$ in the walk are all distinct, we say the walk is a (v, w)-*path*. A (v, w)-walk is a *closed walk* if $k \geq 2$ and $v = w$, a closed walk such that $v_0, v_1, \ldots, v_{k-1}$ and $e_1, \ldots, e_k$ are all distinct is called a *cycle* [6]. Unlike a graph, a simple hypergraph with rank 3 or higher may have a cycle of length 2. For example, a 3-hypergraph with vertices $\{u, v, w, z\}$ and hyperedges $e_1 = uvw, e_2 = vwz$ has the sequence ve_1we_2v as a cycle. We say G is *connected* if for every pair of vertices $v, w \in V$, there is a (v, w)-walk in G.

If $G = (V, E)$ is an r-uniform hypergraph. The (r-dimensional) *adjacency matrix* is a function $f_G : \mathcal{S}_r(V) \to \{0, 1\}$ where $f_G(v_1 v_2 \ldots v_r) = 1$ if and only if $v_1 v_2 \ldots v_r \in E$. If f_G is constantly 0, we say G is an *empty* hypergraph, if f_G is constantly 1, we say G is a *complete* hypergraph.

For this paper, we focus on the query algorithm for r-uniform hypergraph problems. Thus, we are fixing the set of vertices V and treat f_G as a black box oracle that we can query. We usually set $V = [n]$ for convenience. In this context, we are most interested in asking whether a hypergraph has a certain property.

Definition 1. An r-uniform hypergraph *property* $\mathcal{P}$ is a class of mapping $\{\mathcal{P}_n\}_{n \in \mathbb{N}}$ such that $\mathcal{P}_n : \{0, 1\}^{\mathcal{S}_r([n])} \to \{0, 1\}$ and each $\mathcal{P}_n$ is invariant under permutation of vertices. A hypergraph G with n vertices is said to have the property $\mathcal{P}$ (denoted by $G \in \mathcal{P}$) if $\mathcal{P}_n(G) = 1$. Note that isomorphic hypergraphs are either required to all have $\mathcal{P}$ or all don't have $\mathcal{P}$.

An r-hypergraph property $\mathcal{P}$ is *nontrivial* if there is an $n \in \mathbb{N}$ such that for all $d \geq n$, there is an r-hypergraph in d vertices having $\mathcal{P}$ and there is an r-hypergraph in d vertices not having $\mathcal{P}$. The *negation* $\overline{\mathcal{P}}$ of $\mathcal{P}$ is defined by $\overline{\mathcal{P}}_n(G) = 1 - \mathcal{P}_n(G)$. A hypergraph property has the same query complexity as its negation.

In a query algorithm, we rely on the ability to ask the hyperedge oracle O_G whether an element in $S_r(V)$ is a hyperedge of G to determine whether G has a certain property. For $e \in S_r(V)$, we have $\mathcal{O}_G(e) = 1$ if and only if $e \in E$. The query model is formalized in the next section.

2.2 Quantum Query Complexity

Consider the task of computing some function $f : [q]^N \to [q]^M$ that takes arbitrary-sized strings from alphabet $[q]$ as input. Usually, q is taken to be 2, and the corresponding function is Boolean. When $q \geq 2$, we may represent each $z \in [q]$ as $\log(q)$ many Boolean bits and we can decompose f as the component functions $(f_1, \ldots, f_M)$. Therefore, we can focus on the case that $f : \{0,1\}^N \to \{0,1\}$ is a Boolean decision problem.

We may allow f to be *partial* in the sense that f can be only defined on a subset $\mathcal{D}_N \subseteq [q]^N$. Let $\mathcal{D} = \bigcup_{N \in \mathbb{N}} \mathcal{D}_N$ be the domain. If f is partial, computing f is easier because we have the additional promise that $x \in \mathcal{D}$ for any input x. If f is defined for all $x \in [q]^N$ and $N \in \mathbb{N}$, we say that f is *total*.

In the classical query model of computation, the input $x \in [q]^N$ is given as a black box oracle $\mathcal{O}_x$. The computation starts at a state independent of x. At any time step during the computation, we may query $\mathcal{O}_x$ at an index $i \in [n]$ and the oracle will return the value $\mathcal{O}_x(i) = x_i$. The entries of x cannot be inferred in any way other than to query the oracle. Naturally, our goal is to find a query algorithm that reduces the number of queries to input needed to compute f to some degree of satisfaction. Here are some important classical measures that classify this degree.

Definition 2. Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function.

$D(f)$, the *deterministic query complexity of* f is defined as the minimum number (over all classical deterministic algorithms) of the maximum number (over all $x \in \mathcal{D}$) of queries to $\mathcal{O}_x$ needed to compute $f(x)$ exactly.

$R_\epsilon(f)$, the *randomized query complexity of* f *with error* ϵ is defined as the minimum number (over all classical randomized algorithms) of the maximum number (over all $x \in \mathcal{D}$) of queries to O_x needed to compute the Boolean function f' such that $\Pr(f'(x) \neq f(x)) < \epsilon$ for all $x \in \mathcal{D}$.

$R(f)$, the *randomized query complexity of* f is defined to be $R_{\frac{1}{3}}(f)$. When $0 < \epsilon < \frac{1}{2}$ is a constant, we have $R_\epsilon(f) = \Theta(R(f))$. This can be easily justified by amplifying success probability. Thus ϵ is often omitted from the complexity measure.

Example 1. OR_N is the problem of deciding whether an input string $x \in \{0,1\}^N$ has a 1 in any of its entries. The problem is equivalent to computing the Boolean function $f : \{0,1\}^N \to \{0,1\}$ such that $f(x) = 0$ if and only if $x = 0^N$, the string of all 0s.

$D(OR_N) = N$ because if we only use $N-1$ queries to the input x and the query results are all 0, the position that we did not query can be either 0 or 1, which leads to different outcomes of $f(x)$. So, no deterministic algorithm can compute f exactly in the worst case unless we query all N positions.

We can also see that $R(OR_N) = \Theta(N)$. To show this, assume $R(OR_N) = o(N)$, let A be such a query algorithm that uses only $o(N)$ queries to compute OR_N with bounded error. Upon seeing only 0 in all of its queries, A must output 0 or else it cannot compute $OR_N(0^N)$ correctly. However, the average probability that $i \in [N]$ will be queried by A is $o(N)/N = o(1)$. There must be an index $i \in [N]$ such that given the input x that has a unique 1 at position i as the input, the x_i will be queried with $o(1)$ probability. So, $A(x)$ will only output 1 with probability $o(1)$, making A incorrect with input x. $\triangle$

Note that the only resource counted in these complexity measures is the number of queries to the input. This resource is only interested in a limited number of classical problems. On the other hand, quantum query complexity receives much more attention from quantum researchers because many important quantum advantages result from optimizing access to black-box information. Quantum access to the oracle allows us to query the input in a superposition of indices. This is more powerful compared to classical access, where only local information on x can be retrived with each query. Furthermore, it is much easier to lower bound query complexity than to lower bound the time complexity of quantum algorithms. We will discuss this in more details.

Quantum query algorithm works in the Hilbert space spanned by basis vectors of the form $|i\rangle |b\rangle |w\rangle$. $|i\rangle$ denote the query register, $|b\rangle$ denote the (1-qubit) output register, and $|w\rangle$ denote the working or ancilla registers. A unitary quantum query O_x acts on the basis states by $O_x |i\rangle |b\rangle |w\rangle = |i\rangle |b \oplus x_i\rangle |w\rangle$ for all $i \in [n], b \in [q]$. Let $|\bar{0}\rangle = |0\rangle |0\rangle |0\rangle$ be some initial state in the said Hilbert space that can be efficiently prepared and independent of oracle input x. A T-query quantum query algorithm on input x is given by a sequence of operators

$$|\psi_T^x\rangle = U_T O_x U_{T-1} \dots U_2 O_x U_1 O_x U_0 |\bar{0}\rangle \tag{2.1}$$

where $U_0, U_1, \dots, U_T$ are unitary operators. The output of the algorithm is given by some fixed POVM measurement on the final state $|\psi_T^x\rangle$. Just like the classical case, a quantum query complexity only measures the number of quantum access to the oracle, the costs of implementing the unitaries $U_0, U_1, \dots, U_T$ are 0.

Definition 3. Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function.

$Q_\epsilon(f)$, the *quantum query complexity of f with error ϵ* for some $0 < \epsilon < \frac{1}{2}$ is defined to be the minimum number T such that there is a T-query quantum algorithm A that computes a Boolean function f' where $\Pr(f'(x) \neq f(x)) < \epsilon$ for all $x \in \mathcal{D}$.

$Q(f)$, the *quantum query complexity of f* is defined to be $Q_{1/3}(f)$. Similar to randomized query complexity, we have $Q_\epsilon(f) = \Theta(Q(f))$.

$Q_E(f)$, the *exact quantum query complexity of f* is defined to be the minimum number T such that there is a T-query quantum algorithm A that computes the Boolean function f on its domain exactly. (We are using the notation $Q_E(f)$ instead of $Q_0(f)$ to avoid the confusion with zero-error quantum query complexity, which is another complexity measure that we will not study in this paper.)

Quantum query algorithms may take exponentially less query to compute some partial functions than classical algorithms. Deutsch–Jozsa Problem and Hidden Subgroup Problems are some of the most famous examples. However, hypergraph properties are total functions and the best separation between classical and quantum query complexity of total functions is at most polynomial:

Theorem 1. *[1] For all total Boolean function $f : \{0,1\}^N \to \{0,1\}$, $D(f) = O(Q(f)^4)$.*

In the rest of this paper, we assume the Boolean functions are always total. In the next 3 sections, we present techniques to close this bound even further for hypergraph properties, ideally finding a tight bound for the degree of the polynomial.

2.3 Quantum Counting and Quantum Walk

Many interesting hypergraph properties revolve around having a certain substructure inside the hypergraph. Thus deciding the property comes down to searching for a proof that such substructure exists. This is captured by the following definitions:

Definition 4. Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function. Given a subset of indices $S \subseteq [N]$, where $S = \{s_1, \ldots, s_k\}$ is an increasing enumeration of S. A partial assignment on S is a Boolean string $z = (z_{s_1}, z_{s_2}, \ldots, z_{s_k}) \in \{0,1\}^S$. We say an input $x \in \{0,1\}^N$ *extends* the partial assignment z (denoted by $z \subseteq x$) if $x_{s_i} = z_{s_i}$ for all $i \in [k]$. Given $x \in \{0,1\}^N$, the projection of x to S is defined by $x_S := (x_{s_1}, \ldots, x_{s_k})$.

We say that $x \in \{0,1\}^N$ is a *1-input* (*0-input*) of f if $f(x) = 1 (f(x) = 0)$. A 1-certificate (0-certificate) of f is a partial assignment z such that every input x extending z is a 1-input (0-input) of f.

Note that partial assignment encodes the partial knowledge we have about a hypergraph. However, we can output the value of f once we find a certificate of f. In this section, we will introduce some important quantum algorithmic methods for finding this certificate. We will utilize these methods as subroutines for more complicated hypergraph algorithms.

2.3.1 Quantum Counting

The most well-known quantum algorithm is Grover search, which can solve the OR_N problem in example 1 with $\Theta(\sqrt{N})$ quantum queries. Here, we will state a more involved version of Grover search that utilizes the proportion of 1 in the input.

Theorem 2 (Tight Grover Search)**.** *[13] Let $x \in \{0,1\}^N$ be an input string of the OR_N problem. Let $t = w(x)$ which is the (unknown) number of 1 entries in x. Then, if $x \neq 0^N$, in $O(\sqrt{N/t})$ expected number of queries, we can find an index $i \in [n]$ such that $x_i = 1$.*

Grover search was later generalized by Brassard et. al. in [14] to the amplitude amplification procedure that searches for a good outcome of an underlying quantum procedure. Concretely, if there is a quantum procedure $\mathcal{A}$ such that $\mathcal{A}|0^m\rangle = \sqrt{p}\,|\psi_{good}\rangle + \sqrt{1-p}\,|\psi_{bad}\rangle$. We can decide whether the phase $\sqrt{p}$ of the good state $|\psi_{good}\rangle$ is nonzero with an amplification argument similar to Grover search. Moreover, we can estimate p with repeated use of the algorithm, this method allows us to count the number of 1-entries in x approximately and estimate the value t.

Theorem 3 (Quantum Counting)**.** *[14] Let $f : [N] \to \{0,1\}$ be a Boolean function and $\epsilon > 0$. Let $t = |f^{-1}(1)|$. There is a quantum query algorithm that produce an estimate t' of t such that $|t - t'| \leq \epsilon t$ with constant probability, with an expected number of $\Theta(\frac{1}{\epsilon}\sqrt{N/t})$ number of queries to f.*

2.3.2 MNRS Quantum Walk

Consider a search problem given by a set of marked elements $M \subseteq V$ inside a finite set V. An algorithm generally used for solving such a problem is the quantization of the classical random walks, which we call quantum walks. While many different types of quantum walk have been introduced ([4], [36], [24]), we will focus on the discrete time quantum walk on graphs constructed from a classical Markov chain ([37], [30]).

Given a graph G on vertex set V, a discrete time quantum walk on G is represented by an $n \times n$ stochastic matrix $P = (P_{ij})_{i,j \in V}$. P is a Markov chain with P_{ij} the probability that vertex j will transition to vertex i in one time step. Therefore, the j^{th} column of G is a probability vector on the adjacent vertices of the vertex i. We say P is *ergodic* if the greatest common divisor of the time steps it takes between any two states of the chain is 1, and P is *reversible* if there is a vector $(s_1, \ldots, s_n) \geq 0^n$ such that $\sum_{i \in V} s_i = 1$ and $s_i P_{ij} = s_j P_{ji}$ for every $i, j \in V$. If P is ergodic, there is a unique unit eigenvector π of P with eigenvalue 1. Let $\delta(P)$ denote the eigenvalue gap of P, where λ_i ranges from the eigenvalues of P different from 1.

The objective of the quantum walk search algorithm is to find a marked vertex $v \in M \subseteq V$. For the exact algorithm, we refer the readers to [30]. Roughly speaking, the quantum walk starts with an initial state, repeatedly updates the current state, and checks if the current state is marked while the states are in superposition. During the walk, we associate a data structure $D(x)$ for every $x \in V$ and 3 types of query cost.

- Setup Cost $\boldsymbol{S}$: The cost of setting up the initial state of the walk:
$$|S\rangle = \sum_{x \in V} \sqrt{\pi_x} \, |x, D(x)\rangle \, |0\rangle \, .$$
- Update Cost $\boldsymbol{U}$: The cost of making one step of transition:
$$|x, D(x)\rangle \, |0\rangle \mapsto |x, D(x)\rangle \sum_{y \in V} \sqrt{P_{xy}} \, |y, D(y)\rangle \, .$$
- Checking Cost $\boldsymbol{C}$: The cost of a quantum procedure checking if $x \in M$ using the data structure $D(x)$, if x is marked, apply a -1 phase to the state $|x, D(x)\rangle \, |y, D(y)\rangle$

Then, we have the following complexity result.

Theorem 4 (MNRS)**.** *[30] Let P be an ergodic, reversible Markov Chain. Let $\epsilon > 0$ be a lower bound on the probability that an element chosen from the stationary distribution of P is marked when M is nonempty. Let $\delta > 0$ be the eigenvalue gap of P. Then there is a quantum algorithm that finds a marked vertex, if there is one, with constant probability and $O\left(\boldsymbol{S} + \frac{1}{\sqrt{\epsilon}}(\frac{1}{\sqrt{\delta}}\boldsymbol{U} + \boldsymbol{C})\right)$ queries.*

This approach to quantum walk generalizes the version used by Ambainis for his optimal quantum algorithm for the element distinctness problem [5]. Ambainis presents his algorithm with walks on Johnson Graph, which is all we need to solve our subhypergraph finding problems.

Definition 5. For some $0 < k = o(n)$, the *Johnson Graph* $J(n,k)$ is the graph with vertex set $\mathcal{S}(n,k)$. Two vertices $A, B \in \mathcal{S}(n,k)$ is joined by an edge if and only if $|A - B| = 1$ and $|B - A| = 1$, i.e. we can obtain B from A by removing an element of A and adding a new element in $[n]$.

The symmetric walk on $J(n,k)$ is given by a chain P where $P_{A,B} = \dfrac{1}{k(n-k)}$ for all A, B adjacent in $J(n,k)$. Suppose that for some $l < k$, $A \in \mathcal{S}(n,k)$ is marked if and only if A contains a fixed subset of vertices $\{v_1, \ldots, v_l\} \subseteq [n]$. We note that P is ergodic, reversible with stationary distribution π equal to a vector of all $1/n$. Then ϵ is a lower bound on the number of marked states $|M| \Big/ \binom{n}{k} = \binom{n-l}{k-l} \Big/ \binom{n}{k} = \Omega\left(\dfrac{k^l}{n^l}\right)$. The eigenvalue gap of P is $\Theta\left(\dfrac{1}{k}\right)$.

Corollary 5. *Let $l < k = o(n)$ and let P be the symmetric quantum walk on $J(n,k)$. Assume either $M = \emptyset$ or M is the set of $A \in \mathcal{S}_k([n])$ that contains a fixed subset $z \in \mathcal{S}_l([n])$. Then, the quantum walk algorithm finds an $A \in M$ or decides that $M = \emptyset$ with constant success probability and $O\left(\boldsymbol{S} + \left(\dfrac{n}{k}\right)^{l/2}\left(\sqrt{k} \cdot \boldsymbol{U} + \boldsymbol{C}\right)\right)$ many queries.*

Here is an example of a quantum walk algorithm for the triangle finding problem. Note that in this example and for the rest of this paper, we will round down some variables to integer values implicitly. This wouldn't affect the asymptotic bounds of query complexities.

Example 2. [31] Fix a real number $0 \leq a < 1$. Let $J(n, n^a)$ be the Johnson graph where a vertex $A \in \mathcal{S}(n, n^a)$ is marked if and only if A contains an edge of a triangle in G. Let $D(A) := G_A$ be the adjacency matrix of the subgraph induced by A.

The spectral gap of the Johnson graph is $\delta = \Omega(1/n^a)$ and the proportion of marked states is at least $\epsilon = \binom{n-2}{n^a-2}/\binom{n}{n^a} = \Omega(n^{2(a-1)})$. In the setup stage, we have to load up all potential edges in $D(A) = G_A$ for some $A \in \mathcal{S}(n, n^a)$, the setup cost is $\boldsymbol{S} = O\left(\binom{n^a}{2}\right) = O(n^{2a})$. In an update stage, we modify A by removing a vertex u from A and adding a new one v, we have to update the data register by unloading uw from $D(u)$ for all $w \in A - u$ and loading vw to $D(v)$ for all $w \in A - u$. The update cost is $\boldsymbol{U} = O(n^a)$.

In the checking stage, we will look for a vertex u that forms a triangle with an edge vw in A. Fix a $u \in V$, we can use another quantum walk to check if there is an edge vw in G_A that forms a triangle with u with $O(n^{2a/3})$ queries. Iterate over $u \in V$ with amplitude

amplification, the checking cost is $\boldsymbol{C} = O(\sqrt{n} \cdot n^{2a/3})$. By corollary 5, the query complexity of the quantum walk procedure is

$$n^{2a} + n^{1-a}(\sqrt{n^a} \cdot n^a + \sqrt{n} \cdot n^{2a/3}) = n^{2a} + n^{1+a/2} + n^{3/2-a/3} \tag{2.2}$$

This is optimal by taking $a = 3/5$, in which case, the query complexity of triangle finding becomes $\tilde{O}(n^{1.3})$. △

2.4 Quantum Query Lower Bound Techniques

In this section, we will look at two techniques for lower bounding quantum query complexity of total Boolean functions. The first method is the polynomial bound, which relates the degree of a Boolean function to its query complexity. The more important method is adversary bound, we will use this as the primary technique for lower bounding the query complexity of hypergraph properties and it will be presented with more details.

2.4.1 Polynomial Bound

The polynomial method originates from the fact that every Boolean function $f : \{0,1\}^N \to \mathbb{R}$. can be represented uniquely by a multivariate, multilinear polynomial $p(x_1, \ldots, x_N)$. We define $\deg(f)$ to be the degree of this polynomial p. This degree can be used to bound deterministic query complexity.

Theorem 6. *For any Boolean function $f : \{0,1\}^N \to \{0,1\}$, we have $D(f) \geq \deg(f)$.*

To lower bound the quantum query complexity of Boolean functions, we need a related notion of polynomial. We say p is a polynomial that *approximates f to error ϵ* if $|p(x) - f(x)| < \epsilon$ for all $x \in \{0,1\}^N$. Define the approximate degree of f to error ϵ (denoted by $\widetilde{\deg}_\epsilon(f)$) as the minimum degree of a polynomial p approximating f to error ϵ. This gives a well-known lower bound for quantum query complexity.

Theorem 7. *For any Boolean function $f : \{0,1\}^N \to \{0,1\}$, we have $Q_\epsilon(f) \geq \widetilde{\deg}_\epsilon(f)$.*

More results on the polynomial method can be found in [7].

2.4.2 Positive and Generalized Adversary Bound

The adversary technique is one of the most sophisticated methods for lower bounding quantum query complexity. It has many lower bounding applications, including but not limited to the problem of parity, permutation inversion, element distinctness, and deciding graph properties. We will talk about the positive and the generalized version of adversary method, both techniques will be the centerpiece of our lower bound analysis for hypergraph properties.

Given a boolean function f with domain $\mathcal{D} \subseteq \{0,1\}^N$. An *adversary matrix* for f is a $\mathcal{D} \times \mathcal{D}$ real matrix Γ such that $\Gamma[x,y] = \Gamma[y,x]$ for all $x, y \in \mathcal{D}$, and whenever $f(x) = f(y)$, we have $\Gamma[x,y] = 0$. For $i \in [N]$, let D_i be the $\mathcal{D} \times \mathcal{D}, \{0,1\}$-matrix such that $D_i[x,y] = 1$ if and only if $x_i \neq y_i$.

Definition 6. [40] The (generalized) adversary bound for f is

$$\mathrm{Adv}(f) = \max_{\Gamma\text{: adversary matrix for } f} \frac{||\Gamma||}{\max_{j\in[N]} ||\Gamma \circ D_j||}, \tag{2.3}$$

where $\circ$ denotes the Hadamard (entrywise multiplication) product. If in addition, the adversary matrix in equation (2.3) is required to have nonnegative entries, then we get the positive adversary bound $\mathrm{Adv}^+(f)$ for f.

Intuitively, $\Gamma[x,y]$ measures the hardness to distinguish input x from y. We can turn this intuition into a lower bound with a hybrid argument.

Theorem 8. *[21] For any (possibly partial) Boolean function f,*

$$Q_\epsilon(f) \geq \frac{1-2\sqrt{\epsilon(1-\epsilon)}}{2}\,\mathrm{Adv}(f) \geq \frac{1-2\sqrt{\epsilon(1-\epsilon)}}{2}\,\mathrm{Adv}^+(f).$$

In particular, $Q(f) = \Omega(\mathrm{Adv}(f)) = \Omega(\mathrm{Adv}^+(f))$.

The second bound in Theorem 8 is immediate from the definition of the two adversary measures. The proof of the first bound is presented in [21].

Note that equation (2.3) in the adversary bound definition can be formulated as the following optimization problem:

$$\begin{aligned} \max \quad & ||\Gamma|| && \text{(2.4a)} \\ \text{subject to} \quad & ||\Gamma \circ D_i|| \leq 1 \text{ for all } i \in [N] && \text{(2.4b)} \\ & \Gamma \text{ is an adversary matrix for } f. && \text{(2.4c)} \end{aligned}$$

It turns out that equation (2.4) can be transformed into a semidefinite program (SDP). By the theory of semidefinite duality, equation (2.4) has a strong dual program equivalent to the following.

$$\begin{aligned} \min_{u_{x,i}} \quad & \max_{x\in\mathcal{D}} \sum_{i\in[N]} ||u_{x,i}||^2 \\ \text{subject to} \quad & \sum_{\substack{i\in[N] \\ x_i\neq y_i}} \langle u_{x,i}|u_{y,i}\rangle = 1 \text{ for all } x,y\in\mathcal{D} : f(x)\neq f(y) \end{aligned} \tag{2.5}$$

where the notation $X_i \succeq 0$ means X_i is a positive semidefinite matrix.

If we follow a historical pathway, the positive adversary method is developed before the generalized version because when the theory was just developed, it was unclear what negative entries of Γ means. Because of the non-negativity restriction, positive adversary is easier to analysis and more often used to lower bound certain query complexity problems. Many variants of positive adversary exist, including the well-known weight version and the bipartite graph version [40]. However, one of the most useful and versatile versions is the Ambainis' positive adversary method [3]. It is a consequence of the original primal problem in equation (2.4).

