THE UNIVERSITY OF CHICAGO

MODELS OF QUERY COMPLEXITY FOR BOOLEAN FUNCTIONS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
SOURAV CHAKRABORTY

CHICAGO, ILLINOIS
JUNE 2008

To Ma and Baba

ABSTRACT

In this thesis we study various models of query complexity. A query algorithm computes a function under the restriction that the input can be accessed only by making probes to the the bits of the input. The query complexity of a function f is the minimum number of probes made by any query algorithm that computes f. In this thesis, we consider three different models of query complexity, (1) deterministic decision tree complexity (query complexity when the underlying algorithm is deterministic), (2) approximate decision tree complexity aka. property testing (query complexity when the underlying algorithm is probabilistic and only expected to "approximately" compute f) and quantum query complexity (query complexity when the underlying algorithm is allowed to make quantum queries).

The main results in this thesis are:

- We study the relation between deterministic decision tree complexity and other combinatorial measures of complexity measures like sensitivity and block sensitivity. We prove that for minterm-transitive functions the sensitivity is quadratically related to block sensitivity which is polynomially related to deterministic decision tree complexity.

- In the context of property testing we obtain the following two results:

  - Given two binary strings of length n and a primitive group $G \subseteq Sym(n)$ we prove that the query complexity for testing isomorphism of strings under the group action is $\widetilde{\Theta}(\sqrt{n \log |G|})$ or $\widetilde{\Theta}(\log |G|)$ depending on whether we have to query both or only one string.
  - Given an undirected graph G and a pair of vertices s and t in G, we design a tester that tests whether there is a directed path from s to t by querying the orientation of at most a constant number of edges.

- We study quantum query complexity in the context of database search algorithms. Given $f : [n] \rightarrow \{0,1\}$ with $|f^{-1}(0)| = \epsilon n$, we design a quantum algorithm that make only t quantum queries to f and outputs a member of $f^{-1}(1)$ with probability at least $(1 - \epsilon^{2t+1})$. We prove that the error reduction achieved is tight.

ACKNOWLEDGEMENTS

First, I thank my advisor László Babai for his continuous support in the Ph.D. program. Laci was always there to give advice and discuss any kind of academic questions. He taught me how to ask questions and express my ideas. I am grateful to him for the collaboration which led to one of the chapters of this thesis, for introducing me to the notion of sensitivity which is an important part in another chapter of this thesis, for his constant help in improving my writing skills and for all that I learned from him during my stay at the University of Chicago.

The other person without whose help, encouragement and guidance this thesis would not have been possible is Lance Fortnow. He was always there to meet and talk about my ideas, to proofread my papers, and to ask me good questions. I am greatly indebted to him for his patient advice and support on matters, technical and otherwise. I also enjoyed the friendly talks and discussions we had in the Theory Lounge mostly during lunch.

I am greatly indebted to Jaikumar Radhakrishnan, Nandakumar Raghunathan, Eldar Fischer, Arie Matsliah, Ilan Newman and Oded Lachish for their collaboration which led to several of the results discussed in this thesis. I would like to specially mention Jaikumar Radhakrishnan from whom I have learnt a lot during my second and third year in the Ph.D. program and without whose guidance the journey through the Ph.D. program would have been much tougher. I would also like to specially thank Eldar Fischer for inviting me to Technion in my fourth and fifth years. Both times I had a fruitful and enjoyable time in Technion. The problem on testing of st-connectivity in the orientation model which is a key chapter in this thesis was solved during my first visit to Technion.

A special thanks is due to the members of my thesis reading committee: Laci Babai, Lance Fortnow and Prahladh Harsha. They kindly found time to go over large parts of my thesis and give numerous useful comments and suggestions.

During my graduate studies, I had the good fortune of being a summer student at Microsoft Research, India. I am most grateful to my manager Satya Lokam at MSR, India. I had invaluable experience at that place and met many great people.

In the past few years it has been my pleasure to be a part of the wonderful UofC-TTI theory group. I would like to thank Prahladh, Nikhil, Tom, Adam, Daniel, Rahul, Nanda, Varsha, Eric, Raghav, Josh, Hari, Kristoffer, Ivona, Chinmoy and many others for numerous discussions on various problems and topics that helped me improve my understanding of different subjects in theoretical computer science.

I owe a lot to my professors from my college (Chennai Mathematical Institute). Without their encouragement and guidance I doubt whether I would have taken to research in theoretical computer science. A special thanks to Narayan Kumar, K V Subrahmanyam and Meena Mahajan.

My Ph.D. program lasted five years. And it would have been impossible without a wonderful and inspiring set of friends. I would like to thank Deepam, Saravanan, Syed, Abhik, Shreya, Sridhar, Suman, Sameer, Ambarish, Anirban, Sanjib, Nanda, Ankan, Duru, Arie, Krishna and many others with whom I have spent so many memorable moments while cooking, playing settlers, playing cricket, watching movies, discussing random stuff and many other activities. They have made my non-academic life in the past few years an exciting one.

My most important acknowledgement is to my close and loving family: my parents and my sister, who have filled my life with joy. Throughout my Ph.D. program they have been the most important source of inspiration for me. Words cannot express my thanks to my parents for all that they have gone through and done for me.

TABLE OF CONTENTS

ABSTRACT iv

ACKNOWLEDGEMENTS vi

LIST OF FIGURES xi

LIST OF TABLES xii

I Introduction **1**

1 MODELS OF QUERY COMPLEXITY 2
1.1 Deterministic Decision Tree Complexity 3
1.1.1 Deterministic Decision Tree Complexity vs. Sensitivity . . . 4
1.2 Approximate Decision Tree Complexity (aka Property Testing) 5
1.2.1 Testing of Graph Properties 6
1.2.2 Testing of Isomorphism under a transitive group action . . . 8
1.3 Quantum Query Complexity 9

II Sensitivity and Block Sensitivity **11**

2 SENSITIVITY OF CYCLICALLY INVARIANT FUNCTIONS 12
2.1 Introduction 12
2.2 Preliminaries 14
2.2.1 Definitions 14
2.2.2 Previous Results 18
2.3 Cyclically Invariant Function with Sensitivity $\Theta(n^{1/3})$ 19
2.3.1 The auxiliary function 19
2.3.2 The new function 20
2.3.3 Properties of the new function 20
2.4 Lower Bound on Sensitivity for Some Classes of Boolean Functions 22
2.5 Generalization of the Results and Open Problems 25

III Approximate Decision Tree Complexity 27

3