Theorem 9 (Ambainis' Adversary Bound)**.** *Given a Boolean decision problem $f : \mathcal{D} \subseteq \{0,1\}^N \to \{0,1\}$. Suppose $X \subseteq f^{-1}(1)$ is a subset of 1-inputs and $Y \subseteq f^{-1}(0)$ is a subset of 0-inputs. Let $R \subseteq X \times Y$ be a relation, and let $m, m', l_{x,i}, l'_{x,i}$ be values such that*

1. *every $x \in X$ is R-related to at least m many different $y \in Y$,*
2. *every $y \in Y$ is R-related to at least m' many different $x \in X$,*
3. *for every $x \in X$ and $i \in [N]$, there are at most $l_{x,i}$ many different $y \in Y$ that is R-related to x and $x_i \neq y_i$,*
4. *for every $y \in Y$ and $i \in [N]$, there are at most $l'_{x,i}$ many different $x \in X$ that is R-related to y and $x_i \neq y_i$.*

Let $l_{max} = \max_{\substack{(x,y)\in R, i\in[N] \\ x_i\neq y_i}} l_{x,i}l'_{y,i}$. Then, we have $Q(f) = \Omega\left(\sqrt{\frac{mm'}{l_{max}}}\right)$.

Example 3. One of the most famous algorithms from quantum computing is Grover search, which computes the problem OR_n from example 1 with bounded error in $O(\sqrt{n})$

quantum queries. The matching lower bound $Q(OR_n) = \Omega(\sqrt{n})$ can be given by a hybrid argument. But we will show this with an adversary lower bound instead.

For $i \in [n]$, let $x_i \in \{0,1\}^n$ be the Boolean string with a unique 1 entry in position i. Let $X = \{x_1, x_2, \ldots, x_n\}$ is a subset of 1-input, let $Y = \{0^n\}$ be the set of 0-input. Define $R := X \times Y$, then every $x \in X$ is connected to $m = 1$ element in Y, and $0^n \in Y$ is connected to $m' = n$ elements in X. Fix a particular $x_j \in X$ and $i \in [n]$, we have $l_{x_j,i} = 1$ when $i = j$, otherwise $l_{x_j,i} = 0$. Moreover, $l'_{0^n,i} = 1$ because x_i is the only string in X that has its i^{th} coordinate being 1. So by Theorem 9, $Q(OR_n) = \Omega\left(\sqrt{\frac{1 \cdot n}{1 \cdot 1}}\right) = \Omega(\sqrt{n})$. △

The positive adversary matrix only yields a (relatively weak) lower bound on the quantum query complexity of f. However, it was later discovered by Reichardt [32], [33], that allowing negative entries not only gives an improved lower bound, it also makes adversary bound asymptotically tight for quantum query complexity using span program algorithm. This gives the following theorem.

Theorem 10. *For any (possibly partial) Boolean function f, $Q(f) = O(\mathrm{Adv}(f))$. Together with Theorem 8, we have $Q(f) = \Theta(\mathrm{Adv}(f))$.*

Before we conclude this section, we present a result on the quantum complexity of the composition of Boolean functions. If $f : \{0,1\}^N \to \{0,1\}$ and $g : \{0,1\}^M \to \{0,1\}$ are Boolean decision problems. We define the composition $f \circ g^M = f \circ (g, g, \ldots, g) : \{0,1\}^{NM} \to \{0,1\}$ as the function

$$f \circ g^M(x^1 x^2 \ldots x^N) := f(g_1(x^1) g_2(x^2) \ldots g_N(x^N))$$

for $x^1, x^2, \ldots, x^N \in \{0,1\}^M$. To relate the quantum query complexity of the composition with the quantum query complexity of f and g. We will use a consequence of positive and generalized adversary bound.

Theorem 11. *[21] [25] If $f : \{0,1\}^N \to \{0,1\}$ and $g : \{0,1\}^M \to \{0,1\}$ are Boolean functions, we have*

$$\mathrm{Adv}^+(f \circ g^N) = \mathrm{Adv}^+(f) \cdot \mathrm{Adv}^+(g), \text{ and } \mathrm{Adv}(f \circ g^N) = \mathrm{Adv}(f) \cdot \mathrm{Adv}(g).$$

By Theorem 10, we know that generalized adversary is equivalent to quantum query complexity. The following corollary is immediate.

Corollary 12. *If $f : \{0,1\}^N \to \{0,1\}$ and $g : \{0,1\}^M \to \{0,1\}$ are Boolean functions, we have*

$$Q(f \circ g^N) = \Theta(Q(f) \cdot Q(g)).$$

2.5 Learning Graph

2.5.1 Non-adaptive Learning Graph

As powerful as the general adversary method is, the number of equality constraints in equation (2.5) is potentially exponential in n. It is often difficult to find a feasible solution that makes sense for the problem, let alone optimize the objective value. Instead, we apply the learning graph computational framework, which generates feasible solutions to equation (2.5). By Theorem 10, such a feasible solution provides an upper bound for the quantum query complexity of f. We are left with optimizing the objective values.

Definition 7. [9] Let f be a Boolean function with domain $\mathcal{D} \subseteq \{0,1\}^N$. A *(reduced) non-adaptive learning graph for* f is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that

1. every vertex $v \in \mathcal{V}$ is labeled by a subset $s(v) \subseteq [N]$ of indices of inputs to f,
2. $\mathcal{G}$ has a root vertex labeled by the empty set $\emptyset$,
3. every directed edge $e = \overrightarrow{uv} \in \mathcal{E}$ satisfies $s(u) \subseteq s(v)$,
4. every directed edge $e = \overrightarrow{uv} \in \mathcal{E}$ has a length given by $l(e) = |s(v) - s(u)|$,
5. every directed edge $e = \overrightarrow{uv} \in \mathcal{E}$ has a positive weight $w(e) \in \mathbb{R}^+$,
6. every 1-input y of f (that is, $y \in f^{-1}(1)$) has a flow p_y of value 1 on the learning graph $\mathcal{G}$ where the root vertex of $\mathcal{G}$ is the source and every vertex $v \in \mathcal{V}$ such that $s(v)$ contains a 1-certificate of y in f is a sink.

In order to distinguish the vertices and edges of a learning graph from the vertices and edges of a graph in the question, we call the vertices in the learning graphs *L-vertices* and call the directed edges in the learning graphs *L-edges* (or transitions).

A learning graph framework intuitively identifies a quantum query algorithm because we can view each L-vertex v as the state of the computation where we learned oracle entries $\{f(x_i) : i \in s(v)\}$. We call $s(v)$ the set of *loaded elements* of the L-vertex v and we say an L-edge $e = \overrightarrow{uv}$ *loads* elements $u_1, \ldots, u_k$ if $s(v) - s(u) = \{u_1, \ldots, u_k\}$. A flow from the root of the learning graph to L-vertices labeled by a 1-certificate of y corresponds to the process of learning the certificates by query in parallel.

Remark 1. The learning graph $\mathcal{G}$ is called "non-adaptive" because the weights of the L-edges are independent of the input to the function. This means that, unlike a decision tree, in the algorithm corresponding to this learning graph, the position to query next (in superposition) is also independent of the outcome of the previous queries. Such algorithms are more restrictive, but it is still useful for quantum searching while its lower bound is easier to analyze.

$\mathcal{G}$ is also called "reduced" because in Belov's original definition [9], all L-edges of G load exactly 1 element. However, with suitable modification of weights, we can transform a directed path of length k in $\mathcal{G}$ to a single L-edge loading k elements, even when $k = 0$. Note that $\mathcal{G}$ is also allowed to have more than one L-vertex labeled by the same underlying set, thus the label to L-vertices may include additional data when it's convenient.

△

Definition 8. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a non-adaptive learning graph for f. For $\mathcal{F} \subseteq \mathcal{E}$, the *negative complexity* and *positive complexity* of $\mathcal{F}$ is given by

$$C_0(\mathcal{F}) := \sum_{e \in \mathcal{F}} l(e)w(e), \quad C_1(\mathcal{F}, y) := \sum_{e \in \mathcal{F}} l(e)\frac{p_y(e)^2}{w(e)}, \quad C_1(\mathcal{F}) := \max_{y \in f^{-1}(1)} C_1(\mathcal{F}, y). \quad (2.6)$$

The *learning graph complexity* of $\mathcal{G}$ is $\mathcal{LG}(\mathcal{G}) = \sqrt{C_0(\mathcal{E})C_1(\mathcal{E})}$. The *learning graph complexity* $\mathcal{LG}(f)$ of the function f is the minimum complexity of a learning graph for f.

A learning graph $\mathcal{G}$ can be turned into a feasible solution to equation (2.5) with objective value $\mathcal{LG}(\mathcal{G})$ [10]. By Theorem 10, every learning graph $\mathcal{G}$ for f corresponds to a quantum query algorithm for f.

Theorem 13. *For any (possibly partial) Boolean function f, $Q(f) = O(\mathcal{LG}(f))$.*

Here are some conventions on how we will design learning graph $\mathcal{G}$ for function f. Define the i^{th} level of $\mathcal{G}$ by the set of L-vertices at depth i from the root vertex of $\mathcal{G}$. A stage of $\mathcal{G}$ will be the set of L-edges between level i, j for some $i < j$. Usually, the stages we are going to define only has depth 1, that is, $j = i + 1$. We design learning graph by giving L-edges in stages.

Following the convention of [16], we assume the 1-complexity of a stage $\mathcal{F} \subseteq \mathcal{E}$ is always upper bounded by 1, this can be achieved by multiplying the weights of every $e \in \mathcal{F}$ by $C_1(\mathcal{F})$. Let $S = \{u_1, \ldots, u_k\} \subseteq V$, we use Load_S to denote the L-edge that loads $u_1, \ldots, u_k$. Set $w(\text{Load}_S) = k$, then $C_0(\text{Load}_S) = k^2$ and $C_1(\text{Load}_S) \leq 1$.

Definition 9. Suppose $\mathcal{F}$ is a stage with starting L-vertices V_i and ending L-vertices V_j. Let $c := |V_i|, e := |V_j|$. We say $\mathcal{F}$ is *symmetric* if

- every $v \in V_i$ has outdegree d in $\mathcal{F}$,
- the number c' of $v \in V_i$ that receives positive flow from p_y is independent of $y \in f^{-1}(1)$, and the value of these positive flows all equal to $1/c'$,
- for every $v \in V_i$ that receives positive flow from p_y, d' of the d out-edges of v have positive flow of equal values, the value d' is independent of $y \in f^{-1}(1)$,
- the number e' of $w \in V_j$ that receives positive flow from p_y is independent of $y \in f^{-1}(1)$, and the value of these positive flows all equal to $1/e'$.

Let $T = \dfrac{cd}{c'd'}$ be the *speciality* of $\mathcal{F}$, we get the following complexity for $\mathcal{F}$.

Lemma 14. *[28], [16] Let $\mathcal{F}$ be a symmetric stage of $\mathcal{G}$ with speciality T. For every $y \in f^{-1}(1)$, if L is the average length of the L-edges receiving positive flow then the L-edges in $\mathcal{F}$ can be weighted so that*

$$C_0(\mathcal{F}) \leq T \cdot L^2 \quad \text{and} \quad C_1(\mathcal{F}, y) \leq 1.$$

2.5.2 Adaptive Learning Graph

In an adaptive learning graph, the weight of an L-edge may depend on queried entries of the input z to f.

Definition 10. [16] Let f be a (possibly partial) Boolean function with domain $\mathcal{D} \subseteq \{0,1\}^N$. A directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an *adaptive learning graph* for f if it satisfies all properties (1) to (6) in definition 7, except we replace property (5) with

5'. For every $z \in \mathcal{D}$ and directed edge $e = \overrightarrow{uv} \in \mathcal{E}$, there is a positive weight value $w_{z_{s(v)}}(e) \in \mathbb{R}^+$, whose value depends only on e and the partially loaded $s(v)$-entries of the input z.

Since v is clear given the directed edge e, we abbreviate $w_{z_{s(v)}}(e)$ by $w_z(e)$. The corresponding complexity of an adaptive learning graph is given as follows.

Definition 11. Let $\mathcal{G}$ be an adaptive learning graph for f. If $\mathcal{F} \subseteq \mathcal{E}$ is a stage of $\mathcal{G}$, for $x, y \in \mathcal{D}$, we define the negative and positive complexity of $\mathcal{F}$ respectively as

$$C_0(\mathcal{F}, x) := \sum_{e \in \mathcal{F}} l(e) w_x(e), \quad C_0(\mathcal{F}) := \max_{x \in f^{-1}(0)} C_0(\mathcal{F}, x)$$

$$C_1(\mathcal{F}, y) := \sum_{e \in \mathcal{F}} l(e) \frac{p_y(e)^2}{w_y(e)}, \quad C_1(\mathcal{F}) := \max_{y \in f^{-1}(1)} C_1(\mathcal{F}, y)$$

The *adaptive learning graph complexity* of $\mathcal{G}$ is $\mathcal{LG}^{adp}(\mathcal{G}) := \sqrt{C_0(\mathcal{E}) C_1(\mathcal{E})}$. The *adaptive learning graph complexity* $\mathcal{LG}^{adp}(f)$ of f is the minimum complexity of an adaptive learning graph for f.

Observe that definition 7 is a special case of definition 10, so $\mathcal{LG}^{adp}(f) \leq \mathcal{LG}(f)$. There is also a dual adversary reduction for adaptive learning graphs [16], and we get a similar upper bound result.

Theorem 15. *For any (possibly partial) Boolean function f, $Q(f) = O(\mathcal{LG}^{adp}(f))$.*

Before the end of the section, we present two important examples of adaptive learning graphs in [16]. The first is the OR of Boolean functions, an example learning graph is depicted in figure 2.1 (a).

Lemma 16 (OR of Boolean Functions)**.** *Let $\mathcal{G}_1, \ldots, \mathcal{G}_k$ be adaptive learning graphs for Boolean functions $f_1, f_2, \ldots, f_k : \{0,1\}^N \to \{0,1\}$ respectively. Let $f = \vee_{i \in [k]} f_i$, suppose that for every 1-input y of f, there are at least l functions f_i such that $f_i(y) = 1$. Then for every $x \in f^{-1}(0), y \in f^{-1}(1)$,*

$$C_0(\mathcal{G}, x) \leq \frac{k}{l} \cdot \mathop{\mathbb{E}}_{i \in [n]} [C_0(\mathcal{G}_i, x) C_1(\mathcal{G}_i)] \quad \text{and} \quad C_1(\mathcal{G}, y) \leq 1. \tag{2.7}$$

The second example is a learning graph version of quantum walks on Johnson graph. The stages in the learning graph of quantum walks are symmetric. The following lemma is a result of lemma 16.

Lemma 17 (Learning Graph Johnson Walk)**.** *Let $l \leq k = o(n)$. For each $A \in \mathcal{S}_k([n])$, let $f_A : \{0,1\}^N \to \{0,1\}$ be a Boolean function. Define $f = \vee_{A \in \mathcal{S}_k([n])} f_A$.*

Let the data structure D be a monotone mapping from $\mathcal{P}([n])$ to $\mathcal{P}([N])$ such that for every 1-input x of f, there is a $I_x \in \mathcal{S}_l([n])$ where $D(I_x)$ is a 1-certificate of x. For λ

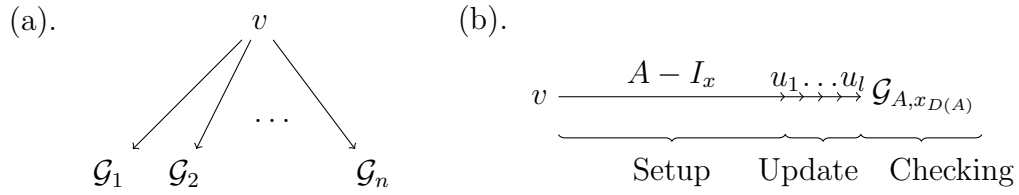


Figure 2.1: (a). Learning Graph of OR of Boolean functions with root L-vertex v. (b). A path of the learning Graph equivalent to MNRS quantum walk on Johnson graph $J(n,k)$ with root L-vertex v. The walk is on a subset $A \in \mathcal{S}(V,k)$ where $I_x = \{u_1, u_2, \ldots, u_l\}$ is a certificate that A is marked.

a partial assignment on $D(A)$, let $f_{A,\lambda}$ be the Boolean function f_A restricted to inputs $z \in \{0,1\}^N$ where $z_{D(A)} = \lambda$. We have $f_A = \vee_\lambda f_{A,\lambda}$ where λ ranges over all partial assignments on $D(A)$. Suppose $\mathcal{G}_{A,\lambda}$ is a learning graph for $f_{A,\lambda}$.

Let $\boldsymbol{S}, \boldsymbol{U}, \boldsymbol{C} > 0$ be values such that for every $x \in f^{-1}(0)$, we have

$$\mathop{\mathbb{E}}_{A \in \mathcal{S}_{k-l}([n])} |D(A)|^2 \leq \boldsymbol{S}^2, \tag{2.8}$$

$$\mathop{\mathbb{E}}_{\substack{A \in \mathcal{S}_i([n]) \\ v \in [n]-A}} |D(A \cup \{v\}) - D(A)|^2 \leq \boldsymbol{U}^2, \text{ for } k-l \leq i < k \tag{2.9}$$

$$\mathop{\mathbb{E}}_{A \in \mathcal{S}_k([n])} \left[C_0(\mathcal{G}_{A,x_{D(A)}}, x) \cdot C_1(\mathcal{G}_{A,x_{D(A)}}) \right] \leq \boldsymbol{C}^2. \tag{2.10}$$

Then there is a learning graph $\mathcal{G}$ for f such that for every $x \in f^{-1}(0), y \in f^{-1}(1)$,

$$C_0(\mathcal{G}, x) = O\left[\boldsymbol{S}^2 + \left(\frac{n}{k}\right)^l \left(k \cdot \boldsymbol{U}^2 + \boldsymbol{C}^2\right) \right] \quad \text{and} \quad C_1(\mathcal{G}, y) \leq 1.$$

Taking a square root of the 0-complexity of $\mathcal{G}$ gives the same complexity bound of the original quantum walk. Figure 2.1 (b). shows a path with positive flows of such a learning graph in stages. Finally, observe that Grover Search is a special case of quantum walk, where we take $k = l = 1$. With setup and update cost 0, and checking cost 1, we get 0-complexity n for unordered searching. Thus, we can also use Grover search as a subroutine in the learning graph context.

Chapter 3

Hypergraph Properties

In this chapter, we ask if there are lower bounds common to properties in a natural class of hypergraph properties. The property classes we look at extend existing graph property classes and we will see how common lower bound techniques generalize when the rank increases. Then, in section 3.3, we use the positive adversary method to give tight lower bounds for deciding connectivity and detecting cycles in r-uniform hypergraphs.

3.1 Subhypergraph-closed Properties

There are many graph properties closed under removing vertices and edges investigated in the quantum query complexity setting. We can naturally generalize this to higher ranks.

Definition 12. An r-uniform hypergraph property $\mathcal{P}$ is *subhypergraph-closed* if whenever $H \leq G$, then $G \in \mathcal{P}$ implies $H \in \mathcal{P}$.

In [17], subgraph-closed properties have an easy $\Omega(n)$ quantum query lower bound. There is a similar result for subhypergraph-closed properties.

Theorem 18. *Every nontrivial, subhypergraph-closed property on r-uniform hypergraphs has quantum query complexity $\Omega(n^{r/2})$.*

Proof. Let $\mathcal{P}$ be such a property. Since $\mathcal{P}$ is nontrivial, there is a constant $d \geq r$ such that for every $n \geq d$, the empty r-hypergraph on n vertices lies in $\mathcal{P}$ but the complete r-hypergraph K_n on n vertices doesn't. Using Ambainis' adversary method in Theorem 9,

we define X as the singleton set consisting of just the empty hypergraph on n vertices and define Y as the set of hypergraphs consisting of one K_d and $n-d$ isolated hyperedges. The size of Y is $\binom{n}{d}$. Note that since $\mathcal{P}$ is a subhypergraph-closed property, $K_d \notin \mathcal{P}$ implies that no hypergraphs in Y are in $\mathcal{P}$. So $X \subseteq \mathcal{P}^{-1}(1)$ and $Y \subseteq \mathcal{P}^{-1}(0)$.

Let R be the complete relation $X \times Y$, we immediately get that $m = |Y| = \Theta(n^d)$ and $m' = 1$. Given empty hypergraph $x \in X$ and potential r-edge $e \in \mathcal{S}(V, r)$, there are exactly $\binom{n-r}{d-r}$ choice of other vertices we can pick to form a complete hypergraph K_d. Thus, $l_{x,e} = \binom{n-r}{d-r}$. Since there is only one element in X, $l'_{y,e} \leq 1$ for any y and e. Thus, $l_{max} = \binom{n-r}{d-r}$ and we conclude that any algorithm deciding $\mathcal{P}$ with constant probability needs

$$\Omega\left(\sqrt{\frac{mm'}{l_{max}}}\right) = \Omega\left(\sqrt{\binom{n}{d}/\binom{n-r}{d-r}}\right) = \Omega\left(\sqrt{n!/(n-r)!}\right) = \Omega(n^{r/2})$$

quantum queries. □

The lower bound for subhypergraph-closed properties is tight. Note that the nontrivial property of being an empty hypergraph is subhypergraph-closed. The negation of this property is equivalent to unordered searching for a hyperedge among the $\binom{n}{r}$ candidates. Grover search gives a simple $O(n^{r/2})$ complexity, matching the equivalent lower bound.

On the other hand, the quantum query complexity of a specific subhypergraph-closed property may be nowhere near this trivial lower bound. Some subgraph-closed properties such as cycle-free, forbidden subgraph, forbidde minor, being r-partite, and having a k-coloring are difficult to decide. We will later analyze the cycle-free problem in section 3.3, and dedicate chapter 4 and 5 to the forbidden simplex problem.

3.2 Monotone Hypergraph Properties

In the last section, we introduced subhypergraph-closed property where subhypergraphs can be obtained by removing vertices and hyperedges. However, vertices are not part of the queries in our model. It is natural to consider properties invariant under removing only hyperedges. The negation of such a property is defined as follows:

Definition 13. An r-uniform hypergraph property $\mathcal{P} : \{0,1\}^{\binom{n}{r}} \to \{0,1\}$ is *monotone* if for every $x \geq y \in \{0,1\}^{\binom{n}{r}}$, we have $\mathcal{P}(x) \geq \mathcal{P}(y)$.

There are monotone hypergraph properties that is not subhypergraph-closed. For example, the properties of being connected, or having a perfect matching are monotone. However, they are not closed under adding new isolated vertices. So monotonicity captures a broader class of properties. Naturally, we want to ask if there is a general lower bound result for monotone hypergraph properties:

Conjecture 1. *For any rank $r \geq 2$, every nontrivial, monotone hypergraph property on r-uniform hypergraphs has quantum query complexity $\Omega(n^{r/2})$.*

In the recent paper by Aaronson, et al. [1], this conjecture is proven true when $r = 2$:

Theorem 19. *For any nontrivial monotone (rank 2) graph properties $\mathcal{P}$, $Q(\mathcal{P}) = \Omega(n)$.*

This is a direct consequence of a known separation between the degree and query complexity of Boolean functions and a degree lower bound of monotone graph properties.

Theorem 20. *[1] For all functions $f : \{0,1\}^n \to \{0,1\}$, we have $\deg(f) = O(Q(f)^2)$.*

Theorem 21. *([34]) If $\mathcal{P}$ is a nontrivial monotone graph property, then $\deg(\mathcal{P}) = \Omega(n^2)$.*

Theorem 21 is proved from an even more general result on Boolean functions:

Theorem 22. *(Rivest, Vuillemin [34]) If $\mathcal{P} : \{0,1\}^N \to \{0,1\}$ is a nontrivial transitive Boolean function where $N = p^\alpha$ is a prime power, then $\deg(\mathcal{P}) = N$.*

Remark 2. In Rivest and Vuillemin's original paper [34], the conclusion of Theorem 22 is the weaker statement $D(\mathcal{P}) = N$ and the conclusion of Theorem 21 is the statement $D(\mathcal{P}) \geq n^2/16 = \Omega(n)$. However, part of the proof in the original paper depends on an intermediate result:

- If $D(\mathcal{P}) \leq k$, then $(1+z)^{n-k} \mid P^1(z)$ where $P^1(z)$ is the weight enumerator

$$P^1(z) := \sum_{i=0}^{n} w_i(\mathcal{P}) \cdot z^i, \quad w_i := |\{x : \mathcal{P}(x) = 1 \text{ and } w(x) = i\}|.$$

This intermediate result can be modified to replace $D(\mathcal{P})$ by $\deg(\mathcal{P})$. Let $p(z)$ be the polynomial representing $\mathcal{P}$ exactly. If $\deg(\mathcal{P}) \leq k$, note that

$$P^1(z) = \sum_{0^n \leq x \leq 1^n} \mathcal{P}(x) \cdot z^{|x|} = \sum_{\substack{m: \\ \text{monomial of } p}} \sum_{0^n \leq x \leq 1^n} m(x) \cdot z^{|x|}.$$

If $m(x) = c_i x_{i_1} x_{i_2} \dots x_{i_j}$ is a monomial of degree $j \leq k$, then $\sum_{0^n \leq x \leq 1^n} m(x) \cdot z^{|x|} = \sum_{0^j \leq x \leq 1^j} c_i x_1 \dots x_j z^{x_1+\dots+x_j} \sum_{0^{n-j} \leq y \leq 1^{n-j}} z^{y_1+\dots+y_{n-j}} = c_i z^j (1+z)^{n-j}$, which is divisible by $(1+z)^{n-k}$. Thus $P^1(z)$ is divisible by $(1+z)^{n-k}$. This modification with the arguments in [34] proves Theorem 22 and 21. △

To generalize the result of Theorem 19 to higher ranks, we want to see how well we can lower bound the degree of a monotone hypergraph property. To our knowledge, the closest result we get for extending Theorem 21 is discovered by Kulkarni, et al. [26] on 3-uniform hypergraphs:

Theorem 23. *If $\mathcal{P} : \{0,1\}^{\binom{n}{3}} \to \{0,1\}$ is a nontrivial monotone property of 3-uniform hypergraphs, then $D(\mathcal{P}) = \Omega(n^3)$.*

It remains to replace the deterministic query complexity of $\mathcal{P}$ by the degree of $\mathcal{P}$ like we did in Remark 2. Unfortunately, we cannot get away with the prime power restriction in Theorem 22 with the ways 3-hyperedges behave. Instead, we partially solve Conjecture 1 and show why is it difficult to generalize our arguments when the rank is 3.

Theorem 24. *For any rank $r \geq 2$ and any nontrivial monotone property $\mathcal{P}$ on r-partite r-uniform hypergraphs with rn vertices, we have $\deg(\mathcal{P}) = \Omega(n^r)$. Moreover, $\mathcal{P}$ has quantum query complexity $\Omega(n^{r/2})$.*

The proof below extends the proof of 4-uniform hypergraph theorem in [26]. It also presents the lower bound technique used to analyze monotone properties in general.

Proof. The second statement follows from the first statement and Theorem 20. To prove the lower bound on the degree of $\mathcal{P}$, let $V_1, \dots, V_r$ be an r-partition of the hypergraph where each V_k has size n. Let S_0 be the empty hypergraph and $S_{2^r} := K_{n,\dots,n}$ be the complete r-partite r-hypergraph on rn vertices. Since $\mathcal{P}$ is nontrivial, pick n large enough so that $\mathcal{P}(S_0) = 0$ and $\mathcal{P}(S_{2^r}) = 1$.

Note that by Bertrand-Chebyshev Theorem(in number theory), there is a prime number p between $n/2$ and n. For each $k \in [r]$, further partition each V_k into two sets $V_{k,0}, V_{k,1}$ where $|V_{k,0}| = p$ and $|V_{k,1}| = n - p$. Let $z^1 = 00\dots0$, $z^2 = 10\dots0$, $z^3 = 010\dots0$, $z^4 = 110\dots0$, $\dots$, $z^{2^r} = 11\dots1$ be the (flipped) lexicographical listing of Boolean strings $\{0,1\}^r$. Consider a finite sequence of hypergraphs $S_0, S_1, S_2, \dots, S_{2^r}$ where for each $i \in [2^r]$, S_i is obtained by adding the hyperedges in $V_{1,z_1^i} \times V_{2,z_2^i} \times \dots \times V_{r,z_r^i}$ to S_{i-1}. For different

i, the above set of hyperedges are disjoint and their union is $V_1 \times \cdots \times V_r = K_{n,\ldots,n}$. Let $\mathcal{P}_i$ be the restriction of $\mathcal{P}$ to the set of hypergraphs

$$\left\{S \subseteq V_1 \times \cdots \times V_r : S_{i-1} \subseteq S \text{ and } \overline{S_i} \subseteq \overline{S}\right\}$$

where $\overline{S}$ denotes the complement of S. These $\mathcal{P}_i$ are also hypergraph properties with respect to the r-partition of vertices $\left(V_{1,z_1^i}, V_{2,z_2^i}, \ldots, V_{r,z_r^i}\right)$. Since the restriction of a monotone property is also monotone, each $\mathcal{P}_i$ is monotone.

Suppose for a contradiction that $\deg(\mathcal{P}) = o(n^r)$. Let $q(x_1, \ldots, x_{n^r})$ be the polynomial that corresponds to $\mathcal{P}$. For $i \in [2^r]$, to obtain the polynomial q_i corresponding to property $\mathcal{P}_i$, we partially evaluate q in the following ways.

- If x_j is a variable that corresponds to an r-edge in S_{i-1}, set $x_j = 1$.
- If x_j is a variable that corresponds to an r-edge not in S_i, set $x_j = 0$.

Note that the degree of the polynomial can only decrease. So $\deg(q_i) = o(n^r)$.

Since $\mathcal{P}$ is a nontrivial monotone property, and the sequence $S_0, S_1, S_2, \ldots, S_{2^r}$ is obtained by repeatedly adding hyperedges, there has to be an index $i \in [2^r]$ such that $\mathcal{P}(S_{i-1}) = 0$ and $\mathcal{P}(S_i) = 1$. If $i = 1$, the size of S_1 is the prime power p^r. $\mathcal{P}_1$ is transitive because for every $v_1 v_2 \ldots v_n, v_1' v_2' \ldots v_n' \in S_1$, there is permutation $\pi = (\pi_1, \ldots, \pi_r) \in S_n \times \cdots \times S_n$ such that $\pi_k(v_k) = v_k'$ for all $k \in [r]$. By Theorem 22, $\deg(\mathcal{P}_1) = p^r = \Omega(n^r)$, a contradiction. If $i \geq 2$, the size of the domain of $\mathcal{P}_i$ may not be a prime power. We can fix this by patching the domain. For each i where $z_k^i = 1$, add $2p - n$ additional vertices from $V_{i,0}$ to $V_{i,1}$. Define a new property $\mathcal{P}_i'$ that extends $\mathcal{P}_i$ by ignoring the values of new r-tuples. Let q_i' be the polynomial corresponding to $\mathcal{P}_i'$. Note that by the way we define $\mathcal{P}_i'$, q_i' has no new variables and it equals q_i. The property $\mathcal{P}_i'$ is nontrivial and monotone, but now its domain has size p^r. So, similar to the $i = 1$ case, we can easily argue that $\mathcal{P}_i'$ is transitive and $\deg(\mathcal{P}_i) = \deg(\mathcal{P}_i') = \Omega(n^r)$, a contradiction. □

When $r = 3$, in a general hypergraph, we may encounter a hyperedge with 2 vertices in one partition and 1 vertex in another, depicted in figure 3.1 (a). The orbit of this hyperedge under vertex-permutations within each partition has size $\binom{n}{2} \cdot n$. For arbitrary n, this is not a prime power and cannot be made a prime power by adding new vertices. Therefore a new method is needed for lower bounding degrees of hyperedges in general and Conjecture 1 remains open.

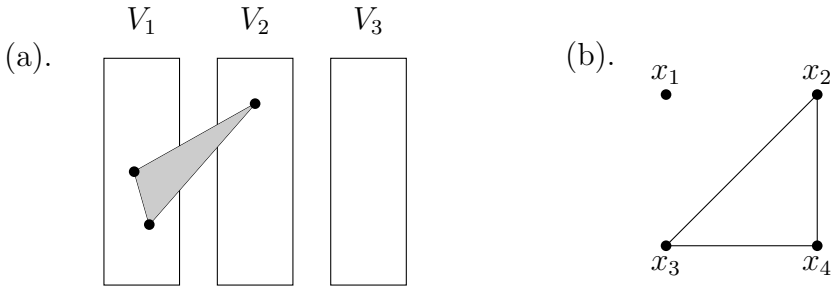


Figure 3.1: (a). In a 3-hypergraph with 3 sets of vertices V_1, V_2, V_3. Under the permutation of vertices within each set, the hyperedge shaded gray has an orbit of size $\binom{n}{2} \cdot n$. This is not a prime power, so this type of 3-edge prevent the use of Theorem 22. (b). A simplicial complex on vertex set $X = \{x_1, x_2, x_3, x_4\}$. It contains a cycle so it is not contractible. By the result of [23], all four vertices need to be queried to decide the membership of a face $A \subseteq X$. However, the polynomial of this simplicial complex is $x_1x_2x_3 + x_1x_3x_4 + x_1x_2x_4 - x_2x_3x_4 - x_1x_3 - x_1x_2 - x_1x_4 + 1$. Its degree is less than 4, so the modified result with degree nonevasiveness doesn't hold.

Remark 3. The proof of Theorem 23 relies on another lower bound approach using a topological argument in [23]. However, this approach doesn't involve a generating function and no natural modification to the degree result exists. In fact, the topological approach to lower bound in [23] relies on a crucial observation:

- Any nonevasive simplicial complex is contractible.

Here, given a finite set X, a simplicial complex Δ is just a set of subsets of X with the property that $B \in \Delta$ and $A \subseteq B$ implies $A \in \Delta$. Contractible means the complex can be continuously deformed to a point. To modify the result for degree lower bound, we have to change the word "nonevasive" to "degree nonevasive". However, a counterexample is given in figure 3.1 (b) for this modification. So the topological approach failed for degrees. $\triangle$

3.3 Hypergraph Connectivity and Acyclicity

In the last two sections, we give an $\Omega(n^{r/2})$ lower bound for some general classes of hypergraph properties. However, these results don't capture the difficulties of searching for large certificates in a nontrivial hypergraph. Here, we will prove stronger lower bound for two of the more specific but natural hypergraph properties:

r-HG Connectivity:

Given an r-uniform hypergraph $G = (V, E)$, decide if G is connected.

r-HG Acyclicity:

Given an r-uniform hypergraph $G = (V, E)$, decide if G contains no cycle.

These two problems have already been well studied in the context of rank-2 simple graphs [18], [17]. We will generalize the ideas presented in these two papers to find the tight quantum query complexity bounds of these two hypergraph problems.

Note that the r-**HG Connectivity** problem is monotone and the r-**HG Acyclicity** problem is subhypergraph closed. Despite that, for the r-**HG Connectivity** problem, each hyperedge can connect up to $r = O(1)$ vertices. If the hypergraph is connected, it can only be certified with a spanning subhypergraph with $\Omega(n)$ hyperedges. Similarly, a cycle of a hypergraph may have size n, forcing a large 0-certificate size in r-**HG Acyclicity**.

Theorem 25. *For any rank $r \geq 2$, we have $Q(r$-**HG Connectivity**$) = \Omega\left(n^{\frac{r+1}{2}}\right)$.*

Proof. When $r = 2$, the theorem is proved in [18], so we assume $r \geq 3$. The proof uses Ambainis' adversary method given in Theorem 9. First, assume the n vertices of a hypergraph are given by a disjoint union $A \cup B$ where $|A| = \frac{n}{2r-3}$ and $|B| = \frac{(2r-4)n}{2r-3}$. Define X as the subset of hypergraphs on n vertices constructed in the following way:

1. Arrange vertices of A in a cycle.
2. For every pair $\{u, v\} \subseteq A$ adjacent by the construction of step 1, pick $r-2$ vertices $u_3, \ldots, u_r \in B$ not already picked by some other pairs in this step and form the r-hyperedge $uvu_3 \ldots u_r$. There are exactly $\frac{(r-2)n}{2r-3}$ vertices in B (which is half of the vertices in B) picked in this step.
3. Pair up the remaining half of the vertices in B into $\frac{n(r-2)}{(2r-3)(r-1)}$ groups of $r-1$. For each $(r-1)$-tuple of vertices $(u_1, \ldots, u_{r-1})$ pick a $z \in A$ that is not already picked by some other tuples in this step and form the hyperedge $zu_1, \ldots, u_{r-1}$.

For convenience, we call the hyperedges created in step 2 the **ring hyperedges** because these hyperedges formed a cycle with vertices in A. We call the hyperedges created in step 3 the **side hyperedges**. Define the subset Y similarly, except in step 1, we partition the

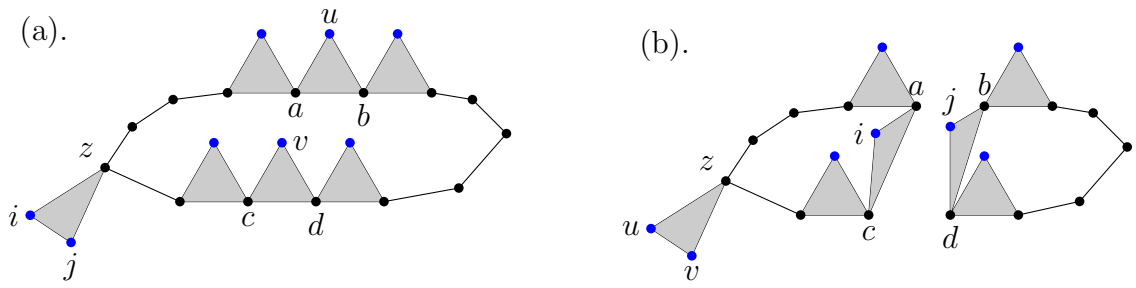


Figure 3.2: Partial Diagram of example hypergraph in X and Y. Vertices in A are colored black and vertices in B are colored blue. The gray-shaded triangles are the 3-hyperedges. Only the relevant hyperedges are shown in these diagrams. (a). depicts a connected hypergraph G in X. (b). depicts a disconnected hypergraph G' in Y that is obtained from G by removing abu, cdv, zij and adding aci, bdj, zuv.

vertices of A into $r-1$ cycles, each of length between $\frac{n}{(2r-3)^2}$ and $\frac{2n}{(2r-3)^2}$. This is possible because $\frac{(r+1)n}{(2r-3)^2} \leq |A| \leq \frac{2(r+1)n}{(2r-3)^2}$ when $r \geq 3$. Figure 3.2 illustrates an example hypergraph from A and B each when the rank is 3. It is easy to see that hypergraphs in X are connected while hypergraphs in Y have $r-1 \geq 2$ connected components.

Now, we define R so that for $G \in X$, $G' \in Y$, we have $(G, G') \in R$ if and only if

1. we can choose $r-1$ ring hyperedges $e_1, \ldots, e_{r-1}$ where e_i is given by the vertices $a_i b_i u_{i,1} \ldots u_{i,r-2}$, with $a_i, b_i \in A$ and $u_{i,1}, \ldots, u_{i,r-2} \in B$, and
2. we can choose $r-2$ side hyperedges $f_1, \ldots, f_{r-2}$ where f_j is given by the vertices $z_j v_{j,1} \ldots v_{j,r-1}$, with $z_j \in A$ and $v_{j,1}, \ldots, u_{j,r-2} \in B$, such that
3. G' is given by removing the hyperedges $e_1, \ldots, e_{r-1}, f_1, \ldots, f_{r-2}$ from $E(G)$, and
4. adding $r-1$ ring hyperedges $e'_1, \ldots, e'_{r-1}$ where for $i \in [r-1]$, e'_i is given by $a_i b_{i+1} v_{1,i} \ldots v_{r-2,i}$ (b_{i+1} becomes b_1 when $i = r-1$), and
5. adding $r-2$ side hyperedges $f'_1, \ldots, f'_{r-2}$ where for $j \in [r-2]$, f'_j is given by $z_j u_{1,j} u_{2,j} \ldots u_{r-1,j}$.

We remark that although the above construction makes use of the ordering of vertices labeled by $u_{i,j}$ and $v_{i,j}$, the ordering doesn't matter as the value of m, m', l, l' will change

by a value depending on only the constant r. This conversion is illustrated in figure 3.2 when the rank is 3.

Given $G \in X$, there are $\Theta(n^{r-1})$ choices of the $r-1$ ring hyperedges of G that can be selected in the above conversion to break down A into $r-1$ cycles each of length between $\frac{n}{(2r-3)^2}$ and $\frac{2n}{(2r-3)^2}$. There are $\Theta(n^{r-2})$ choices of side hyperedges f_j. This gives at least $m = \Theta(n^{2r-3}$ many $G' \in Y$ related to G by R. Similarly, given $G' \in Y$, we can get a hypergraph in X by choosing one ring hyperedge from each one of the $r-1$ cycles of G', then choose $r-2$ side hyperedges from the $\Theta(n^{r-2})$ possibilities. This gives at least $m' = \Theta(n^{2r-3})$ many $G \in X$ related to G' by R.

Now given $(G, G') \in R$ and an r-tuple of vertices $e \in \mathcal{S}_r(V)$ where $\mathcal{O}_G(e) \neq \mathcal{O}_{G'}(e)$. There are four cases:

1. If $e = a_i b_i u_{i,1} \dots u_{i,r-2}$ is a ring hyperedge of G. Then G' is given by picking the remaining $r-2$ ring hyperedges in $\Theta\left(n^{r-2}\right)$ ways and picking $r-2$ side hyperedges in one of the $\Theta(n^{r-2})$ ways. So $l_{G,e} = \Theta(n^{2r-4})$. Note that $e \notin E(G')$, but e has to be a ring hyperedge to be added to G' because it has two vertices in A. In this case, a_i, b_i ensure that there are only 4 possible choices of ring hyperedges to be removed in 2 of the $r-1$ cycles of G'. Furthermore, by the conversion defined above, the vertices $u_{i,1}, \dots, u_{i,r-2}$ each uniquely determine a distinct side hyperedge to be removed in G'. Thus, it remains to choose one hyperedge to remove in each of the remaining $r-3$ cycles. This makes $l_{G',e} = \Theta(n^{r-3})$.

2. If $e = a_i b_{i+1} v_{1,i} \dots v_{r-2,i}$ is a ring hyperedge to be added to G. We can show that $l_{G,e} = \Theta(n^{r-3})$, and $l_{G',e} = \Theta(n^{2r-4})$ in a similar way case 1 does.

3. If $e = z_j v_{j,1} \dots v_{j,r-1}$ is a side hyperedge of G. Then G' is given by choosing $r-1$ ring hyperedges to remove in $\Theta(n^{r-1})$ ways and choosing the remaining $r-3$ side hyperedges in $\Theta(n^{r-3})$ ways. So $l_{G,e} = \Theta(n^{2r-4})$. On the other hand , $e \notin G'$ and it is to be added to G'. This means $v_{j,1}, \dots, v_{j,r-1}$ are each incident to a distinct ring hyperedge to be removed from G'. Since $z_j \in A$ is uniquely identified by e, the side hyperedge in G' containing z_j will be removed. So, we are left with picking $r-3$ more side hyperedges from one of the $\Theta(n^{r-3})$ possibilities. This gives $l_{G',e} = \Theta(n^{r-3})$.

4. If $e = z_j u_{1,j} u_{2,j} \notin G$ is a side hyperedge to be added to G. We can show that $l_{G,e} = \Theta(n^{r-3})$, and $l_{G',e} = \Theta(n^{2r-4})$ in a similar way case 3 does.

We have shown that, in all four cases above, $l_{max} = l_{G,e} l_{G',e} = \Theta(n^{3r-7})$. By Ambainis' adversary method, deciding connectivity for r-uniform hypergraph has quantum query

complexity

$$\Omega\left(\sqrt{\frac{mm'}{l_{max}}}\right) = \Omega\left(\sqrt{\frac{n^{2r-3}\cdot n^{2r-3}}{n^{3r-7}}}\right) = \Omega\left(n^{\frac{r+1}{2}}\right).$$

□

In Childs and Kothari's minor-closed property paper [17], it was shown that testing cyclicity is difficult because it is hard to distinguish a long path from a long cycle. This is also true in the hypergraph setting. We will modify the above proof to get a similar result:

Theorem 26. *For any rank $r \geq 2$, we have $Q(r$-**HG Acyclicity**$) = \Omega\left(n^{\frac{r+1}{2}}\right)$.*

Proof. When $r = 2$, the theorem is proved in [17], so we assume $r \geq 3$. Consider the proof of Theorem 25 with the following modifications:

1. In step 1 of the construction of X, arrange the vertices of A in a simple path instead of a cycle.
2. In step 1 of the construction of Y, arrange the vertices of A into $r-2$ cycles and 1 simple path, each of length between $\frac{n}{(2r-3)^2}$ and $\frac{2n}{(2r-3)^2}$.

In this case, X is a subset of spanning trees on n vertices and Y consists of hypergraph with $r-2 \geq 1$ cycles, so $X \subseteq f^{-1}(1)$ and $Y \subseteq f^{-1}(0)$.

Note that the graphs in X, Y changed in this proof by removing 1 hyperedge in one of the original cycles in A, there is now 1 less hyperedge to choose from in the analysis of values m, m', l, l'. This minor modification doesn't alter the asymptotic bound of the values. So the rest of the proof follows from the proof of Theorem 25. □

Finally, we will show that the lower bounds above are actually tight by giving the quantum query algorithm with matching matching complexity.

Theorem 27. *For any rank $r \geq 2$, we have $Q(r$-**HG Connectivity**$) = O\left(n^{\frac{r+1}{2}}\right)$.*

Proof. Given an r-uniform hypergraph G, consider the following algorithm. First, let V be the vertex set of G and let A be a set of hyperedges that is initially empty. At the beginning, each vertex is its own connected component, and we are trying to connect at least two components at each iteration of the algorithm. Suppose at stage i, $A_1, \ldots, A_k$

are the connected components of $G' = (V, A)$, we are trying to find a hyperedge $e \in E(G)$ such that there is $i \neq j$ with $A_i \cap e, A_j \cap e \neq \emptyset$ with Grover search. If G is connected, since each r-hyperedge can connect at most $r = O(1)$ connected components, there are at least $\frac{k-1}{r-1}$ such hyperedges e. So by tight Grover search in Theorem 2, we can find such a hyperedge in $O\left(\sqrt{\frac{(r-1)\binom{n}{r}}{(k-1)}}\right)$ expected queries. Thus, in at most $n-1$ iteration of the above Grover search, we can find a spanning subhypergraph of G in

$$\sum_{i=2}^{n} \sqrt{\frac{(r-1)n^r}{i-1}} = O(n^{r/2}) \sum_{i=1}^{n-1} \sqrt{i^{-1}} \leq O\left(n^{r/2}\right) \left(1 + \int_1^{n-1} \sqrt{i^{-1}} di\right) = O\left(n^{\frac{r+1}{2}}\right)$$

expected number of queries. For some appropriate constant c, if the procedure takes $cn^{\frac{r+1}{2}}$ queries without finding such a spanning subgraph, terminate the procedure and conclude that G is not connected. This conclusion only errs with constant probability by Markov inequality. □

Theorem 28. *For any rank $r \geq 2$, we have $Q(r$-**HG Acyclicity**$) = O\left(n^{\frac{r+1}{2}}\right)$.*

Proof. Observe that if G is acyclic, it has to be a forest, so m, which is the number of hyperedges in G, must be at most n. The first step of the algorithm is to apply Quantum Counting in Theorem 3 to the hyperedge oracle to approxiamately count the number of hyperedges in G. Fix some small error ϵ, we can decide if $m \geq (1+\epsilon)n$ with constant probability in $O(\sqrt{n^r/n}) = O(n^{\frac{r-1}{2}})$ expected numbers of queries. If $m \geq (1+\epsilon)n$, we conclude that there must be a cycle in G. If not, G is a sparse hypergraph and we try to find all hyperedges of G by repeatedly running the tight Grover Search Algorithm for hyperedges not yet found. If G has t hyperedges, then the i^{th} hyperedge can be found in $O(\sqrt{n^r/(t-i+1)})$ expected number of queries. Since $t \leq n$, the total expected number of queries needed is

$$\sum_{i=1}^{t} O\left(\sqrt{\frac{n^r}{t-i+1}}\right) = O(n^{r/2}) \sum_{i=1}^{t} \sqrt{i^{-1}} \leq O(n^{r/2}) \sum_{i=1}^{n} \sqrt{i^{-1}} \leq O\left(n^{\frac{r+1}{2}}\right).$$

Terminate the repeated Grover search procedure after $cn^{\frac{r+1}{2}}$ queries for some appropriate constant c, we can obtain all hyperedges of G with constant probability due to Markov inequality. It remains to check if the hyperedges form a cycle without further queries. □

Chapter 4

Upper Bound on Finding n-simplices

In this chapter and the next, we study the quantum query complexity of finding a fixed, constant-sized subhypergraph. We call an $(r+1)$-clique on an r-uniform hypergraph an *r-simplex*. Specifically, a 2-simplex is a triangle, and a 3-simplex is called a *tetrahedron*. Similar to the triangle finding problem, we are interested in the problem of finding r-simplicies for arbitrary rank $r \geq 2$.

Tetrahedron Finding Problem (Tetrahedron):

> Given a 3-uniform hypergraph $G = (V, E)$ with n vertices and m hyperedges. Suppose the hyperedge set E is given as the oracle input as a subset of $\mathcal{S}_3(V)$. Find four distinct vertices a, b, c, d such that $abc, abd, acd, bcd \in E$ or decide that no such vertices exist.

Given vertices $\{v_1, \ldots, v_r\}$, we use $e_{\hat{i}}$ to denote the subset $v_1 v_2 \ldots v_{i-1} v_{i+1} \ldots v_{r-1} v_r \in \mathcal{S}_{r-1}(V)$. We write e_S for $S \subseteq [r]$ to denote the subset $\{v_i : i \in S\} \in \mathcal{S}_{|S|}(V)$. For vertex $u \in V$, we use ue_S to abbreviate the subset $\{u\} \cup e_S$.

r-Simplex Finding Problem (r-SF):

> Given a r-uniform hypergraph $G = (V, E)$ with n vertices and m hyperedges. Suppose the hyperedge set E is given as the oracle input as a subset of $\mathcal{S}_r(V)$. Find $r+1$ distinct vertices $v_1, v_2, \ldots, v_{r+1}$ such that $e_{r\hat{+}1}, e_{\hat{r}}, \ldots, e_{\hat{1}} \in E$ or decide that no such vertices exist.

Just like the triangle finding problem, there is a trivial quantum algorithm and a trivial quantum lower bound for r-simplex finding. Since a 1-certificate of an r-simplex is given by finding an $(r+1)$-subset of vertices and checking $r+1$ possible r-edges in this subset. We can find an r-simplex in G by Grover searching the function $F : \mathcal{S}_{r+1}(V) \to \{0,1\}$ where $F(v_1, \ldots, v_{r+1}) = 1$ if and only if $e_{r\hat{+}1}, e_{\hat{r}}, \ldots, e_{\hat{1}} \in E$. Each query to F takes $r+1$ queries to the hypergraph oracle, therefore

$$Q(\textbf{r-SF}) = O\left(\sqrt{\binom{n}{r+1}(r+1)}\right) = O\left(n^{\frac{r+1}{2}}\right). \tag{4.1}$$

For the lower bound, we consider the r-hypergraph $G = (V, E)$ where the vertex set V is given by $\{v_0, v_1, \ldots, v_{n-1}\}$. For each subset of indices $S = \{i_1, \ldots, i_{r-1}\} \in \mathcal{S}_{r-1}([n-1])$, we are promised that $e_{\{0\}\cup S} = v_0 v_{i_1} \ldots v_{i_{r-1}} \in E$. Under this promise, to find an r-simplex in G, it is necessary and sufficient to find an r-edge in $\mathcal{S}_r(\{v_1, v_2, \ldots, v_{n-1}\})$. This is equivalent to unordered searching the function $F : \mathcal{S}_r(\{v_1, \ldots, v_{n-1}\}) \to \{0,1\}$ where each query to F is equivalent to one query to the hypergraph oracle. This search requires $\Omega(\sqrt{(n-1)^r}) = \Omega(n^{r/2})$ queries by Example 3. Since making promises on the r-hypergraphs can only reduce query complexity, we obtain the lower bound

$$Q(\textbf{r-SF}) = \Omega\left(n^{r/2}\right). \tag{4.2}$$

These two results are fairly straightforward, the more interesting problem is to find the exponent $\frac{r}{2} \le a \le \frac{r+1}{2}$ where $Q(\textbf{r-SF}) = O(n^a \cdot g(n)) \cap \Omega(n^a / g(n))$ for some subpolynomial factor $g(n)$, or at least reduce the range we have on this exponent a. Usually, $g(n)$ is a poly-logarithmic factor and we use the tilde sign $\tilde{\Theta}(n^a)$ to hide such a poly-logarithmic factor. In sections 2.3 and 2.5, we were presented with other techniques for generating quantum query algorithms other than Grover search. For the rest of this chapter, we utilize these techniques and present some nontrivial algorithms solving the r-simplex finding problem for some specific rank $r \ge 3$.

4.1 Tetrahedron Finding with Vertex Quantum Walk

In this subsection, we try to build an algorithm for 3-simplex finding by generalizing existing triangle finding algorithms. Some well-known quantum query complexity results for the **Triangle** problem are listed below. They all rely on either the quantum walk or the non-adaptive learning graph technique.

- [31] Magniez, Santha, and Szegedy's $O(n^{1.3})$ algorithm with MNRS quantum walk. In fact, this result preceded and inspired MNRS's paper on general quantum walk.
- [9] Belovs' $O(n^{35/27})$(where $\frac{35}{27} \approx 1.296$) learning graph algorithm for finding triangle achieved by using random subgraphs.
- [28] Lee, Magniez, and Santha's $O(n^{9/7})$(where $\frac{9}{7} \approx 1.286$) learning graph algorithm for finding triangle improving Belovs' $O(n^{35/27})$ result by allowing more parameterization in the database used during the algorithm.
- [19] Le Gall improved the query complexity to $\tilde{O}(n^{5/4}) = \tilde{O}(n^{1.25})$ using the quantum walk approach and combinatorial arguments related to triangle structures. The polylogorithmic factor of this upper bound can be removed [16].

The $O(n^{5/4})$ upper bound is currently best for the **Triangle** problem.

Unfortunately, as the rank r increase, any algorithm that wish to achieve a nontrivial query upper bound is much harder to come up with. Most triangle finding algorithms don't generalize to r-simplex finding since the combinatorial structure associated with an r-simplex is more complicated. This difficulty is evident even when $r = 3$. In this section, we show how Le Gall's $\tilde{O}(n^{5/4})$ algorithm needs to be modified to get a slightly nontrivial algorithm for tetrahedron finding. The proof of this algorithm is quite involved, but it has been simplified in [16] by casting the combinatorial arguments and quantum walks into an equivalent adaptive learning graph algorithm. That's how we will present our algorithm.

Before we start, let's first make some notational definitions. Let

$$T_{u,v,w} := \{z \in V : uvz, vwz, uwz \in E\}. \tag{4.3}$$

Let $X, A, B, C \subseteq V$, where $|X| = n^x, |A| = n^a, |B| = n^b, |C| = n^c$. Define

$$\Delta(X) := \{(u, v, w) \in \mathcal{S}_3(V) : T_{u,v,w} \cap X = \emptyset\}. \tag{4.4}$$

Let $\Delta(X, B) := (B^2 \times V) \cap \Delta(X)$, this means that w can be taken to be any vertex. Define

$$\Gamma(X, B, w, z) := \{(u, v) \in \mathcal{S}_2(B) : (u, v, w) \in \Delta(X) \text{ and } z \in T_{u,v,w}\}. \tag{4.5}$$

A trivial nested quantum walk that search for all 4 vertices in a tetrahedron of the 3-hypergraph has quantum query complexity equivalent to a simple Grover search. To utilize quantum walks in a meaningful way, we want the inner quantum walks to handle easier update and checking tasks and assign more heavy-duty searches to the outer walks. This idea can be realized because of the following lemma.

Lemma 29. *Fix a subset $B \subseteq V$ of size n^b. For a subset $X \subseteq V$ of size n^x and $w, z \in \mathcal{S}_2(V)$, all taken uniformly at random, we have*

$$\mathop{\mathbb{E}}_{X,w,z} \left[\left| \Gamma(X, B, w, z) \right| \right] \leq n^{2b-x}.$$

Proof. Note that

$$\begin{aligned}
\mathop{\mathbb{E}}_{X,w,z} \left[\left| \Gamma(X, B, w, z) \right| \right] &= \sum_{uv \in \mathcal{S}_2(B)} \Pr_{X,w,z} \left[uv \in \Gamma(X, B, w, z) \right] \\
&= \sum_{uv \in \mathcal{S}_2(B)} \sum_{w \in V - \{u,v\}} \Pr_{X,z} \left[(u, v, w) \in \Delta(X) \text{ and } z \in T_{u,v,w} | w \right] \Pr(w) \\
&= \sum_{uv \in \mathcal{S}_2(B)} \sum_{w \in V - \{u,v\}} \Pr_{X} \left[(u, v, w) \in \Delta(X) \right] \Pr_{z} \left[z \in T_{u,v,w} \right] \Pr(w)
\end{aligned}$$

The last equality holds because the two events inside the probability are independent. Fix u, v, w, let $t = |T_{u,v,w}|$, then $\Pr_z \left[z \in T_{u,v,w} \right] = \Theta(t/n)$. Moreover, since elements in X are taken uniformly at random, we have $\Pr_X \left[(u, v, w) \in \Delta(X) \right] = \Theta \left[(1 - t/n)^{n^x} \right]$. Take $\alpha = tn^{x-1}$, then $\Pr_{X,z} \left[(u, v, w) \in \Delta(X) \text{ and } z \in T_{u,v,w} \right] = \frac{\alpha}{n^x} \left[1 - \frac{\alpha}{n^x} \right]^{n^x} \leq \frac{\alpha e^{-\alpha}}{n^x} \leq n^{-x}$. So the expected value is less than or equal to

$$\sum_{uv \in \mathcal{S}_2(B)} \sum_{w \in V - \{u,v\}} \frac{1}{n^x} \Pr(w) \leq n^{2b-x}.$$

□

The main result of this section is to provide an algorithm that beats the trivial $O(n^2)$ query complexity for the **Tetrahedron** problem.

Theorem 30. *There is an adaptive learning graph algorithm for computing the tetrahedron finding problem with $O(n^{1.9})$ quantum queries.*

Proof. Let G be a 3-hypergraph on vertex set V, and let $0 \leq x, a, b, c < 1$ be real parameters. Assume that G has a tetrahedron, let $u, v, w, z \in \mathcal{S}_4(V)$ be its vertices and a 1-certificate is the set of 3-edges $\{uvw, uvz, uwz, vwz\}$. Let $f : \{0,1\}^{\mathcal{S}_r(V)} \to \{0,1\}$ be the Boolean function where $f(G) = 1$ if and only if G contains a tetrahedron as subhypergraph.

The $O(n^{1.9})$ upper bound for f is given by an adaptive learning graph $\mathcal{G}$ with the matching complexity. $\mathcal{G}$ is defined by 8 stages. For stage $i \in [8]$, let V_{i-1}, V_i be the set of vertices at the beginning, end of the stage, respectively. We will describe each stage in detail.

Stage 1: The first step of the algorithm is to choose a subset $X \subseteq V$ of size n^x uniformly at random. This is formalized in $\mathcal{G}$ by the OR of Boolean functions learning graph (lemma 16) where all branches compute **Tetrahedron**, so $k = l = \binom{n}{n^x}$. This stage loads no elements and the subset X is included in the lebels of L-vertices in V_1 as ancillary information. 0-complexity of $\mathcal{G}$ is unaffected by this stage.

Stage 2: In this stage, we search for a tetrahedron vertex z in X. Given any 4 vertices, we can tell if they form a tetrahedron with just 4 queries to O_G. Thus we can find such a vertex with Grover search on the set of 4-tuples $X \times \mathcal{S}_3(V)$. If X has a tetrahedron vertex, the search will return true with constant probability. Otherwise, no triple of vertices $uvw \in \mathcal{S}_r(V)$ where $T_{u,v,w} \cap X \neq \emptyset$ can be a 3-edge of G. So it suffices to find a tetrahedron in $\Delta(X)$.

Formally, every L-vertices in V_1 branches into two cases by lemma 16. First, if $\{u, v, w, z\} \cap X \neq \emptyset$, the tetrahedron can be found by a learning graph $\mathcal{G}_{X \times \mathcal{S}_3(V)}$ representing Grover search with 0-complexity n^{3+x}. If $\{u, v, w, z\} \cap X = \emptyset$, we describe how to find this tetrahedron in $\Delta(X)$ in the remaining stages.

Stage 3: Perform Quantum Walk on the Johnson Graph $J(n, n^a)$ where the subset $A \in \mathcal{S}(n, n^a)$ is marked if it has a tetrahedron vertex z. We do not keep a data structure for this walk, so it has setup and update cost 0. Using lemma 32 with parameters $n' = n, k' = n^a, l' = 1$, the 0-complexity of this stage is $O(n^{1-a} \cdot \boldsymbol{C}^2)$ where the checking procedure is given by another quantum walk in the remaining stages.

Stage 4: Perform Quantum Walk on the Johnson Graph $J(n, n^b)$ where the subset $B \in \mathcal{S}(n, n^b)$ is marked if there is a pair of vertices $I_G = \{u, v\}$ in $\Delta(X, B)$ that belongs to a tetrahedron. In this level of quantum walk, we keep a data structure that stores the queried triples of vertices $D(B) = X \times \mathcal{S}_2(B) \cup A \times \mathcal{S}_2(B)$. Let $m := \max(x, a)$, the setup cost is $\boldsymbol{S} = |D(B)| = O(n^{2b+m})$. When a vertex s is added to B or removed from B, we need to query the set of triples $\{s\} \times X \times B \cup \{s\} \times A \times B$. So the update cost is $\boldsymbol{U} = O(n^{b+m})$. Checking is described in the remaining stages. With parameters $n' = n, k' = n^b, l' = 2$, lemma 32 shows that the 0-complexity of this stage is

$$O\left[\boldsymbol{S}^2 + \left(\frac{n}{n^b}\right)^2 \left(n^b \cdot \boldsymbol{U}^2 + \boldsymbol{C}^2\right)\right] = O\left[n^{4b+2m} + n^{2-2b}\left(n^{3b+2m} + \boldsymbol{C}^2\right)\right].$$

Stage 5: Choose and load a vertex w from V uniformly at random. This is again achieved by taking Or of functions by lemma 16. The vertex w is valid (that is, it receives positive

flow from p_G in $\mathcal{G}$) if it forms a tetrahedron with some $z \in A, u, v \in B$. Since there may be only 1 of the n branches that results in loading a valid w, the 0-complexities of the rest of the stages are multiplied by n.

Stage 6: Perform Quantum Walk on the Johnson Graph $J(n^b, n^c)$ where the walk is over subsets of B. We say a subset $C \in \mathcal{S}(B, n^c)$ of B is marked if and only if it contains a pair of vertices $I_G = \{u, v\}$ that belongs to the tetrahedron described above. We set the data structure $D(C)$ as $\{w\} \times C \times A \cup \{w\} \times C \times X$. Similar to the analysis in stage 4, the setup cost is $\boldsymbol{S} = |D(C)| = O(n^{c+m})$, the update cost is $\boldsymbol{U} = O(n^m)$. With parameters $n' = n^b, k' = n^c, l' = 2$, the 0-complexity of this stage is

$$O\left[\boldsymbol{S}^2 + \left(\frac{n^b}{n^c}\right)^2 \left(n^c \cdot \boldsymbol{U}^2 + \boldsymbol{C}^2\right)\right] = O\left[n^{2c+2m} + n^{2b-2c}\left(n^{c+2m} + \boldsymbol{C}^2\right)\right].$$

The checking procedure is described in the remaining 2 stages.

Stage 7: Choose a vertex z from A uniformly at random by the Or of Boolean functions learning graph. Load the vertex z into the L-vertices of V_6. The vertex z is valid if u, v, w, z form a tetrahedron for some $u, v \in B$. Since there may be at most 1 valid z in A, by lemma 16, the 0-complexity of the last stage needs to be multiplied by n^a.

Stage 8: This stage appears in the innermost loop of the quantum walks and we want to look for $\{u, v\} \in \mathcal{S}_2(C)$ such that u, v, w, z form a tetrahedron. Note that the data structure in stage 4 keeps track of $A \times \mathcal{S}_2(B)$, which contains the queried information $O_G(uvz)$. The data structure in stage 6 keep track of $\{w\} \times C \times A$, which contains $O_G(uwz), O_G(vwz)$. Thus, it remains to look for $u, v \in C$ such that $uvw \in E, uvw \in \Delta(X)$, and $z \in T_{u,v,w}$. This is equivalent to unordered search for a 3-edge of G in $\Gamma(X, C, w, z)$.

Since the data structure in stages 4 and 6 provide query information to $X \times \mathcal{S}_2(C) \cup X \times C \times \{w\}$, we can check whether $(u, v, w) \in \Delta(X)$ without queries. Stages 4 and 6 also provide query information to $\{z\} \times \mathcal{S}_2(C) \cup C \times \{z, w\}$. This allows us to check if $z \in T_{u,v,w}$ queries as well. Thus, we have full information on the set $\Gamma(X, C, w, z)$. By performing Grover search for u, v on $\Gamma(X, C, w, z)$ where all X, w, z are chosen uniformly at random. At the end of the Grover search, the L-vertices ω where $s(w)$ contains the 1-certificate $\{uvw, uvz, uwz, vwz\}$ are sinks of the learning graph $\mathcal{G}$. By lemma 32, the 0-complexity of this stage, which is the square of checking cost, is the

expected cost of Grover search learning graph $\mathcal{G}_{\Gamma(X,C,w,z)}$. By lemma 29, this value is

$$\begin{aligned}\boldsymbol{C}^2 &= \mathop{\mathbb{E}}_{\substack{X\in\mathcal{S}(n,n^x),w\in V\\ z\in C\in\mathcal{S}(n,n^c)}} \left[C_0\left(\mathcal{G}_{\Gamma(X,C,w,z)}\right) C_1\left(\mathcal{G}_{\Gamma(X,C,w,z)}\right)\right] \\ &= \mathop{\mathbb{E}}_{X,w,z} \left[|\Gamma(X,C,w,z)|\right] \leq n^{2c-x}.\end{aligned}$$

This completes the description of a learning graph $\mathcal{G}$ that computes **Tetrahedron**. Composing the complexity in each of the stages, the 0-complexity of $\mathcal{G}$ is

$$\begin{aligned}&O\left[n^{3+x}+n^{1-a}\left(n^{4b+2m}+n^{2-2b}\left(n^{3b+2m}+n\left(n^{2c+2m}+n^{2b-2c}\left(n^{c+2m}+n^{a+2c-x}\right)\right)\right)\right)\right] \\ &= O\left[n^{3+x}+n^{1-a+4b+2m}+n^{3-a+b+2m}+n^{4-a-2b+2c+2m}+n^{4-a-c+2m}+n^{4-x}\right].\end{aligned}$$

This upper bound is optimized by taking $b=3/5, c=2/5, a=x=1/5$, in which case the 0-complexity becomes $O(n^{19/5})$. Since by convention $C_1(\mathcal{G})=O(1)$, by theorem 15, the quantum query complexity of the tetrahedron finding problem is

$$Q(\textbf{Tetrahedron}) = O\left[\mathcal{LG}(\textbf{Tetrahedron})\right] = O\left(\sqrt{C_0(\mathcal{G})}\right) = O\left(n^{\frac{19}{10}}\right) = O(n^{1.9}).$$

□

4.2 Leanrning Graph for Nested Quantum Walk

The algorithm in Theorem 30 utilizes a quantum walk as a subroutine inside another quantum walk. Similarly, multiple instances of quantum walk can be nested together and we say a walk has *higher level* if it appears inside the checking procedure of another walk. In 2013, Jeffery, Kothari, and Magniez presented nested quantum walk [22]. This structure has r levels of quantum walk nested together. It uses a state tuple $(A_1, A_2, \ldots, A_r)$ where A_i is the state of the i^{th} level quantum walk. However, instead of keeping a separate data structure $D(A_i)$ at each level, it keeps track of a data structure in quantum state $|A_1, A_2, \ldots, A_r, D(A_1, \ldots, A_r)\rangle$ in the outer-most walk. This allows us to push all setup costs to the beginning of the computation.

In this section, we are trying to generalize the nested quantum walk algorithm to an adaptive learning graph. In the next section, we will utilize this framework to find a non-trivial algorithm for 4-simplex finding by searching for singletons, pairs, triples, and quadruples of vertices of a 4-simplex. Before we do that, we need to outline a few important modifications to the approach used in Theorem 30.

4.2.1 Configuration Package

Similar to lemma 17, we try to formulate the learning graph for nested quantum walks on Johnson graphs $J([n_i], k_i)$. However, we need the state space of a quantum walk dependent on the state of the lower level walks. To fix that, we label the L-vertices of our learning graph by ordered partial subsets instead of subsets. This allows us to refer to a particular element in the state by its position.

For a set X and an integer k, define the set of ordered partial subsets of size k as

$$\mathcal{P}(X,k) := \Big\{(x_1,\ldots,x_k) \in (X \cup \{\star\})^k : \text{ for } i \neq j \in [k], x_i = x_j \implies x_i = \star\Big\}. \tag{4.6}$$

We also define the set of ordered subsets of size k as

$$\mathcal{P}(X,=k) := \Big\{(x_1,\ldots,x_k) \in X^k : \text{ for } i \neq j \in [k], x_i \neq x_j\Big\}. \tag{4.7}$$

Here the $\star$ symbol is a placeholder that refers to an element of the subset not yet determined. We can treat $A \in \mathcal{P}(X,k)$ as a set by ignoring the star symbols and treat each element in A as unordered, this allows us to generalize membership check and set difference to A. We define the size of A (denoted by $|A|$) as the number of non-star elements in A. If $|A| < k$, we say A is partially filled. If A is partially filled, then for $v \notin A$, we use the notation $A \cup \{v\}$ to randomly replace a $\star$ symbol in A by v. For $A, B \in \mathcal{P}(X,k)$, we say $A \subseteq B$ if for every $i \in [k]$ where $A_i \neq \star$, we have $A_i = B_i$.

The i^{th} level of the nested walk is labeled by $\mathcal{P}([n_i], k_i)$. The certificate of the nested quantum walk is given by a sequence $I_y = (I_{y,1}, \ldots, I_{y,r})$ where each $I_{y,i}$ has a fixed size l_i. Define the set

$$\overline{I_{y,i}} := \{A'_i \in \mathcal{P}([n_i], k_i) : |A'_i| = k_i - l_i,\ A'_i \cap I_{y,i} = \emptyset\}\,. \tag{4.8}$$

We say that a state $A'_i \in \mathcal{P}([n_i], k_i)$ *avoids* the certificate $I_{y,i}$ if $|A'_i| \in \overline{I_{y,i}}$. We usually attach a prime symbol for elements of $\overline{I_{y,i}}$ and we will use these states often during the setup stages of the nested quantum walk. In the setup of the i^{th} level state A_i, we assume we have the setup states of the lower levels $A'_1, \ldots, A'_{i-1}$. So the certificate of the i^{th} level can utilize these information and we further assume that $I_{y,i} = I_{y,i,A'_1,\ldots,A'_{i-1}}$ depends on these setup states.

When we design the flow for a learning graph of quantum walk, a valid state in the i^{th} level should have the form $A'_i \cup I_{y,i}$. However, there may be special circumstances we want to avoid, even when A_i contains the certificate $I_{y,i}$. For this purpose, we define

the availability function C such that $C(A'_1, \ldots, A'_{i-1}) \subseteq \overline{I_{y,i}}$. We design learning graph such that A_i is valid if and only if $A_i = A'_i \cup I_{y,i}$ for some $A'_i \in C(A'_1, \ldots, A'_{i-1})$. In our applications, the proportion of unavailable states avoiding $I_{y,i}$ is small. That is, for some function $\alpha = o(1)$, a function of n and fixed setup states $A'_1, \ldots, A'_{i-1}$, we assume that

$$\Pr_{A'_i \in \overline{I_{y,i}}}(A'_i \notin C(A'_1, \ldots, A'_{i-1})) \leq \alpha. \tag{4.9}$$

In this case, we call $C(A'_1, \ldots, A'_{i-1})$ an *α-subset* of $\overline{I_{y,i}}$. If $C(A'_1, \ldots, A'_{i-1}) = \overline{I_{y,i}}$, we say that C is *trivial* for this level.

The data structure associated with the nested quantum walk is given by a monotone function D mapping from $\prod_{i=1}^{r} \mathcal{P}([n_i], k_i)$ to $\mathcal{P}([N])$. This data structure is kept at the lowest level of the nested quantum walk so that all levels have access to the data structure.

Finally, let's formalize these ideas by grouping all the sets and parameters defined above into a configuration package used to define the learning graph of a nested Johnson walk.

Definition 14. For each $i \in [r]$, let $0 < l_i \leq k_i = o(n_i)$ be integer parameter where l_i is a constant. Let $f_{A_1,\ldots,A_r} : \{0,1\}^N \to \{0,1\}$ be Boolean functions and suppose

$$f = \bigvee_{A_i \in \mathcal{P}([n_i],=k_i),\ i \in [r]} f_{A_1,\ldots,A_r} \tag{4.10}$$

is the function we are trying to compute. Define the *configuration* of a nested Johnson walk learning graph computing f as the tuple

$$\Big(\big\{ f_{A_1,\ldots,A_r} \big\}_{A_i \in \mathcal{P}([n_i],=k_i), i \in [r]},\ \big\{ (n_i, k_i, l_i) \big\}_{i \in [r]},\ \big\{ I_y \big\}_{y \in f^{-1}(1)},\ C\ , \alpha,\ D, \\ \big\{ \mathcal{G}_{A_1,\ldots,A_r,\lambda} \big\}_{A_i \in \mathcal{P}([n_i],=k_i),\ i \in [r],\ \lambda \text{ partial assignment on } D(A_1,\ldots,A_r)} \Big). \tag{4.11}$$

Here, r is the number of levels in the nesting structure. $\alpha(n) = o(1)$ is a function of n. The variables $\{I_y\}, C, D$ respectively denotes the sequence of certificates, the availability function, and the data structure explained in this subsection. For each λ a partial assignment on $D(A_1, \ldots, A_r)$, let $f_{A_1,\ldots,A_r,\lambda}$ be the partial Boolean function $f_{A_1,\ldots,A_r}$ restricted to inputs $z \in \{0,1\}^N$ where $z_{D(A_1,\ldots,A_r)} = \lambda$. Then, we have $f_{A_1,\ldots,A_r} = \bigvee_\lambda f_{A_1,\ldots,A_r,\lambda}$. Furthermore, each $\mathcal{G}_{A_1,\ldots,A_r,\lambda}$ is a learning graph for $f_{A_1,\ldots,A_r,\lambda}$.

The following conditions on the configuration make sure the sequence of certificates can depend on previous setup states, as we explained above.

Definition 15. We say that the configuration in equation (4.11) is *admissible* if for every $y \in f^{-1}(1)$, there is $I_{y,1} \in \mathcal{S}([n_1], l_1)$ and an α-subset $C() \subseteq \overline{I_{y,1}}$, such that for every $A'_1 \in C()$, there is $I_{y,2} \in \mathcal{S}([n_2], l_2)$ and an α-subset $C(A'_1) \subseteq \overline{I_{y,2}}$, such that for every $A'_2 \in C(A'_1)$, ..., there is $I_{y,r} \in \mathcal{S}([n_r], l_r)$ and an α-subset $C(A'_1, \ldots, A'_{r-1}) \subseteq \overline{I_{y,r}}$, such that for every $A'_r \in C(A'_1, \ldots, A'_{r-1})$, we have

$$f_{A'_1 \cup I_{y,1}, \ldots, A'_r \cup I_{y,r}}(y) = 1. \tag{4.12}$$

4.2.2 α-symmetric Stage

Before we construct the learning graph for nested Johnson walk, we investigate what happens when we drop an α-fraction of valid L-vertices from its flows.

Definition 16. Suppose $\mathcal{F}$ is a learning graph stage with starting L-vertices V_i and ending L-vertices V_j. Define $c := |V_i|$, $e = |V_j|$ and set s a constant. Let $V_{i,y}, V_{j,y}$ be the set of vertices in V_i, V_j, respectively, that receives positive flow from p_y. For $v \in V_{i,y} \cup V_{j,y}$, let $p'_{v,y}$ denote the value of the positive flow through vertex v. We say $\mathcal{F}$ is *α-symmetric with constant s* if it can be obtained via the following operations:

1. Suppose we have a symmetric stage $\mathcal{F}'$ in definition 9 with parameters c, c', d, d', e, e'. We let $\mathcal{F}$ inherit the L-vertices and L-edges of $\mathcal{F}'$. It remains to define the flow of $\mathcal{F}$.

2. For any $y \in f^{-1}(1)$, there is a set $V'_{i,y} \subseteq V_i$ of beginning vertices receiving positive flow from $p_y(\mathcal{F}')$ where $|V'_{i,y}| = c'$. The set $V_{i,y}$ is obtained by removing a small fraction of L-vertices from $V'_{i,y}$ such that $(1-\alpha)^s c' \leq |V_{i,y}| \leq c'$.

3. Let $V'_{j,y} \subseteq V_j$ be the set of ending vertices receiving positive flow from $p_y(\mathcal{F}')$ where $|V'_{j,y}| = e'$. The set $V_{j,y}$ is obtained by removing L-vertices from $V'_{j,y}$ such that

$$(1-\alpha)^{s+1} e' \leq |V_{j,y}| \leq e' \quad \text{and} \quad \frac{|v_+ \cap V_{j,y}|}{|v_+ \cap V'_{j,y}|} \geq 1 - \alpha \quad \text{for every } v \in V_{i,y} \tag{4.13}$$

   where v_+ denotes the set of out-neighbours of a vertex v. This ensures the subset to be removed from V_j doesn't target any $v \in V_{i,y}$.

4. The flows in $\mathcal{F}$ inherit the flows in $\mathcal{F}'$ with a few alternations. The flow of an L-edge is reduced to zero if the L-edge doesn't lie in $V_{i,y} \times V_{j,y}$. To compensate for the total value loss, the flows of the L-edges in $V_{i,y} \times V_{j,y}$ are scaled by a factor of at most $\frac{1}{(1-\alpha)^{s+1}}$. This ensures that for $v \in V_{i,y}, w \in V_{j,y}$, we have

$$\frac{1}{c'} \leq |p'_{v,y}| \leq \frac{1}{(1-\alpha)^s c'} \quad \text{and} \quad \frac{1}{e'} \leq |p'_{w,y}| \leq \frac{1}{(1-\alpha)^{s+1} e'}. \tag{4.14}$$

Note that if stage s of a learning graph is α-symmetric with constant s, then operation 2 holds for stage $s+1$ since operation 3 holds for stage s. Thus, we can design stage $s+1$ as an α-symmetric stage with constant $s+1$. We will design learning graphs consisting of sequential α-symmetric stages. Provided the amount of stages is finite, the constant s is irrelevant to the overall complexity of the learning graph.

The following lemma analyzes the complexity of one α-symmetric stage.

Lemma 31. *Let $\mathcal{F}$ be an α-symmetric stage of $\mathcal{G}$ with constant s and let $\mathcal{F}'$ be its underlying symmetric stage. Let $T := cd/c'd'$. For every $y \in f^{-1}(1)$, if L is the average length of the L-edges receiving positive flow in $\mathcal{F}'$, then the L-edges in $\mathcal{F}$ can be weighted so that*

$$C_0(\mathcal{F}) \leq T \cdot L^2 \quad \text{and} \quad C_1(\mathcal{F}, y) \leq (1-\alpha)^{-2(s+1)} = O(1).$$

Proof. By lemma 14, we can assign weights $w(e)$ to symmetric stage $\mathcal{F}'$ so that $\mathcal{F}'$ has 0-complexity $\leq T \cdot L^2$ and 1-complexity ≤ 1. Now if the same weight assignment is to be applied to $\mathcal{F}$, according to definition 8, the 0-complexity stays the same. The 1-complexity may differ in the following ways:

- The 1-complexity of the $\mathcal{F}$ may reduce because terms in equation (2.6) corresponding to L-edges that don't belong to $V_{i,y} \times V_{j,y}$ should be removed from the calculation. This doesn't change the 1-complexity upper bound.
- Each L-edge has its flow scaled by a factor at most $\frac{1}{(1-\alpha)^{s+1}}$, so every term in equation (2.6) is multiplied by a factor of at most $(1-\alpha)^{-2(s+1)}$.

The statement of the lemma follows immediately. □

4.2.3 Main Learning Graph Construction

Now, we are ready to construct an adaptive learning graph for an r-level nested Johnson walk in the following general-purpose lemma.

Lemma 32 (Learning Graph for Nested Johnson Walk)**.** *Let*

$$\left(\left\{f_{A_1,\ldots,A_r}\right\}, \left\{(n_i, k_i, l_i)\right\}, \left\{I_y\right\}, C, \alpha, D, \left\{\mathcal{G}_{A_1,\ldots,A_r,\lambda}\right\}\right)$$

be an admissible configuration defined in definitions 14 and 15. Let $\boldsymbol{S}, \boldsymbol{U}_1, \ldots, \boldsymbol{U}_r, \boldsymbol{C} > 0$ *be values such that for every* $x \in f^{-1}(0)$*, we have*

$$\mathop{\mathbb{E}}_{A'_i \in \mathcal{S}([n_i], k_i - l_i),\ \forall i \in [r]} |D(A'_1, \ldots, A'_r)|^2 \leq \boldsymbol{S}^2, \tag{4.15}$$

$$\mathop{\mathbb{E}}_{A_i \in \mathcal{S}([n_i], k_i)} \left[C_0(\mathcal{G}_{A_1,\ldots,A_r, x_{D(A_1,\ldots,A_r)}}, x) \cdot C_1(\mathcal{G}_{A_1,\ldots,A_r, x_{D(A_1,\ldots,A_r)}}) \right] \leq \boldsymbol{C}^2, \tag{4.16}$$

and for every $i \in [r]$ *and* $k_i - l_i \leq h < k_i$*,*

$$\mathop{\mathbb{E}}_{\substack{A_j \in \mathcal{S}([n_j], k_j),\ \forall j \in [i-1] \\ A_j \in \mathcal{S}([n_j], k_j - l_j),\ \forall j \in [i+1, r] \\ A_i \in \mathcal{S}([n_i], h),\ v \in [n_i] - A_i}} |D(A_1, \ldots, A_i \cup \{v\}, \ldots, A'_r) - D(A_1, \ldots, A_r)|^2 \leq \boldsymbol{U}_i^2. \tag{4.17}$$

Then there is a learning graph $\mathcal{G}$ *for* f *such that for every* $x \in f^{-1}(0), y \in f^{-1}(1)$*, we have* $C_1(\mathcal{G}, y) \leq 1$ *and*

$$C_0(\mathcal{G}, x) = O\left[\boldsymbol{S}^2 + \sum_{i=1}^{r} \left(\prod_{j=1}^{i} \left(\frac{n_j}{k_j} \right)^{l_j} \right) k_i \cdot \boldsymbol{U}_i^2 + \left(\prod_{i=1}^{r} \left(\frac{n_i}{k_i} \right)^{l_i} \right) \boldsymbol{C}^2 \right]. \tag{4.18}$$

Proof. We will construct a learning graph $\mathcal{G}$ computing f consisting of the setup, update, and checking stages analogous to the procedures of a nested quantum walk. $\mathcal{G}$ consists of $r + \sum_{i=1}^{r} l_i + 1$ stages. The first r stages are for setup, the last stage is for checking. The stages for update are labeled by lexicographically ordered pairs (i, h) for $i \in [r], h \in [l_i]$. All setup and update stages in $\mathcal{G}$ are α-symmetric. The labels of the L-vertices in $\mathcal{G}$ are given by $(A_1, \ldots, A_r)$ where $A_i \in \mathcal{P}([n_i], k_i)$. The root vertex is labeled by $(\emptyset, \ldots, \emptyset)$ where $\emptyset$ is represented by the tuple of stars $(\star, \ldots, \star)$.

Now, we try to define stages using definition 16. For $i \in [r]$, the i^{th} setup stage is given by loading $k_i - l_i$ elements to A_i. In this stage, we have

$$V'_{i,y} = \left\{ (A'_1, \ldots, A'_{i-1}) : A'_k \in \overline{I_{y,k}} \text{ for } k \in [i-1] \right\},$$
$$V'_{j,y} = \left\{ (A'_1, \ldots, A'_i) : A'_k \in \overline{I_{y,k}} \text{ for } k \in [i] \right\}.$$

So, $c' = \prod_{j=1}^{i-1} \binom{k_i}{l_i} \cdot P_{k_i - l_i}^{n_i - l_i}$. Counting the number of possible setup labels, the number of starting vertices is $c = \prod_{j=1}^{i-1} \binom{k_j}{l_j} \cdot P_{k_j - l_j}^{n_j}$. The beginning L-vertices have outdegree $d = \binom{k_i}{l_i} \cdot P_{k_i - l_i}^{n_i}$. We define

$$V_{i,y} := \left\{ (A'_1, \ldots, A'_{i-1}) \in V'_{i,y} : A'_k \in C(A'_1, \ldots, A'_{k-1}) \text{ for } k \in [i-1] \right\}.$$

With the setup information $A'_1, \ldots, A'_{i-1}$, the certificate $I_{y,i}$ is fixed. The L-edges receive positive flow if and only if no elements in $I_{y,i}$ are loaded to A_i and $A_i \in C(A'_1, \ldots, A'_{i-1})$. Since our configuration is admissible, we have $d' = \binom{k_i}{l_i} \cdot P^{n_i-l_i}_{k_i-l_i}$. Since $C(A'_1, \ldots, A'_{i-1})$ is an α-subset of $I_{y,i}$, equation (4.13) is satisfied with this construction. Since l_i are constants and $\alpha = o(1)$, the speciality of this stage is $cd/c'd' = O(1)$. By lemma 31, the sum of 0-complexity of these r stages is at most $O(\mathbb{E}\left[|D(A'_1, \ldots, A'_r)|\right]^2) \leq O(\boldsymbol{S}^2)$.

For $i \in [r]$ and $h \in [l_i]$, stage (i, h) consists of beginning L-vertices $(A_1, \ldots, A_r)$ where $|A_j| = k_j$ for $j \in [i-1]$, $|A_j| = k_j - l_j$ for $j \in [i+1, r]$, and $|A_i| = k_i - l_i + h - 1$. Here,

$$V'_{i,y} = \{(A_1, \ldots, A_r) : I_{y,j} \subseteq A_j \text{ for } j \in [i-1],\ A_j \cap I_{y,j} = \emptyset \text{ for } j \in [i+1, r], \\ \text{and } |I_{y,i} \cap A_i| = h - 1\}.$$

The L-edges of this stage load a new element to A_i, and an L-edge in this stage receives positive flow from p_y if and only if the new element loaded belongs to $I_{y,i}$. The corresponding parameter values for this stage are

$$c = \binom{k_j}{l_j - h + 1} \cdot P^{n_j}_{k_j-l_j+h-1} \cdot \prod_{j=1}^{i-1} P^{n_j}_{k_j} \cdot \prod_{j=i+1}^{r} \binom{k_j}{l_j} \cdot P^{n_j}_{k_j-l_j},$$
$$c' = \prod_{j=1}^{r} \binom{k_j}{l_j} \cdot P^{n_j-l_j}_{k_j-l_j}, \quad d = (l_j - h + 1)(n_i - (k_j - l_j + h - 1)),$$
$$d' = (l_j - h + 1)^2.$$

The speciality of this stage is $\dfrac{cd}{c'd'} = O\left(n_i \left(\dfrac{n_i}{k_i}\right)^{h-1} \prod_{j=1}^{i-1} \left(\dfrac{n_j}{k_j}\right)^{l_j}\right)$. By lemma 31, the 0-complexity of this stage is at most $T \cdot \mathbb{E}\left[|D(A_1, \ldots, A_i \cup \{v\}, \ldots, A_r) - D(A_1, \ldots, A_r)|\right]^2 = O\left(k_i \left(\dfrac{n_i}{k_i}\right)^{h} \prod_{j=1}^{i-1} \left(\dfrac{n_j}{k_j}\right)^{l_j} \cdot \boldsymbol{U}_i^2\right)$.

The final stage of $\mathcal{G}$ performs the checking operation. For every beginning L-vertex $(A_1, \ldots, A_r)$ where $|A_i| = k_i$ for all $i \in [r]$, attach the learning graph $\mathcal{G}_{A_1,\ldots,A_r,x_{D(A_1,\ldots,A_r)}}$ to this L-vertex and rescale the weights of the L-edges in $\mathcal{G}_{A_1,\ldots,A_r,x_{D(A_1,\ldots,A_r)}}$ by $\lambda_{A_1,\ldots,A_r} = C_1(\mathcal{G}_{A_1,\ldots,A_r,x_{D(A_1,\ldots,A_r)}}) \Big/ \prod_{i=1}^{r} P^{n_i-l_i}_{k_i-l_i} P^{k_i}_{l_i}$. There are $\prod_{i=1}^{r} P^{n_i}_{k_i}$ beginning L-vertices and more than $(1-\alpha)^r \prod_{i=1}^{r} P^{n_i-l_i}_{k_i-l_i} P^{k_i}_{l_i}$ of them receives positive flow. The values of flow in these subroutine learning graphs inherit from the values of flow in the original learning graph,

rescaled by $\Theta\left(\prod_{i=1}^{r} P_{k_i-l_i}^{n_i-l_i} P_{l_i}^{k_i}\right)^{-1}$. Our choice of rescaling ensures that the 1-complexity of this stage is

$$\sum_{\substack{A_i' \in \mathcal{S}([n_i-l_i],k_i-l_i),\forall i\in[r] \\ A_i' \in \overline{I_{y,i}},\ A_i' \cup I_{y,i} = A_i}} \frac{C_1(\mathcal{G}_{A_1,\ldots,A_r,x_{D(A_1,\ldots,A_r)}})}{\Theta\left(\prod_{i=1}^{r}\left(P_{k_i-l_i}^{n_i-l_i} P_{l_i}^{k_i}\right)^2\right)\cdot \lambda_{A_1,\ldots,A_r}} = O(1).$$

The 0-complexity of the final stage is

$$\sum_{A_i\in\mathcal{S}([n_i],k_i),\forall i\in[r]} \lambda_{A_1,\ldots,A_r} C_0(\mathcal{G}_{A_1,\ldots,A_r,x_{D(A_1,\ldots,A_r)}},x) = O\left(\prod_{i=1}^{r}\left(\frac{n_i}{k_i}\right)^{l_i}\cdot \boldsymbol{C}^2\right).$$

□

4.3 4-Simplex Finding with Nested Quantum Walk

We let $HG(n,m,r)$ denote the hypergeometric distribution function, where n is the total number of instances, m is the number of good instances, and r is the number of draws without replacement. The tail bound of this distribution is given below.

Lemma 33. *[20] Suppose $X \sim HG(n,m,r)$ with mean value $\mu = \frac{rm}{n}$, we have*

1. *for any $0 < \delta \leq 1$, $\Pr\left(X \geq (1+\delta)\mu\right) \leq \exp\left(\frac{\mu\delta^2}{3}\right)$,*
2. *for any $\delta > 2e-1$, $\Pr\left(X > (1+\delta)\mu\right) < 2^{-(1+\delta)\mu}$.*

Now, we extend the algorithm presented in [20] to simplex finding in rank-4 hypergraph.

Theorem 34. *There is an adaptive learning graph algorithm for computing the 4-simplex finding problem with $O(n^{2.4548})$ quantum queries.*

Proof. This algorithm is based on a nested quantum walk where we load all vertices of a 4-simplex first, then load the pairs of these vertices, the triples of these vertices, and the 4-hyperedges of this 4-simplex in order. This nested quantum walk utilizes 30 real parameters $0 \leq a_i, b_{ij}, c_{ijk}, d_{ijkl} < 1$ for $ijkl \in \mathcal{S}_4([5])$. For convenience of notation, we also define 15 dependent values

$$\begin{aligned} m_{ijk} &= b_{ij} + b_{ik} + b_{jk} - a_i - a_j - a_k, \\ m_{ijkl} &= c_{ijk} + c_{ijl} + c_{ikl} + c_{jkl} - b_{ij} - b_{ik} - b_{il} - b_{jk} - b_{jl} - b_{kl} + a_i + a_j + a_k + a_l. \end{aligned}$$

We say that the set of parameters $\{a_i, b_{ij}, c_{ijk}, d_{ijkl} : ijkl \in \mathcal{S}_4([5])\}$ is *admissible* if the following set of (possibly strict) linear conditions hold.

$$\begin{aligned}
b_{ij} &\leq a_i + a_j && \text{for all } ij \in \mathcal{S}_2([5]),\\
c_{ijk} &\leq m_{ijk} && \text{for all } ijk \in \mathcal{S}_3([5]),\\
d_{ijkl} &\leq m_{ijkl} && \text{for all } ijkl \in \mathcal{S}_4([5]),\\
a_i - b_{ij} &< 0 && \text{for all } ij \in \mathcal{S}_2([5]),\\
b_{ij} - m_{ijk} &< 0 && \text{for all } ijk \in \mathcal{S}_3([5]),\\
c_{ijk} - m_{ijkl} &< 0 && \text{for all } ijkl \in \mathcal{S}_4([5]),\\
-c_{ijl} - c_{ikl} + b_{il} + b_{jl} + b_{kl} - a_l &< 0 && \text{for all } ijkl \in \mathcal{S}_4([5]).
\end{aligned}$$

Suppose G is a 4-uniform hypergraph defined on vertex set V containing a 4-simplex as a sub-hypergraph. Let u_1, u_2, u_3, u_4, u_5 be the vertices of this 4-simplex. We will use lemma 32 to define an adaptive learning graph $\mathcal{G}$ that finds this 4-simplex. There are $r' = 30$ levels to this walk.

In level $i \in [5]$, we search for vertex u_i using a walk over the Johnson Graph $J(n, n^{a_i})$. Let's denote the state of this walk by $A_i \in \mathcal{P}([n], n^{a_i})$. Let $V_i = \{v_s : s \in A_i\}$, we say A_i is marked if and only if $u_i \in V_i$. In the context of lemma 32, the associated parameters of this level are $n_i = n, k_i = n^{a_i}, l_i = 1$ and $I_{y,i} = \{s : v_s = u_i\}$. The available set $C(A_1, \ldots, A_{i-1})$ is trivially defined for this level.

We label the next 10 levels by pairs of indices $ij \in \mathcal{S}_2([5])$. In level ij where $i < j$, we invoke a quantum walk over the Johnson Graph $J(n^{a_i+a_j}, n^{b_{ij}})$. Let $B_{ij} \subseteq [n^{a_i+a_j}]$ be the state of this walk and let $V_{ij} = \{v_{s_i}v_{s_j} : (s_i, s_j) \in (A_i \times A_j)[B_{ij}]\}$ be the associated pairs of vertices. We say B_{ij} is marked if it satisfies the following conditions.

1. $u_iu_j \in V_{ij}$,
2. for all $v_i \in V_i$, we have $n^{b_{ij}-a_i}/2 \leq |\{v_j \in V_j : v_iv_j \in V_{ij}\}| \leq 2n^{b_{ij}-a_i}$,
3. for all $v_j \in V_j$, we have $n^{b_{ij}-a_j}/2 \leq |\{v_i \in V_i : v_iv_j \in V_{ij}\}| \leq 2n^{b_{ij}-a_j}$,
4. whenever k is an index such that the level ik comes before ij in the nested structure, we have $|\{v_i \in V_i : v_iv_j \in V_{ij}, v_iv_k \in V_{ik}\}| \leq 11n^{m_{ijk}-b_{jk}}$ for every $v_j \in V_j, v_k \in V_k$.

Note that this is formalized in the learning graph model by taking parameters $n_{ij} = n^{a_i+a_j}, k_{ij} = n^{b_{ij}}, l_{ij} = 1$, setting $I_{y,ij} = \{s : (A_i \times A_j)[s] = (s_i, s_j), v_{s_i}v_{s_j} = u_iu_j\}$. Define $C(A_1, \ldots, B_{ij-1}) = \{B_{ij} :$ Condition 2, 3, 4 holds for $B_{ij}\}$. Here, B_{ij-1} is just a notation

referring to the state prior to B_{ij}. The following lemma is presented in [20] and shows that the fraction of B_{ij} for which condition 2, 3, or 4 doesn't hold is small.

Lemma 35. *Given marked states $A_1, \ldots, B_{ij-1}$, we have*

$$\Pr(B_{ij} \notin C(A_1, \ldots, B_{ij-1})) \leq O\left(n^{a_i} \exp\left(n^{a_i - b_{ij}}\right) + n^{a_j} \exp\left(n^{a_j - b_{ij}}\right) + n^{a_i + a_j} \exp\left(n^{b_{jk} - m_{ijk}}\right)\right).$$

We will not restate its proof but its idea is captured by the proof of lemma 36. Provided the set of parameters is admissible, we can take $\alpha(n)$ an exponentially decreasing function.

The next 10 levels are labeled by triples of indices $ijk \in \mathcal{S}_3([5])$. In level ijk where $i < j < k$, we quantum walk over the Johnson Graph $J(11n^{m_{ijk}}, n^{c_{ijk}})$. Let $C_{ijk} \subseteq [11n^{m_{ijk}}]$ be the state of this walk and define

$$\Gamma_{ijk} = \left\{(s_i, s_j, s_k) \in A_i \times A_j \times A_k : v_{s_i} v_{s_j} \in V_{ij}, v_{s_i} v_{s_k} \in V_{ik}, v_{s_j} v_{s_k} \in V_{jk}\right\}$$
$$V_{ijk} = \left\{v_{s_i} v_{s_j} v_{s_k} : (s_i, s_j, s_k) = \Gamma_{ijk}[s] \text{ for } s \in C_{ijk}, s \leq |\Gamma_{ijk}|\right\}.$$

We say C_{ijk} is marked if it satisfies the following conditions.

1. $u_i u_j u_k \in V_{ij}$,
2. for all $v_j v_k \in V_{jk}$, we have $|\{v_i \in V_i : v_i v_j v_k \in V_{ijk}\}| \leq \frac{1}{6} n^{c_{ijk} - b_{jk}}$,
3. for all $v_i v_k \in V_{ik}$, we have $|\{v_j \in V_j : v_i v_j v_k \in V_{ijk}\}| \leq \frac{1}{6} n^{c_{ijk} - b_{ik}}$,
4. for all $v_i v_j \in V_{ij}$, we have $|\{v_k \in V_k : v_i v_j v_k \in V_{ijk}\}| \leq \frac{1}{6} n^{c_{ijk} - b_{ij}}$,
5. whenever l is an index such that the levels ijl, ikl come before ijk in the nested structure, we have $|\{v_i \in V_i : v_i v_j v_k \in V_{ijk}, v_i v_j v_l \in V_{ijl}, v_i v_k v_l \in V_{ikl}\}| \leq \frac{1}{11} n^{m_{ijkl} - c_{jkl}}$ for every $v_j \in V_j, v_k \in V_k, v_l \in V_l$.

In the learning graph model, the associated parameters are $n_{ijk} = 11n^{m_{ijk}}, k_{ijk} = n^{c_{ijk}}$, $l_{ijk} = 1$. We set $I_{y,ijk} = \{s : \Gamma_{ijk}[s] = (s_i, s_j, s_k), v_{s_i} v_{s_j} v_{s_k} = u_i u_j u_k\}$ and define $C(A_1, \ldots, C_{ijk-1}) = \{C_{ijk} : \text{ Condition 2 to 5 holds for } C_{ijk}\}$. Assuming $A_1, \ldots, C_{ijk-1}$ are marked. By condition 4 of the definition of marked B_{ik}, we have

$$|\Gamma_{ijk}| = \sum_{v_j v_k \in B_{jk}} |\{v \in V_i : v v_j \in V_{ij} \text{ and } v v_k \in V_{ik}\}| \leq n^{b_{jk}} \cdot 11n^{m_{ijk} - b_{jk}} = 11n^{m_{ijk}}.$$

This ensures that the certificate $u_i u_j u_k$ will not overflow and such an index s exists for $I_{y,ijk}$. Similar to lemma 35, we have a lemma showing that the fraction of C_{ijk} for which conditions 2 to 5 don't hold is also small.

Lemma 36. *Given marked states $A_1, \ldots, C_{ijk-1}$, the value $\Pr(C_{ijk} \notin C(A_1, \ldots, C_{ijk-1}))$ is an exponentially close to 0 with respect to n, provided the set of parameters are admissible.*

To avoid interrupting the presentation of the algorithm, the proofs of this lemma and the lemmas in the rest of this subsection are presented in appendix A.

The last 5 levels are labeled by quadruples of indices $(i, j, k, l) \in \mathcal{S}_4([5])$. In level $ijkl$ where $i < j < k < l$, we invoke a quantum walk over Johnson Graph $J\left(\Theta(n^{m_{ijkl}}), n^{d_{ijkl}}\right)$. Let $D_{ijkl} \subseteq [\Theta(n^{m_{ijkl}})]$ be the state of this walk. Define

$$\begin{aligned}\Gamma_{ijkl} = \{(s_i, s_j, s_k, s_l) \in A_i \times A_j \times A_k \times A_l : v_{s_i}v_{s_j}v_{s_k} \in V_{ijk}, v_{s_i}v_{s_j}v_{s_l} \in V_{ijl},\\ v_{s_i}v_{s_k}v_{s_l} \in V_{ikl}, v_{s_j}v_{s_k}v_{s_l} \in V_{jkl}\}\\ V_{ijkl} = \left\{v_{s_i}v_{s_j}v_{s_k}v_{s_l} : (s_i, s_j, s_k, s_l) = \Gamma_{ijkl}[s] \text{ for } s \in D_{ijkl}, s \le |\Gamma_{ijkl}|\right\}.\end{aligned}$$

We say that D_{ijkl} is marked if and only if $u_iu_ju_ku_l \in V_{ijkl}$. In the learning graph, the corresponding parameters are $n_{ijkl} = \Theta(n^{m_{ijkl}}), k_{ijkl} = n^{d_{ijkl}}, l_{ijkl} = 1$. We set $I_{y,ijkl} = \{s\}$ where $\Gamma_{ijkl}[s] = (s_i, s_j, s_k, s_l)$ and $v_{s_i}v_{s_j}v_{s_k}v_{s_l} = u_iu_ju_ku_l$. Assuming $A_1, \ldots, D_{ijkl-1}$ are marked. By condition 7 of the definition of marked C_{ijl}, we have

$$\begin{aligned}|\Gamma_{ijkl}| &= \sum_{\substack{(s_j,s_k,s_l)=\Gamma_{jkl}[s]\\ s\in C_{jkl}}} \left|\left\{v \in V_i : vv_{s_j}v_{s_k} \in V_{ijk}, vv_{s_j}v_{s_l} \in V_{ijl}, vv_{s_k}v_{s_l} \in V_{ikl}\right\}\right|\\ &\le n^{c_{jkl}} \cdot \frac{1}{11}n^{m_{ijkl}-c_{jkl}} = \Theta\left(n^{m_{ijkl}}\right).\end{aligned}$$

This ensures that the certificate $u_iu_ju_ku_l$ will not overflow and such an index s exists for $I_{y,ijkl}$. $C(A_1, \ldots, D_{ijkl-1})$ is trivially defined for this level.

Let $\boldsymbol{A} = (A_1, \ldots, A_5, B_{12}, \ldots, B_{45}, C_{123}, \ldots, C_{345}, D_{1234}, \ldots, D_{2345})$ be the sequence of states, the associated data structure is given by $D(\boldsymbol{A}) := \bigcup_{ijkl\in\mathcal{S}_4([5])} V_{ijkl}$. It is important to note that a state in $\mathcal{P}([n_i], k_i)$ may only be partially filled. If any of the entries in the $A_i's, B'_{ij}s, C'_{ijk}s, D_{ijkl}$ necessary to identify a quadruple in V_{ijkl} is missing, this quadruple will not be listed in V_{ijkl}. The cost of setup is $\boldsymbol{S} \le \sum_{ijkl\in\mathcal{S}_4([5])} n^{d_{ijkl}}$. Once we have the $\mathcal{O}_G$-query information to $D(\boldsymbol{A})$, it is trivial to check if $u_1, \ldots, u_5$ form a 4-simplex. Thus, the cost of checking $\boldsymbol{C}$ is 0 and it remains to find and justify the update costs based on the size of the tuple of vertices we are loading.

In update stage $i \in [5]$, we start with beginning L-vertex $\boldsymbol{A} = (A_1, \ldots, D_{2345})$ where $|A_i| = n^{a_i} - 1$. If we are loading $s \notin A_i$ to A_i, the queries needed to update the data

structure are precisely the number of newly identifiable quadruples in V_{ijkl} due to loading s. The following lemma identifies the update costs.

Lemma 37. *Let Γ'_{ijk}, V'_{ijk} be the sets Γ_{ijk}, V_{ijk} obtained after loading a random element s to A_i, then*

$$\mathop{\mathbb{E}}_{\boldsymbol{A},s} |\Gamma'_{ijkl} - \Gamma_{ijkl}| = O(n^{m_{ijkl}-a_i}) \text{ and } \mathop{\mathbb{E}}_{\boldsymbol{A},s} |V'_{ijkl} - V_{ijkl}| = O(n^{d_{ijkl}-a_i}).$$

By the above lemma, we can conclude that

$$\begin{aligned}\boldsymbol{U}_i &= O\left(\mathop{\mathbb{E}}_{\boldsymbol{A},s} |D(\dots, A_i \cup \{s\}, \dots, D_{2345}) - D(\dots, A_i, \dots, D_{2345})|\right) \\ &= O\left(\sum_{j,k,l:ijkl\in\mathcal{S}_4([5])} \mathop{\mathbb{E}}_{\boldsymbol{A},s} |V'_{ijkl} - V_{ijkl}|\right) = O\left(\sum_{j,k,l:ijkl\in\mathcal{S}_4([5])} n^{d_{ijkl}-a_i}\right).\end{aligned}$$

In update stage $ij \in \mathcal{S}_2([5])$ where $i < j$, we start with beginning L-vertex $\boldsymbol{A} = (A_1, \dots, D_{2345})$ where $|B_{ij}| = n^{b_{ij}} - 1$. If we are loading s to B_{ij}, we are again looking for the newly identifiable quadruples in V_{ijkl} due to loading s.

Lemma 38. *Let $\Gamma'_{ijkl}, V'_{ijkl}$ be the set Γ_{ijkl}, V_{ijkl} obtained after loading index s to B_{ij}, then*

$$\mathop{\mathbb{E}}_{\boldsymbol{A},v} |\Gamma'_{ijkl} - \Gamma_{ijkl}| = O(n^{m_{ijkl}-b_{ij}}) \text{ and } \mathop{\mathbb{E}}_{\boldsymbol{A},v} |V'_{ijkl} - V_{ijkl}| = O(n^{d_{ijkl}-b_{ij}}).$$

The above lemma shows that

$$\begin{aligned}\boldsymbol{U}_{ij} &= O\left(\mathop{\mathbb{E}}_{\boldsymbol{A},v} |D(\dots, B_{ij} \cup \{t_i t_j\}, \dots) - D(\dots, B_{ij}, \dots)|\right) \\ &= O\left(\sum_{k,l:ijkl\in\mathcal{S}_4([5])} \mathop{\mathbb{E}}_{\boldsymbol{A},t_i t_j} |V'_{ijkl} - V_{ijkl}|\right) = O\left(\sum_{k,l:ijkl\in\mathcal{S}_4([5])} n^{d_{ijkl}-b_{ij}}\right).\end{aligned}$$

In update stage $ijk \in \mathcal{S}_3([5])$ where $i < j < k$, we start with begining L-vertex $\boldsymbol{A} = (A_1, \dots, D_{2345})$ where $|C_{ijk}| = n^{c_{ijk}} - 1$. If we are loading the triple $v_i v_j v_k$ to C_{ijk}, we look for newly identifiable quadruples in V_{ijkl} due to loading $v_i v_j v_k$.

Lemma 39. *Let $\Gamma'_{ijkl}, V'_{ijkl}$ be the set Γ_{ijkl}, V_{ijkl} obtained after loading a random triple $v_i v_j v_k$ to C_{ijk}, then*

$$\mathop{\mathbb{E}}_{\boldsymbol{A},v} |\Gamma'_{ijkl} - \Gamma_{ijkl}| = O(n^{m_{ijkl}-c_{ijk}}) \text{ and } \mathop{\mathbb{E}}_{\boldsymbol{A},v} |V'_{ijkl} - V_{ijkl}| = O(n^{d_{ijkl}-c_{ijk}}).$$

level s	i	ij	ijk	$ijkl$
n_s	n	$n^{a_i+a_j}$	$\Theta\left(n^{m_{ijk}}\right)$	$\Theta\left(n^{m_{ijkl}}\right)$
k_s	n^{a_i}	$n^{b_{ij}}$	$n^{c_{ijk}}$	$n^{d_{ijkl}}$
U_s	$O\left(\max_{j,k,l} n^{d_{ijkl}-a_i}\right)$	$O\left(\max_{k,l} n^{d_{ijkl}-b_{ij}}\right)$	$O\left(\max_{l} n^{d_{ijkl}-c_{ijk}}\right)$	1

Table 4.1: Parameters and quantum query complexities of update for each level of the nested quantum walk learning graph for 4-simplex finding.

The above lemma shows that

$$\begin{aligned}\boldsymbol{U}_{ijk} &= O\left(\mathop{\mathbb{E}}_{\boldsymbol{A},v} |D(\dots, C_{ijk}\cup\{s_is_js_k\},\dots) - D(\dots, C_{ijk},\dots)|\right)\\ &= O\left(\sum_{l:ijkl\in\mathcal{S}_4([5])} \mathop{\mathbb{E}}_{\boldsymbol{A},s_is_js_k} |V'_{ijkl} - V_{ijkl}|\right) = O\left(\sum_{l:ijkl\in\mathcal{S}_4([5])} n^{d_{ijkl}-c_{ijk}}\right).\end{aligned}$$

Finally, in update stage $ijkl \in \mathcal{S}_4([5])$ where $i<j<k<l$, the cost of update is at most $\boldsymbol{U}_{ijkl} = 1$. By lemma 32, the query complexity of this learning graph is

$$O\left(\boldsymbol{S} + \sum_{i=1}^{30}\left(\prod_{j=1}^{i}\sqrt{\frac{n_j}{k_j}}\right)\cdot\sqrt{k_i}\cdot\boldsymbol{U}_i\right).$$

We summarize the parameters that appear in the above complexity in table 4.1.

Optimizing a linear program involving parameters $a_i, b_{ij}, c_{ijk}, d_{ijkl}$, the optimal complexity comes down to $O(n^{2.455})$ by taking (approximate) parameter values

$a_1 = 0.30435$, $a_2 = 0.65217$, $a_3 = 0.82609$, $a_4 = 0.91304$, $a_5 = 0.95652$,
$b_{12} = 0.95652$, $b_{13} = 1.13043$, $b_{14} = 1.21739$, $b_{15} = 1.16579$, $b_{23} = 1.45059$,
$b_{24} = 1.45059$, $b_{25} = 1.54567$, $b_{34} = 1.49802$, $b_{35} = 1.64032$, $b_{45} = 1.75494$,
$c_{123} = 1.75494$, $c_{124} = 1.75494$, $c_{125} = 1.75494$, $c_{134} = 1.80237$, $c_{135} = 1.84958$,
$c_{145} = 1.87440$, $c_{234} = 1.95477$, $c_{235} = 2.04985$, $c_{245} = 2.13966$, $c_{345} = 2.07817$,
$d_{1234} = 2.25911$, $d_{1235} = 2.25911$, $d_{1245} = 2.25911$, $d_{1345} = 2.16864$, $d_{2345} = 2.13966$.

□

Chapter 5

Lower Bound of Certificate Structure

Equation (4.2) from the last chapter shows that the **r-SF** problem has a trivial quantum query lower bound $\Omega(n^{r/2})$. Unfortunately, this is also the best lower bound we know for this problem, even in the popular case $r = 2$. Despite that, we may be able to find interesting lower bound results by augmenting the triangle finding problem to make the certificate more difficult for quantum searching.

In section 5.1, we will show that a nontrivial lower bound for the triangle finding problem gives rise to a nontrivial lower bound for simplex finding problems of higher rank. In section 5.2, we will introduce certificate structure, expanding the concept of a function certificate. We will also present the simplex sum problem, which contains the simplex finding problem as a specific case. In section 5.3, we will show that when the rank is 3, the simplex sum problem is complex enough to possess a nontrivial lower bound.

5.1 Rank Reduction

Intuitively, a tetrahedron should have more structural information than a triangle. So, if the quantum query complexity of triangle finding is $\omega(n)$, then whatever property of a triangle that makes it difficult for quantum algorithms to search should also be a property of higher rank simplices. This makes the tetrahedron finding problem nontrivial for quantum algorithms. The following theorem shows that this is indeed the case.

Theorem 40. *For any rank $r \geq 2$, we have* $Q\big(\textbf{\textit{(r+1)-SF}}\big) = \Omega\big(\sqrt{n} \cdot Q(\textbf{\textit{r-SF}})\big)$

Proof. This bound is given by a randomized reduction. Consider two disjoint sets of vertices A, B where $|A| = |B| = n$. For every vertex $v \in A$, assume there is an associated r-uniform hypergraph G_v on vertex set B. Let E_v be the set of r-edges of G_v and suppose that E_v can be accessed with an oracle query to the pair $(v, e) \in A \times \mathcal{S}_r(B)$. Then the problem of finding an r-simplex in any of the G_v is equivalent to the Boolean function $OR_n \circ \textbf{r-SF}^n$. By corollary 12, the quantum query complexity of this problem is $Q(OR_n \circ \textbf{r-SF}^n) = \Theta\left(Q(OR_n)Q(\textbf{r-SF})\right) = \Theta\left(\sqrt{n} \cdot Q(\textbf{r-SF})\right)$.

Let $\textbf{r-SF}_{A,B}$ be the r-simplex finding problem on r-hypergraph G' with the promise that G' has vertex set $A \cup B$, no r-edge in G' has more than 1 vertex in A, and G'_B is a complete r-partite hypergraph with r-partition $B_1, \ldots, B_r$ of equal size. We call the r-edges with 1 vertex in A type 1 hyperedges and the r-edges in G'_B the type 2 hyperedges. Note that type 1 and type 2 hyperedges are disjoint. Furthermore, the r-partition $B_1, \ldots, B_r$ is given in the promise and therefore deciding type 2 hyperedges doesn't cost any queries.

Given an instance of the $OR_n \circ \textbf{r-SF}^n$ problem described above, we will "increase the rank" and construct an $(r+1)$-uniform hypergraph G with randomization. Let the vertex set of G be $V = A \cup B$ and define the hyperedges in G by two types $E = E_1 \cup E_2$. Let $E_1 := \{e \cup \{v\} : v \in A, e \in E_v\}$ be the set of $(r+1)$-edges of G constructed from r-edges in G_v. To construct E_2, we will uniformly randomly pick an $(r+1)$-partition $B_1, B_2, \ldots, B_{r+1}$ of B such that $|B_1| = |B_2| = \ldots |B_{r+1}| = \frac{n}{r+1}$. Then define E_2 as $K_{B_1,B_2,\ldots,B_{r+1}}$, the $(r+1)$-edges of the complete $(r+1)$-partite graph. Note that the $(r+1)$-hypergraph G we constructed is an instance of the $\textbf{(r+1)-SF}_{A,B}$ problem. This construction is depicted in figure 5.1. Moreover, if G' is an $(r+1)$-hypergraph with the promise of the $\textbf{(r+1)-SF}_{A,B}$ problem, then for every $v \in A$, we can define an r-hypergraph H_v on vertex set B such that $v_1 v_2 \ldots v_r$ is an r-edge of H_v if and only if $v v_1 v_2 \ldots v_r$ is a type 1 hyperedge of G'_v. Note that $(v, H_v)_{v \in A}$ is an instance of the $OR_n \circ \textbf{r-SF}^n$ problem and G' can only be obtained from $(v, H_v)_{v \in A}$ in the rank-increase construction.

Suppose there are vertices $v_1, v_2, \ldots, v_{r+1} \in \mathcal{S}_{r+1}(B)$ that formed an r-simplex in G_v. Then for each $i \in [r+1]$, $e_{\hat{i}} \in E_v$ and $\{v\} \cup e_{\hat{i}}$ are type 1 hyperedges of G. Let P be the event that each of these $r+1$ vertices fall in a distinct partition of B, then

$$\Pr(P) = \Pr_{B = B_1 \cup \cdots \cup B_{r+1}} \left[\exists \pi \in Sym_{r+1}, \forall i \in [r+1], v_i \in B_{\pi(i)}\right] = \prod_{i=1}^{r} \frac{r+1-i}{r+1} \cdot \frac{n}{n-i}$$

where Sym_{r+1} is the symmetric group of $r+1$ vertices. Note that $\Pr(P)$ is a constant when r is a constant. In the event of P, $v_1 v_2 \ldots v_{r+1}$ becomes a type 2 hyperedge of G. Together with the type 1 hyperedges $\{v\} \cup e_{\hat{i}}$ in G, the vertices $\{v, v_1, v_2, \ldots, v_{r+1}\}$ form an $(r+1)$-simplex of G.

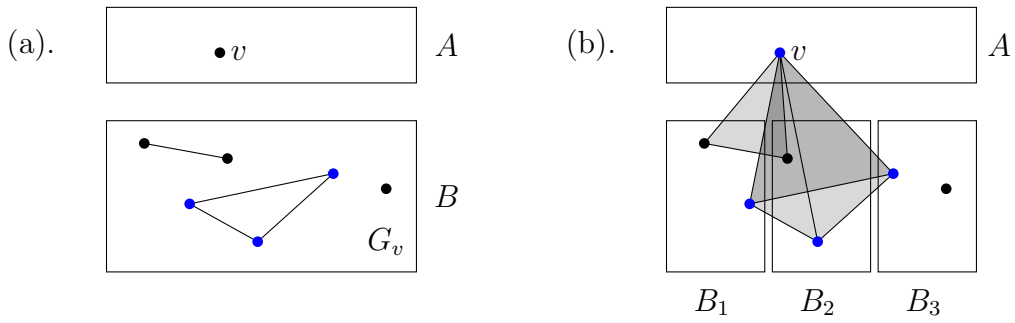


Figure 5.1: Diagram depicting an example of the rank lower bound reduction when $r = 2$. This shows that $Q\big(\textbf{Tetrahedron}\big) = \Omega\big(\sqrt{n} \cdot Q(\textbf{Triangle})\big)$. In particular, any nontrivial lower bound of the triangle finding problem implies a nontrivial lower bound for the tetrahedron finding problem. (a). depicts an instance of the $OR_n \circ \textbf{Triangle}^n$ problem, G_v is shown for a particular $v \in A$. The blue vertices form a triangle. (b). depicts an instance of the $\textbf{3-SF}_{A,B}$ problem obtained from (a). by the rank increase construction. The gray-shaded triangles are the type 1 3-hyperedges. The 3-partition of B is randomly chosen and forms a complete 3-hypergraph, so the blue vertices form a tetrahedron.

Suppose G' is an instance of $\textbf{(r+1)-SF}_{A,B}$ where G' is obtained from $(v, H_v)_{v\in A}$ in the rank-increase construction. If $u, u_1, \ldots, u_{r+1} \in A \times \mathcal{S}_{r+1}(B)$ is a set of vertices that formed an $(r+1)$-simplex in G', then $u_1, \ldots, u_{r+1}$ must be an r-simplex in H_u. Moreover, every type 1 hyperedge query in $\textbf{(r+1)-SF}_{A,B}$ is equivalent to a query of the form (u, e) in $OR_n \circ \textbf{r-SF}^n$. Therefore, we can solve the $OR_n \circ \textbf{r-SF}^n$ problem by solving an $\textbf{(r+1)-SF}_{A,B}$ problem in the same amount of quantum queries. This randomized reduction success with probability at least $\Pr(P) = \Theta(1)$, so

$$Q\big(\textbf{(r+1)-SF}\big) = \Omega\Big[Q\big(\textbf{(r+1)-SF}_{A,B}\big)\Big] = \Omega\Big[Q(OR_n \circ \textbf{r-SF}^n)\Big] = \Omega\big(\sqrt{n} \cdot Q(\textbf{r-SF})\big).$$

□

Taking the converse of the above theorem, we know that an upper bound of a simplex finding problem can be used to show an upper bound on lower-rank simplex finding problems. In particular,

Corollary 41. *If there is a quantum algorithm solving* ***Tetrahedron*** *with $o(n^{1.75})$ queries, then there is a quantum algorithm solving* ***Triangle*** *with $o(n^{1.25})$ queries.*

This provides a way to look for better triangle finding algorithms. However, finding a good upper bound of the tetrahedron problem remains equally, if not more, challenging.

5.2 Certificate Structure

In section 2.5, we see that a learning graph of f translates to a quantum query algorithm of f. But in general, it doesn't provide a corresponding lower bound. However, the learning graph complexity and the quantum query complexity are asymptotically tight for classes of problems given by a certificate structure.

Definition 17. A *certificate structure* $\mathcal{C}$ *with* N *variables* is a collection of subsets $M \subseteq 2^{[N]}$ with the superset closure property:

- If $S \in M$ and $S \subseteq T \subseteq [N]$, then $T \in M$.

Let $f : \mathcal{D} \to \{0,1\}$ be a function where $\mathcal{D} \subseteq [q]^N, q \in \mathbb{N}$. We say f *possesses* certificate structure $\mathcal{C}$ if

- for every 1-input $y \in f^{-1}(1)$, there exists an $M \in \mathcal{C}$ such that for every $S \in M$, y_S is a 1-certificate of f.
- for every $M \in \mathcal{C}$, there is a 1-input $y \in f^{-1}(1)$ such that y_S is a 1-certificate of f for every $S \in M$.

We call such M a *certificate set* of y in f.

We usually don't think of $\mathcal{C}$ as a collection of certificate sets but as the collection of functions having $\mathcal{C}$ as a certificate structure. We are thus interested in the general complexity of solving this collection of functions. let's define the complexity measure of $\mathcal{C}$ in two ways.

Definition 18. Let $\mathcal{C}$ be a certificate structure. The *quantum query complexity of* $\mathcal{C}$ (denoted by $Q(\mathcal{C})$) is the maximum quantum query complexity of a function f possessing the certificate structure $\mathcal{C}$. The *learning graph complexity of* $\mathcal{C}$ (denoted by $\mathcal{LG}(\mathcal{C})$) is the minimum complexity of a non-adaptive learning graph computing any function f possessing the certificate structure $\mathcal{C}$.

Since every learning graph of f gives rise to a quantum query algorithm computing f. We see that

$$Q(\mathcal{C}) = \max_{f:f \text{ possesses } \mathcal{C}} Q(f) \leq \max_{f:f \text{ possesses } \mathcal{C}} \mathcal{LG}(f) \leq \mathcal{LG}(\mathcal{C}). \tag{5.1}$$

The reverse direction of equation (5.1) also holds asymptotically.

Theorem 42. *[11] Let $\mathcal{C}$ be a certificate structure, we have $Q(\mathcal{C}) = \Theta(\mathcal{LG}(\mathcal{C}))$.*

It's important to note that this theorem doesn't imply $Q(f) = \Theta(\mathcal{LG}(f))$ for every f possessing $\mathcal{C}$. For such an equality to hold for a function f possessing $\mathcal{C}$, we need f to be "complete" over the class of functions possessing $\mathcal{C}$, in the sense that f is a problem general enough to have $Q(f) = Q(\mathcal{C})$. We will provide an example of the r-simplex certificate structure defined below.

5.2.1 r-simplex Certificate Structure

Definition 19. Let there be n vertices and $N = \binom{n}{r}$ potential hyperedges. The *r-simplex certificate structure* $\mathcal{C}$ has the potential hyperedges as variables. The certificate sets $M \in \mathcal{C}$ are defined by an $(r+1)$-tuple of vertices $A_M = \{a_1, a_2, \ldots, a_{r+1}\}$, where $S \in M$ iff $S \supseteq A_M$. Since there are $\binom{n}{r+1}$ many such tuples, there are exactly $\binom{n}{r+1}$ certificate sets in $\mathcal{C}$.

Note that the r-simplex finding problem has the r-simplex certificate structure. However, it is not considered the hardest problem within this certificate structure. For the rest of the section and the next section, we argue that the following hypergraph search problem is the hardest.

r-Simplex Sum Problem (r-SSum):

> Let G be a complete graph on vertex set V, $|V| = n$. Given a finite (additive) abelian group A of size q (q may depend on n) and a specified value $t \in A$. Suppose the edges of G are weighted by elements of A, where the weight function $w : \mathcal{S}_r(V) \to A$ is given as the oracle input. Find $r+1$ vertices $a_0, a_1, \ldots, a_r \in \mathcal{S}_{r+1}(V)$ such that $w(e_{\hat{0}}) + w(e_{\hat{1}}) + \cdots + w(e_{\hat{r}}) = t$ or decide that no such vertices exist.

Note that the 1-certificates of an r-simplex sum problem are given by $r+1$ vertices of G, so the r-simplex sum problem possesses the r-simplex certificate structure. Moreover, with the promise that weights of the hyperedges being 0 or 1 and setting the target t to $r+1$, the r-simplex sum problem becomes the r-simplex finding problem. Thus, **r-SSum** contains **r-SF** as a special case.

Theorem 43. *Let $\mathcal{C}$ be the r-simplex certificate structure. The quantum query complexity of the r-simplex sum problem is lower bounded by the learning graph complexity of $\mathcal{C}$, i.e. $Q(\textbf{\textit{r-SSum}}) = \Omega(\mathcal{LG}(\mathcal{C}))$. By equation 5.1, we conclude that $Q(\textbf{\textit{r-SSum}}) = \Theta(\mathcal{LG}(\mathcal{C}))$.*

Note that given an instance G with target value $t \in A$ of the r-simplex sum problem, the vertices $A_M = \{a_0, a_1, \ldots, a_r\} \in \mathcal{S}_{r+1}(V)$ form a 1-certificate of G if and only if the $(r+1)$-tuple $(w(e_{\hat{0}}), w(e_{\hat{1}}), \ldots, w(e_{\hat{r}}))$ lies in the set $T_M = \{(x_0, x_1, \ldots, x_r) \in A^{r+1} : x_0 + x_1 + \cdots + x_r = t\}$. The set T_M has a special structure, which we will define below.

Definition 20. For $k \geq 1$, $T \subseteq [q]^k$ is an *orthogonal array* if for every index $i \in [n]$ and sequence $x_1, \ldots, \hat{x_i}, \ldots, x_k \in [q]^{k-1}$, there are exactly $\dfrac{|T|}{q^{k-1}}$ many $x_i \in [q]$ such that $x_1, \ldots, x_i, \ldots, x_k \in T$. We call $|T|$ the size of the array and k the length of the array.

It is easy to see that each $T_M \subseteq A^{r+1}$ is an orthogonal array with size q^r since for any $x_1, \ldots, x_r \in A^r$, there is exactly one $x_0 \in A$ such that $x_0 + x_1 + \cdots + x_r = t$. We also note that every certificate set of the r-simplex certificate structure is uniquely defined by the unique smallest set A_M. There is a name for this type of certificate structure:

Definition 21. A certificate structure $\mathcal{C}$ is *boundedly generated* if every certificate set $M \in \mathcal{C}$ has a generating element $A_M \in M$ such that $|A_M| = O(1)$ and $S \in M$ if and only if $A_M \subseteq S$.

The following theorem completes the proof of theorem 43.

Theorem 44. *Let $\mathcal{C}$ be a boundedly generated certificate structure with N variables. Assume the alphabet of the functions possessing $\mathcal{C}$ is $[q]$ where $q \geq 2|\mathcal{C}|$. For each $M \in \mathcal{C}$, assume the generating set A_M of M is equipped with an orthogonal array $T_M \subseteq [q]^{r+1}$ of size q^r. Let $f : [q]^N \to \{0,1\}$ be the function given by $f(x) = 1$ if and only if there is an $M \in C$ such that $x_{A_M} \in T_M$. Then, $Q(f) = \Omega(\mathcal{LG}(f))$.*

We refer the proof of the above theorem to [11].

5.2.2 Dual of a Learning Graph for Certificate Structures

When $r = 2, 3$, we call the r-simplex certificate structure the triangle certificate structure, tetrahedron certificate structure respectively. Let $\mathcal{C}$ be a certificate structure. We claim

that the learning graph complexity of $\mathcal{C}$ is given by the following optimization problem.

$$\inf_{w_e, p_e(M)} \quad \sqrt{\sum_{e \in \mathcal{E}} w_e} \qquad \text{subject to} \tag{5.2a}$$

$$\sum_{e \in \mathcal{E}} \frac{p_e(M)^2}{w_e} \leq 1 \qquad \text{for every } M \in \mathcal{C} \tag{5.2b}$$

$$\sum_{\substack{e=\overrightarrow{uv} \in \mathcal{E} \\ s(u)=S}} p_e(M) = \sum_{\substack{e=\overrightarrow{uv} \in \mathcal{E} \\ s(v)=S}} p_e(M) \qquad \text{for every } M \in \mathcal{C} \text{ and } S \neq \emptyset, S \notin M \tag{5.2c}$$

$$\sum_{\substack{e=\overrightarrow{uv} \in \mathcal{E} \\ s(u)=\emptyset}} p_e(M) = 1 \qquad \text{for every } M \in \mathcal{C} \tag{5.2d}$$

$$w_e > 0 \tag{5.2e}$$

By definition 8, if we increase all the weights of the learning graph by a positive constant α, the negative complexity is multiplied by a factor of α, and the positive complexity is multiplied by a factor of $1/\alpha$. Therefore, we can transfer a positive constant between the negative and positive complexity of the learning graph without changing the overall learning graph complexity. Thus, we try to maximize the negative complexity in (5.2a) while upper-bounding the positive complexity in (5.2b) by a constant.

The constraints (5.2c) - (5.2e) encode the definition of a learning graph. Note that the length of all L-edges is assumed to be 1, this is equivalent to a learning graph with arbitrary length due to the remark of section 2.5. Furthermore, it assumes all L-edges exist between L-vertices S and $S \cup j$, (where $j \notin S, S \notin M$ for every $M \in \mathcal{C}$). If we want to remove an L-edge e to form a potentially better learning graph, we can take the weight and the flow on the e to be 0. This is justified by setting $p_e(M) = 0$ for all M and taking an infimum where $w_e \to 0^+$ of the objective function (5.2a).

Note that (5.2b) is a convex constraint. By convex duality, the optimal value of (5.2a) is the same as the optimal dual value given by:

$$\sup_{\alpha_S(M)} \quad \sqrt{\sum_{M \in \mathcal{C}} \alpha_\emptyset(M)^2} \qquad \text{subject to} \tag{5.3a}$$

$$\sum_{M \in \mathcal{C}} (\alpha_{s(u)}(M) - \alpha_{s(v)}(M))^2 \leq 1 \qquad \text{for every } e = \overrightarrow{uv} \in \mathcal{E} \tag{5.3b}$$

$$\alpha_S(M) = 0 \qquad \text{for every } S \in M \tag{5.3c}$$

Since the optimal value (5.3a) equals $\mathcal{LG}(\mathcal{C})$, to show a lower bound on the query complexity of $\mathcal{C}$, it suffices to generate a feasible solution of (5.3b) - (5.3c). In [11], Belovs and Rosmanis used this technique to show a nontrivial $\tilde{\Omega}(n^{9/7})$ lower bound for the learning graph complexity of triangle certificate structure. In the next section, we will see how the nontrivial lower bound can generalize when the rank is 1 higher.

5.3 Tetrahedron Sum Problem

Taking $r = 3$, the trivial lower bound of the tetrahedron sum problem is $Q(\textbf{3-SSum}) = \Omega(n^{1.5})$. In this section, we show that this lower bound can be improved by getting a nontrivial lower bound of the tetrahedron certificate structure. The following main theorem generalizes the result in [11] with a slightly modified proof.

Theorem 45. *The learning graph complexity of the tetrahedron certificate structure is* $\Omega(n^{12/7}/\sqrt{\log n}) = \tilde{\Omega}(n^{1.7142})$.

Proof. Let $\mathcal{C}$ be the tetrahedron certificate structure, $[n]$ be the vertex set, and $E = \mathcal{S}_3([n])$ be the set of potential hyperedges. The size of E is $N = \binom{n}{3}$. We are trying to define values $\alpha_S(M)$ that is feasible to equations (5.3b) - (5.3c).

Given $S \subseteq E$, and $j \in E - S$, let $F(S, j) \subseteq \mathcal{C}$ be the collection of certificate sets M where $S \notin M$ but $S \cup \{j\} \in M$. Let $a, b \geq 0$ be real parameters we will optimize later. We will later define a nonnegative function $g(S, M)$ so that $g(\emptyset, M) = 0$, and $g(S, M) \leq n^{-a}$ for all S and M.

Let $M \in \mathcal{C}$, define $\alpha_S(M) := 0$ if $S \in M$. For $S \notin M$, define

$$\alpha_S(M) := \max\left\{n^{-a} - n^{-2}|S| - g(S, M), 0\right\}. \tag{5.4}$$

Note that constraint (5.3c) is automatically satisfied, and we want to show that

$$\sum_{M \in \mathcal{C}} (\alpha_S(M) - \alpha_{S \cup \{j\}}(M))^2 = O(\log(n)) \tag{5.5}$$

so that if we divide each $\alpha_S(M)$ by a factor of $\sqrt{\log(n)}$, then constraint (5.3b) will hold and the objective (5.3a) equals to $\sqrt{\dfrac{\binom{n}{4} n^{-2a}}{\log(n)}} = \Theta\left(\dfrac{n^{2-a}}{\sqrt{\log(n)}}\right)$.

First, note that if $|S| > n^{2-a}$, then $\alpha_S(M) = 0$ for any $M \in \mathcal{C}$. So we will assume $|S| = O(n^{2-a})$. For any $S \subseteq E$, and $j \notin S$, we want to show that

$$\sum_{M \in \mathcal{C} - F(S,j)} [g(S \cup \{j\}, M) - g(S, M)]^2 = O(\log(n)), \tag{5.6}$$

$$\sum_{M \in F(S,j)} \left[n^{-a} - g(S, M)\right]^2 = O(\log(n)). \tag{5.7}$$

This gives us

$$\begin{aligned}
&\sum_{M \in \mathcal{C}} \left(\alpha_S(M) - \alpha_{S \cup \{j\}}(M)\right)^2 \\
&\leq \sum_{M \in \mathcal{C} - F(S,j)} \left(\alpha_S(M) - \alpha_{S \cup \{j\}}(M)\right)^2 + \sum_{M \in F(S,j)} \alpha_S(M)^2 \\
&\leq \sum_{M \in \mathcal{C} - F(S,j)} \left[n^{-2} + g(S \cup \{j\}, M) - g(S, M)\right]^2 + \sum_{M \in F(S,j)} \left[n^{-a} - g(S, M)\right]^2 \\
&\leq O\left(\binom{n}{4} \left(n^{-2}\right)^2 + \sum_{M \in \mathcal{C} - F(S,j)} [g(S \cup \{j\}, M) - g(S, M)]^2\right) + O(\log(n)) \\
&\leq O(\log(n)) + O(\log(n)) = O(\log(n)).
\end{aligned}$$

It remains to define g and verify equations (5.6) and (5.7). Let $\mu(x)$ be the median of $0, x$, and 1, and define

$$T_{S,u,v,w} := \{z \in V : uvz, vwz, uwz \in S\}. \tag{5.8}$$

Define

$$\nu(S, u, v, w, z) := \sum_{y \in T_{S,v,w,z}} \left[\frac{\min\left(\deg_S(u), \deg_S(y)\right)}{\max\left(\deg_S(u), \deg_S(y)\right)}\right]^b, \tag{5.9}$$

$$g(S, M) := \max_{u \in A_M} g(S, M, u), \quad g(S, M, u) := n^{-a}\mu\left(\frac{\nu(S, u, v, w, z)}{n^{2a}} - 1\right). \tag{5.10}$$

where v, w, z are the three vertices in A_M other than u. When u, v, w, z are clear from context, we use $\Delta_S(y)$ to abbreviate the term $\left[\frac{\min\left(\deg_S(u), \deg_S(y)\right)}{\max\left(\deg_S(u), \deg_S(y)\right)}\right]^b$.

For some $l \approx \log_2(n)$, let $I_i := [2^i, 2^{i+1}]$ for $i \in [l]$ be the critical intervals the degree of a vertex may lie in. Let $M \in F(S, j)$. By putting the maximum inside the nondecreasing

function μ, we have $n^{-a} - g(S,M) \le n^{-a}\left[1-\mu\left(\max_{u\in A_M}\frac{\nu(S,u,v,w,z)}{n^{2a}}-1\right)\right]$. The latter expression is at most n^{-a}, and this value equals to 0 when $n^{-2a}\max_{u\in A_M}\nu(S,u,v,w,z)\ge 2$. In this case, we want to count the number of certificate sets $M\in F(S,j)$ such that $A_M=\{u,v,w,z\}$, $u\in T_{S,v,w,z}$, $j=vwz$, and $\nu(S,u,v,w,z)<2n^{2a}$.

Suppose that $j=vwz$, we need to pick $u\in T_{S,v,w,z}$ such that $\nu(S,u,v,w,z)<2n^{2a}$. Note that u has degree at least 3. For a particular $i\in[l]$, there cannot be more than $O(n^{2a})$ many u to be picked such that $\deg_S(u)\in I_i$. Otherwise, the ratio of the degree of any pairs of such u is at least $1/2$ because their degrees lie in the same interval I_i. So

$$\nu(S,u,v,w,z)\ge \sum_{y\in T_{S,v,w,z},\ \deg_S(y)\in I_i}\Delta_S(y)\ge \frac{1}{2^b}\omega(n^{2a}) \tag{5.11}$$

for any u that can be picked, a contradiction. Thus, ranging over all $i\in[l]$, there can be at most $O(n^{2a}\log(n))$ many such u to be picked. So

$$\sum_{M\in F(S,j)}(n^{-a}-g(S,M))^2\le O(n^{2a}\log(n))(n^{-a})^2=O(\log(n)). \tag{5.12}$$

We have verified equation (5.7), and we will also similarly use variables u,v,w,z to count the number of M in later analyzes.

To verify equation (5.6), let $M\in\mathcal{C}-F(S,j)$. Let $u'=\arg\max_{z\in A_M} g(S,M,z)$ and $u=\arg\max_{z\in A_M} g(S\cup\{j\},M,z)$, then $g(S\cup\{j\},M)-g(S,M)=g(S\cup\{j\},M,u)-g(S,M,u')\le g(S\cup\{j\},M,u)-g(S,M,u)$. Thus, it suffices to fix $u\in A_M$ and verify equation (5.6) with $g(S,M)$ replaced by $g(S,M,u)$. We analyze how g will change by the following (possibly overlapping) cases.

- Case 1. ν increases because $j=yu_1u_2$ is incident to a vertex $y\in T_{S,v,w,z}$. Let $h\in[l]$ be the index where $\deg_S(y)\in I_h$. Note that

  $$|g(S,M,u)-g(S\cup\{j\},M,u)|\le n^{-a}n^{-2a}|\nu(S,u,v,w,z)-\nu(S\cup\{j\},u,v,w,z)|.$$

  If $\deg_S(y)\ge\deg_S(u)$, then

  $$\begin{aligned}|\Delta_S(y)-\Delta_{S\cup\{j\}}(y)| &= \frac{\deg_S(u)^b}{(\deg_S(y)+1)^b}O\left(\frac{b}{\deg_S(y)}+O\left(\frac{1}{\deg_S(y)^2}\right)\right)\\ &= \frac{1}{\deg_S(y)}=O\left(\frac{1}{2^h}\right).\end{aligned}$$

If $\deg_S(y) < \deg_S(u)$, then

$$|\Delta_S(y) - \Delta_{S\cup\{j\}}(y)| = \frac{4\deg_S(y)^{b-1} + O\left(\deg_S(y)^{b-2}\right)}{\deg_S(u)^b} = O\left(\frac{1}{\deg_S(y)}\right) = O\left(\frac{1}{2^h}\right).$$

The $M \in \mathcal{C} - F(S,j)$ that has nonzero change in g by this cause is given by choosing a vertex of j to be y, picking v, w, z from the neighbours of y such that vw, wz, vz are adjacent to y. Since $\deg_S(y) \le 2^{h+1}$, there are at most $O(2^{3h/2})$ many such triples. So, there are at most $n2^{3h/2}$ many such M and

$$\sum_{M\in\mathcal{C}-F(S,j)} (g(S\cup\{j\},M,u) - g(S,M,u))^2 \le \sum_{\substack{y \text{ incident to } j \\ h=\deg_S(y)}} n2^{3h/2} O\left(\frac{n^{-3a}}{2^h}\right)^2 = O(1).$$

as long as $a \ge \frac{1}{6}$.

- Case 2. $\deg_S(u)$ may increase by adding the edge $j = uu_1u_2$. Let $i \in [l]$ be the index where $\deg_S(u) \in I_i$. Note that $|g(S,M,u) - g(S\cup\{j\},M,u)|$ is nonzero only if $n^{2a} \le \nu(S,u,v,w,z) \le 2n^{2a}$. For each $y \in T_{S,v,w,z}$, if $\deg_S(y) > \deg_S(u)$, then

$$\begin{aligned}|\Delta_S(y) - \Delta_{S\cup\{j\}}(y)| &= \frac{b\deg_S(u)^{b-1} + O\left(\deg_S(u)^{b-2}\right)}{\deg_S(y)^b} \\ &= O\left(\frac{\deg_S(u)^{b-1}}{\deg_S(y)^b}\right) = O\left(\frac{\Delta_S(y)}{\deg_S(u)}\right).\end{aligned}$$

If $\deg_S(y) \le \deg_S(u)$, then

$$\begin{aligned}|\Delta_S(y) - \Delta_{S\cup\{j\}}(y)| &= \frac{\deg_S(y)^b}{(\deg_S(u)+1)^b}\left(\frac{b}{\deg_S(u)} + O\left(\frac{1}{\deg_S(u)^2}\right)\right) \\ &= O\left(\frac{\deg_S(y)^b}{\deg_S(u)^b}\cdot\frac{1}{\deg_S(u)}\right) = O\left(\frac{\Delta_S(y)}{\deg_S(u)}\right).\end{aligned}$$

Therefore,

$$\begin{aligned}&|g(S,M,u) - g(S\cup\{j\},M,u)| \le n^{-a}n^{-2a}|\nu(S,u,v,w,z) - \nu(S\cup\{j\},u,v,w,z)| \\ &\le n^{-3a}\sum_{y\in T_{S,v,w,z}} |\Delta_S(y) - \Delta_{S\cup\{j\}}(y)| = O\left(n^{-3a}\frac{1}{\deg_S(u)}\sum_{y\in T_{S,v,w,z}}\Delta_S(y)\right) \\ &= O\left(\frac{n^{-3a}\nu(S,u,v,w,z)}{\deg_S(u)}\right) = O\left(\frac{n^{-a}}{2^i}\right).\end{aligned}$$

The certificate sets $M \in \mathcal{C} - F(S,j)$ with nonzero changes in g are given by picking u from a vertex of j. Let $h \in [l]$, recall that we are assuming $|S| \le n^{2-a}$, so there are at most $O(n^{2-a}/2^h)$ many choices of y with degree in I_h. Moreover, since $n^{2a} \le \nu(S,u,v,w,z) \le 2n^{2a}$, there can be at most $O(2^{b|i-h|}n^{2a})$ many y with its degree in I_h. So, there are at most $\min\left(\frac{n^{2-a}}{2^h}, O(2^{b|i-h|}n^{2a})\right) \cdot (2^{h+1})^{3/2}$ many quadruples (y,v,w,z) such that $y \in T_{S,v,w,z}$, $\deg_S(y) \in I_h$. However, for every triples v,w,z where $\nu(S,u,v,w,z) = \Theta(n^{2a})$, there must be an index $h' = h(v,w,z) \in [l]$ such that v,w,z uses at least $\frac{2^{b|i-h'|}n^{2a}}{\log(n)}$ of these quadruples (y,v,w,z) with $\deg_S(y) \in I_{h'}$. Given this particular h', there can be at most $O\left(\min\left(\frac{n^{2-3a}2^{h'/2}}{2^{b|i-h'|}}, 2^{3h'/2}\right)\log(n)\right)$ choices of v,w,z to be picked. Therefore, for a particular $h \in [l]$,

$$\begin{aligned}
&\sum_{\substack{M \in \mathcal{C}-F(S,j) \\ h(v,w,z)=h}} (g(S\cup\{j\},M,u) - g(S,M,u))^2 \\
&\le O\left[\log(n)\min\left(\frac{n^{2-3a}2^{h/2}}{2^{b|i-h|}}, 2^{3h/2}\right)\left(\frac{n^{-a}}{2^i}\right)^2\right] \qquad (5.13)\\
&= O\left[\log(n)\min\left(\frac{n^{2-5a}2^{h/2}}{2^{b|i-h|}2^{2i}}, \frac{2^{3h/2}}{2^{2i}n^{2a}}\right)\right].
\end{aligned}$$

Taking $a = 2/7, b = 4$. If $2^{2(h-i)} \le n^{4/7}/\log(n)$, by the second term of the minimum, we have equation (5.13) $\le \log(n)2^{2(h-i)}n^{-4/7} = O(1)$. Otherwise, we have $2^{2(h-i)} > n^{4/7}/\log(n)$, $h > i$, and $h > \frac{2}{7}\log(n) - \frac{1}{2}\log\log(n)$. So, by the first term of the minimum, equation (5.13) $\le \frac{n^{4/7}2^h\log(n)}{2^{4(h-i)}2^{2i}} = \frac{n^{4/7}\log(n)}{2^{3h-2i}} = \frac{\log^2(n)}{2^h} = o(1)$. Thus, summing over all possible $h \in [l]$, we have

$$\sum_{M \in \mathcal{C}-F(S,j)} (g_i(S\cup\{j\},M,u) - g_i(S,M,u))^2 = O(\log(n)).$$

- Case 3. ν increases because adding $j = vwy$ increases the set $T_{S,v,w,z}$. Let $i,h \in [l]$ be indices such that $\deg_S(y) \in I_h$ and $\deg_S(u) \in I_i$. Note that g changes by this cause only if $ywz, yvz \in S$, and $n^{2a} \le \nu(S,u,v,w,z) \le 2n^{2a}$. Moreover, $g(S\cup\{j\},M,u) - g(S,M,u) \le n^{-a}n^{-2a}\Delta_{S\cup\{j\}}(y) = n^{-3a}2^{-b|i-h|}$. The $M \in \mathcal{C} - F(S,j)$ that satisfy these conditions are given by choosing an endpoint of j to be y, the other endpoints

are v, w, pick z from the $\leq 2^{h+1}$ neighbours of y. Recall that we are always assuming $|S| \leq n^{2-a}$, so there are at most $O(n^{2-a}/2^i)$ many choices of u such that $\deg_S(u) \in I_i$. Thus there are $2^{h+1}O(n^{2-a}/2^i) = O(n^{2-a}2^{h-i})$ many such M and

$$\sum_{i\in[l]} \sum_{\substack{M\in\mathcal{C}-F(S,j)\\ \deg_S(u)\in I_i}} (g_i(S\cup\{j\}, M, u) - g_i(S, M, u))^2$$
$$\leq \sum_{i\in[l]} O\left(n^{2-a}2^{h-i}\right)\left(n^{-3a}2^{-b|i-h|}\right)^2 \leq \sum_{i\in[l]} 2^{-(2b-1)|i-h|} = O(\log(n))$$

provided that $a \geq \frac{2}{7}$.

Note that adding j may be an instance of more than one case above, but the changes add up only a constant number of times in the worst case and the total change to g is still $O(1)$. By taking parameters $a = 2/7$ and $b = 4$, we conclude that equation (5.6) holds. Thus equation (5.3a) equals to $\Theta\left(\frac{n^{2-a}}{\sqrt{\log(n)}}\right) = \tilde{\Theta}\left(n^{12/7}\right)$, which is a lower bound on the learning graph complexity of $\mathcal{C}$. □

Let f denote the **Tetrahedron** problem and suppose $\mathcal{G}$ is a non-adaptive learning graph for f. Note that for every M in the tetrahedron certificate structure $\mathcal{C}$, the hypergraph G_M where $E(G_M) = \mathcal{S}_3(A_M)$ is a 1-input of f. From definition 7, the sinks of a flow p_{G_M} in $\mathcal{G}$ are the L-vertices that contain a 1-certificate of G_M. Since A_M is the unique minimal 1-certificate of G_M, the sinks of p_{G_M} in $\mathcal{G}$ are precisely the L-vertices labeled by an element $S \in M$. Thus $C_1(\mathcal{G}) = \max_{M\in\mathcal{C}} C_{1,G_M}(\mathcal{G})$. Moreover, $\mathcal{G}$ is a learning graph for every function g that possesses $\mathcal{C}$ as a certificate structure using only flows from $\{p_{G_M}(\mathcal{G}) : M \in \mathcal{C}\}$. This is because we can define the flow of $x \in g^{-1}(1)$ as the flow of $y \in f^{-1}(1)$, for some y with the same certificate set M as x. Therefore any learning graph lower bound of a function g that possesses $\mathcal{C}$ is a learning graph lower bound for the tetrahedron finding problem. By theorem 45, we get an interesting corollary.

Corollary 46. *Any non-adaptive learning graph algorithm for the tetrahedron finding problem has quantum query complexity* $\Omega(n^{12/7}/\sqrt{\log n}) = \tilde{\Omega}(n^{1.7142})$.

Chapter 6

Conclusion and Open Problems

In this thesis, we worked to develop new quantum query upper and lower bounds for multiple hypergraph problems using existing techniques. We started by generalizing the connectivity and cyclicity graph properties to hypergraph, using the fact that these problems come down to quantum searching for an $O(n)$ 1-certificate. We showed the tight $\Theta\left(n^{\frac{r+1}{2}}\right)$ query complexity bound for both problems using Ambainis's adversary bound and iterated Grover search.

Then, we focused on using an adaptive learning graph to find nontrivial quantum query algorithms for the r-simplex finding problem for $r = 3, 4$. As the rank r increases, we need more complicated algorithms for a query upper bound whose exponent of n is only slightly different from that of the trivial upper bound. So far, there are no nontrivial algorithms for r-simplex finding with $r \geq 5$. However, the nested quantum walk approach from section 4.3 provides a method of generating potentially nontrivial quantum query algorithms for simplex finding with arbitrary constant rank. From the update cost of table 4.1, we expect the r-simplex finding problem can be solved by a nested quantum walk that searches for 1-tuple, 2-tuple, ..., r-tuple of 1-certificate vertices. For $k \leq r$, the update cost at level $i_1 i_2 \dots i_k$ is expected to be $O\left(\max_{i_{k+1} \dots i_r} n^{a_{i_1 \dots i_r} - a_{i_1 \dots i_k}}\right)$. For $r \geq 5$, it seems impractical to work out the analysis of the nested quantum walk by hand as the definition of $I_{y,i}, C, V_{i_1 \dots i_k}$ becomes convoluted, and the number of constraints in the final linear program scale exponentially with respect to r. However, it's' interesting to find out whether such an approach will give a nontrivial quantum query upper bound for every rank $r = O(1)$.

Although the best quantum query lower bound we know for any r-simplex finding

problem is still trivial, we used a randomized reduction to show that if there is a nontrivial lower bound, the higher-rank simplex findings also exhibit a nontrivial lower bound with the same exponent offset. We also see that a non-adaptive learning graph of certificate structures can be formulated as a supremum problem in equation (5.3a), which allows us to find a nontrivial lower bound for the more difficult 3-simplex sum problem with an appropriate feasible solution.

For future research about closing the gap between quantum query upper and lower bound of the above hypergraph problems, we list some directions and open problems.

- Can we drop the r-partite condition in theorem 24 and have the result hold for arbitrary r-uniform hypergraph?
- The algorithmic approach in section 4.1 seems to fail when $r = 4$. But, does the nested quantum walk approach in section 4.3 generalize to higher ranks? At what rank, if any, does this algorithm become no better than trivial?
- The lower bound in theorem 45 includes an unpleasant logarithmic factor. Can we remove this factor?
- In section 5.3, we find a feasible solution to equations (5.3b) - (5.3c) for a lower bound. Improving our approach to generating feasible solutions for better lower bounds is an interesting research direction.

References

[1] Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and Quantum implications of Huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1330–1342, Virtual Italy, June 2021. ACM.

[2] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59, Hersonissos Greece, July 2001. ACM.

[3] Andris Ambainis. Quantum Lower Bounds by Quantum Arguments. *Journal of Computer and System Sciences*, 64(4):750–767, June 2002.

[4] Andris Ambainis. Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, 1(04):507–518, May 2004. arXiv:quant-ph/0403120.

[5] Andris Ambainis. Quantum Walk Algorithm for Element Distinctness. *SIAM Journal on Computing*, 37(1):210–239, January 2007.

[6] Mohammad A. Bahmanian and Mateja Sajna. Connection and separation in hypergraphs. *Theory and Applications of Graphs*, 2(2), December 2015.

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

[8] Aleksandrs Belovs. Learning-Graph-Based Quantum Algorithm for k-Distinctness. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 207–216, New Brunswick, NJ, USA, October 2012. IEEE.

[9] Aleksandrs Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 77–84, New York New York USA, May 2012. ACM.

[10] Aleksandrs Belovs and Troy Lee. Quantum Algorithm for k-distinctness with Prior Knowledge on the Input, August 2011. arXiv:1108.3022 [quant-ph].

[11] Aleksandrs Belovs and Ansis Rosmanis. On the Power of Non-adaptive Learning Graphs. *computational complexity*, 23(2):323–354, June 2014.

[12] Aleksandrs Belovs and Robert Spalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 323–328, Berkeley California USA, January 2013. ACM.

[13] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight Bounds on Quantum Searching. *Fortschritte der Physik*, 46(4-5):493–505, June 1998.

[14] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In Samuel J. Lomonaco and Howard E. Brandt, editors, *Contemporary Mathematics*, volume 305, pages 53–74. American Mathematical Society, Providence, Rhode Island, 2002.

[15] H. Buhrman, R. Cleve, R. De Wolf, and C. Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 358–368, New York City, NY, USA, 1999. IEEE Comput. Soc.

[16] Titouan Carette, Mathieu Laurière, and Frédéric Magniez. Extended Learning Graphs for Triangle Finding. *Algorithmica*, 82(4):980–1005, April 2020.

[17] Andrew M. Childs and Robin Kothari. Quantum query complexity of minor-closed graph properties. *SIAM Journal on Computing*, 41(6):1426–1450, 2012.

[18] Christoph Durr, Mehdi Mhalla, and Yaohui Lei. Quantum query complexity of graph connectivity, April 2003. arXiv:quant-ph/0303169.

[19] François Le Gall. Improved Quantum Algorithm for Triangle Finding via Combinatorial Arguments. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 216–225, October 2014. arXiv:1407.0085 [quant-ph].

[20] François Le Gall, Harumichi Nishimura, and Seiichiro Tani. Quantum Algorithms for Finding Constant-sized Sub-hypergraphs. *Theoretical Computer Science*, 609:569–582, January 2016. arXiv:1310.4127 [quant-ph].

[21] Peter Hoyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 526–535, San Diego California USA, June 2007. ACM.

[22] Stacey Jeffery, Robin Kothari, and Frederic Magniez. Nested Quantum Walks with Quantum Data Structures. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1474–1485. Society for Industrial and Applied Mathematics, January 2013.

[23] Jeff Kahn, Michael Saks, and Dean Sturtevant. A topological approach to evasiveness. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 31–33, Tucson, AZ, USA, November 1983. IEEE.

[24] Julia Kempe. Quantum random walks - an introductory overview. *Contemporary Physics*, 44(4):307–327, July 2003. arXiv:quant-ph/0303081.

[25] Shelby Kimmel. Quantum Adversary (Upper) Bound. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7391, pages 557–568. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Lecture Notes in Computer Science.

[26] Raghav Kulkarni, Youming Qiao, and Xiaoming Sun. Any monotone property of 3-uniform hypergraphs is weakly evasive. *Theoretical Computer Science*, 588:16–23, July 2015.

[27] Troy Lee, Frederic Magniez, and Miklos Santha. Learning graph based quantum query algorithms for finding constant-size subgraphs. *Chicago Journal of Theoretical Computer Science*, 18(1):1–21, 2012.

[28] Troy Lee, Frédéric Magniez, and Miklos Santha. Improved Quantum Query Algorithms for Triangle Detection and Associativity Testing. *Algorithmica*, 77(2):459–486, February 2017.

[29] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Palek, and Mario Szegedy. Quantum Query Complexity of State Conversion. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 344–353, Palm Springs, CA, USA, October 2011. IEEE.

[30] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via Quantum Walk. *SIAM Journal on Computing*, 40(1):142–164, January 2011. arXiv:quant-ph/0608026.

[31] Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum Algorithms for the Triangle Problem. *SIAM Journal on Computing*, 37(2):413–424, January 2007.

[32] Ben W. Reichardt. Span Programs and Quantum Query Complexity: The General Adversary Bound Is Nearly Tight for Every Boolean Function. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 544–551, Atlanta, GA, USA, October 2009. IEEE.

[33] Ben W. Reichardt. Reflections for quantum query algorithms. In *SODA '11: 22nd ACM-SIAM Symposium on Discrete Algorithms*, San Francisco California, January 2011. Society for Industrial and Applied Mathematics. Publisher: arXiv Version Number: 1.

[34] Ronald L. Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, December 1976.

[35] Miklos Santha. Quantum walk based search algorithms. In *Theory and Applications of Models of Computation*, volume 4978, pages 31–46, Berlin, Heidelberg, August 2008. Springer. arXiv:0808.0059 [quant-ph].

[36] Neil Shenvi, Julia Kempe, and K. Birgitta Whaley. A Quantum Random Walk Search Algorithm. *Physical Review A*, 67(5):052307, May 2003. arXiv:quant-ph/0210064.

[37] M. Szegedy. Quantum Speed-Up of Markov Chain Based Algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41, Rome, Italy, 2004. IEEE.

[38] John Watrous. Quantum simulations of classical random walks and undirected graph connectivity. *Journal of Computer and System Sciences*, 62(2):376–391, March 2001. arXiv:cs/9812012.

[39] Yechao Zhu. Quantum Query Complexity of Constant-sized Subgraph Containment. *International Journal of Quantum Information*, 10(03):1250019, April 2012.

[40] Robert Špalek and Mario Szegedy. All Quantum Adversary Methods Are Equivalent. In *Automata, Languages and Programming*, volume 3580, pages 1299–1311. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. Series Title: Lecture Notes in Computer Science.

APPENDICES

Appendix A

Proof of Lemmas in Section 4.3

In this appendix, we will provide the proof of lemma 36 - 39 presented in section 4.3.

Proof of lemma 36. Fix $v_jv_k \in V_{jk}$, define the set $S_1 = \{v_i \in V_i : v_iv_jv_k \in V_{ijk}\}$. For any $v_i \in V_i$, we have

$$\begin{aligned}\Pr(v_iv_jv_k \in V_{ijk}) &= \Pr(v_iv_jv_k \in V_{ijk} | v_iv_j \in V_{ij}, v_iv_k \in V_{ik}) \Pr(v_iv_j \in V_{ij}) \Pr(v_iv_k \in V_{ik}) \\ &= \frac{1}{11} n^{c_{ijk}-m_{ijk}} \cdot n^{b_{ij}-a_i-a_j} \cdot n^{b_{ik}-a_i-a_k} = \frac{1}{11} n^{c_{ijk}-b_{jk}-a_i}. \end{aligned} \tag{A.1}$$

Therefore, $|S_1|$ is a random variable with distribution $HG(n^{a_i+b_{jk}}, n^{a_i}, n^{c_{ijk}}/11)$. It has mean value $\frac{1}{11}n^{c_{ijk}-b_{jk}}$ and by lemma 33 (1),

$$\Pr\left(|S_1| \geq \frac{1}{6} n^{c_{ijk}-b_{jk}}\right) \leq \exp\left(-\frac{1}{55} n^{c_{ijk}-b_{jk}}\right). \tag{A.2}$$

We can prove a similar bound for conditions 3 and 4. By union bound, we see that

$$\begin{aligned}&\Pr(\text{Conditions } 2,3,4 \text{ don't hold}) \leq \\ &\quad n^{b_{jk}} \exp\left(-\frac{1}{55} n^{c_{ijk}-b_{jk}}\right) + n^{b_{ik}} \exp\left(-\frac{1}{55} n^{c_{ijk}-b_{ik}}\right) + n^{b_{ij}} \exp\left(-\frac{1}{55} n^{c_{ijk}-b_{ij}}\right). \end{aligned} \tag{A.3}$$

For condition 5, fix vertices $v_j \in V_j, v_k \in V_k, v_l \in V_l$ such that $v_jv_k \in V_{jk}, v_jv_l \in V_{jl}$, $v_kv_l \in V_{kl}$. Define $S_1' := \{v_i \in V_i : v_iv_jv_l \in V_{ijl}\}$ and $S_2 := \{v_i \in S_1' : v_iv_kv_l \in V_{ikl}\}$. Note

that for any $v_i \in V_i$, we have

$$\begin{aligned}\Pr(v_iv_kv_l \in V_{ikl} | v_i \in S_1') &= \Pr(v_iv_kv_l \in V_{ikl} | v_iv_l \in V_{il}, v_iv_k \in V_{ik}) \Pr(v_iv_k \in V_{ik}) \\ &= \frac{1}{11} n^{c_{ikl}-m_{ikl}} \cdot n^{b_{ik}-a_i-a_k} = \frac{1}{11} n^{c_{ikl}-b_{kl}-b_{il}+a_l}. \end{aligned} \tag{A.4}$$

So, $|S_2|$ follows the distribution $HG(|S_1'||\Gamma_{jkl}|,\ |S_1'|,\ \frac{1}{11}|S_1'||\Gamma_{jkl}|n^{c_{ikl}-b_{kl}-b_{il}+a_l})$. It has mean $\frac{1}{11}|S_1'|n^{c_{ikl}-b_{kl}-b_{il}+a_l}$. Since we assume C_{ijl} is marked, we have $|S_1'| \leq \frac{1}{6} n^{c_{ijl}-b_{jl}}$. Apply lemma 33 (2) with $\delta = n^{c_{ijl}-b_{jl}}/|S_1'| - 1 > 2e - 1$, we have

$$\Pr\left(|S_2| > \frac{1}{11} n^{c_{ijl}+c_{ikl}-b_{il}-b_{jl}-b_{kl}+a_l}\right) \leq \exp\left(-\frac{\log 2}{11} n^{c_{ijl}+c_{ikl}-b_{il}-b_{jl}-b_{kl}+a_l}\right). \tag{A.5}$$

Finally, define

$$S_3 := \{v_i \in V_i : v_iv_jv_k \in V_{ijk}, v_iv_jv_l \in V_{ijl}, v_iv_kv_l \in V_{ikl}\} = \{v_i \in S_2 : v_iv_jv_k \in V_{ijk}\}.$$

For any $v_i \in V_i$, we have

$$\Pr\left(v_iv_jv_k \in V_{ijk} | v_i \in S_2\right) = \Pr\left(v_iv_jv_k \in V_{ijk} | v_iv_j \in V_{ij}, v_iv_k \in V_{ik}\right) = \frac{1}{11} n^{c_{ijk}-m_{ijk}}. \tag{A.6}$$

So, $|S_3|$ follows the distribution $HG(|S_2||\Gamma_{jkl}|,\ |S_2|,\ \frac{1}{11}|S_2||\Gamma_{jkl}|n^{c_{ijk}-m_{ijk}})$. It has mean $\frac{1}{11}|S_2|n^{c_{ijk}-m_{ijk}}$. Under the condition that $|S_2| \leq \frac{1}{11} n^{c_{ijl}+c_{ikl}-b_{il}-b_{jl}-b_{kl}+a_l}$, we apply lemma 33 (2) with $\delta = n^{c_{ikl}+c_{ijl}-b_{il}-b_{jl}-b_{kl}+a_l}/|S_2| - 1 > 2e - 1$, we have

$$\Pr\left(|S_3| > \frac{1}{11} n^{m_{ijkl}-c_{jkl}} \,\middle|\, |S_2| \leq \frac{1}{11} n^{c_{ijl}+c_{ikl}-b_{il}-b_{jl}-b_{kl}+a_l}\right) \leq \exp\left(-\frac{\log 2}{11} n^{m_{ijkl}-c_{jkl}}\right). \tag{A.7}$$

Combining equations (A.5) and (A.7), we get

$$\begin{aligned}&\Pr(\text{Condition 5 doesn't hold}) \leq \\ &\qquad \Theta(n^{m_{jkl}}) \left[\exp\left(n^{c_{ijl}+c_{ikl}-b_{il}-b_{jl}-b_{kl}+a_l}\right) + \exp\left(n^{m_{ijkl}-c_{jkl}}\right)\right] \end{aligned} \tag{A.8}$$

The statement of this lemma follows from equations (A.3), (A.8), and probability union bound. □

Proof of lemma 37. By the definition of the set Γ_{ijkl}, we see that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{A},s}\left[|\Gamma'_{ijkl}-\Gamma_{ijkl}|\right]=\Theta\left(\frac{1}{n}\right)\sum_s \mathop{\mathbb{E}}_{\boldsymbol{A}}\left[|\Gamma'_{ijkl}-\Gamma_{ijkl}|\right]\\
&\quad\le\Theta\left(\frac{1}{n}\right)\sum_s \mathop{\mathbb{E}}_{\boldsymbol{A}}|\{v_jv_kv_l\in V_{jkl}: v_sv_jv_k\in V_{ijk}, v_sv_jv_l\in V_{ijl}, v_sv_kv_l\in V_{ikl}\}|\\
&\quad=\Theta\left(\frac{1}{n}\right)\sum_s\sum_{v_jv_kv_l\in V_{jkl}}\Pr(v_sv_jv_k\in V_{ijk}, v_sv_jv_l\in V_{ijl}, v_sv_kv_l\in V_{ikl})\\
&\quad=\Theta\left(n^{c_{jkl}}\right)\Theta\left(n^{c_{ijl}-b_{jl}-a_i}\right)\Theta\left(n^{c_{ikl}-b_{kl}-b_{il}+a_l}\right)\Theta\left(n^{c_{ijk}-m_{ijk}}\right)=\Theta\left(n^{m_{ijkl}-a_i}\right).
\end{aligned}$$

The third equality is a consequence of equations (A.1), (A.4), (A.6). Finally, since every tuple in Γ_{ijkl} becomes a quadruple in V_{ijkl} only with probability $\Theta\left(n^{d_{ijkl}-m_{ijkl}}\right)$, we have $\mathbb{E}_{\boldsymbol{A},s}|V'_{ijkl}-V_{ijkl}|=O\left(n^{d_{ijkl}-m_{ijkl}}\cdot\mathbb{E}_{\boldsymbol{A},s}|\Gamma'_{ijkl}-\Gamma_{ijkl}|\right)=O(n^{d_{ijkl}-a_i})$. □

Proof of lemma 38. Fix $v_i\in V_i, v_j\in V_j$, $v_kv_l\in V_{kl}$, we see that

$$\begin{aligned}
&\Pr\left(v_iv_kv_l\in V_{ikl}, v_jv_kv_l\in V_{jkl}\right)\\
=\;&\Pr\left(v_iv_kv_l\in V_{ikl}, v_jv_kv_l\in V_{jkl}|v_iv_k\in V_{ik}, v_jv_k\in V_{jk}\right)\Pr\left(v_iv_k\in V_{ik}, v_jv_k\in V_{jk}\right)\\
=\;&\Theta\left(n^{c_{ikl}-m_{ikl}}\right)\cdot n^{b_{ik}-a_i-a_k}\cdot n^{b_{il}-a_i-a_l}\cdot\Theta\left(n^{c_{jkl}-m_{jkl}}\right)\cdot n^{b_{jk}-a_j-a_k}\cdot n^{b_{jl}-a_j-a_l}\\
=\;&\Theta\left(n^{c_{ikl}+c_{jkl}-2b_{kl}-a_i-a_j}\right).
\end{aligned}\tag{A.9}$$

Similar to the proof of lemma 37 we see that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{A},v}|\Gamma'_{ijkl}-\Gamma_{ijkl}|=\Theta\left(\frac{1}{n^{a_i+a_j}}\right)\sum_{t_it_j}\mathop{\mathbb{E}}_{\boldsymbol{A}}|\Gamma'_{ijkl}-\Gamma_{ijkl}|\\
\le\;&\Theta\left(\frac{1}{n^{a_i+a_j}}\right)\sum_{t_it_j}\mathop{\mathbb{E}}_{\boldsymbol{A}}|\{v_kv_l\in V_{kl}: v_iv_kv_l\in V_{ikl}, v_jv_kv_l\in V_{jkl}, v_iv_jv_k\in V_{ijk}, v_iv_jv_l\in V_{ijl}\}|\\
=\;&\Theta\left(\frac{1}{n^{a_i+a_j}}\right)\sum_{t_it_j}\sum_{v_kv_l\in V_{kl}}\Pr(v_iv_jv_k, v_iv_jv_l|v_iv_kv_l, v_jv_kv_l)\Pr(v_iv_kv_l, v_jv_kv_l)\\
=\;&\Theta(n^{b_{kl}})\Theta(n^{c_{ijk}-m_{ijk}+c_{ijl}-m_{ijl}})\Theta\left(n^{c_{ikl}+c_{jkl}-2b_{kl}-a_i-a_j}\right)=\Theta\left(n^{m_{ijkl}-b_{ij}}\right).
\end{aligned}$$

The second equality is a consequence of equation (A.9). Since every tuple in Γ_{ijkl} becomes a quadruple in V_{ijkl} only with probability $\Theta\left(n^{d_{ijkl}-m_{ijkl}}\right)$, we have $\mathbb{E}_{\boldsymbol{A},v}|V'_{ijkl}-V_{ijkl}|=O\left(n^{d_{ijkl}-m_{ijkl}}\cdot\mathbb{E}_{\boldsymbol{A},v}|\Gamma'_{ijkl}-\Gamma_{ijkl}|\right)=O(n^{d_{ijkl}-b_{ij}})$. □

Proof of lemma 39. Similar to the proof of lemma 37 we see that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\boldsymbol{A},v} |\Gamma'_{ijkl} - \Gamma_{ijkl}| = \frac{1}{|\Gamma_{ijk}|} \sum_{s_i s_j s_k \in \Gamma_{ijk}} \mathop{\mathbb{E}}_{\boldsymbol{A}} |\Gamma'_{ijkl} - \Gamma_{ijkl}| \\
&\leq \frac{1}{|\Gamma_{ijk}|} \sum_{s_i s_j s_k \in \Gamma_{ijk}} \mathop{\mathbb{E}}_{\boldsymbol{A}} |\{v_l \in V_l : v_{s_i} v_{s_j} v_l \in V_{ijl}, v_{s_i} v_{s_k} v_l \in V_{ikl}, v_{s_j} v_{s_k} v_l \in V_{jkl}\}| \\
&= \frac{1}{|\Gamma_{ijk}|} \sum_{s_i s_j s_k \in \Gamma_{ijk}} \sum_{v_l \in V_l} \Pr(v_{s_i} v_{s_j} v_l \in V_{ijl}, v_{s_i} v_{s_k} v_l \in V_{ikl}, v_{s_j} v_{s_k} v_l \in V_{jkl}) \\
&= \Theta(n^{a_l}) \Theta\left(n^{c_{ijl} - b_{ij} - a_l}\right) \Theta\left(n^{c_{ikl} - b_{ik} - b_{il} + a_i}\right) \Theta\left(n^{c_{jkl} - m_{jkl}}\right) = \Theta\left(n^{m_{ijkl} - c_{ijk}}\right).
\end{aligned}$$

The third equality is again a consequence of equations (A.1), (A.4), (A.6). Since every tuple in Γ_{ijkl} becomes a quadruple in V_{ijkl} only with probability $\Theta\left(n^{d_{ijkl} - m_{ijkl}}\right)$, we have $\mathbb{E}_{\boldsymbol{A},v} |V'_{ijkl} - V_{ijkl}| = O\left(n^{d_{ijkl} - m_{ijkl}} \cdot \mathbb{E}_{\boldsymbol{A},v} |\Gamma'_{ijkl} - \Gamma_{ijkl}|\right) = O(n^{d_{ijkl} - c_{ijk}})$. □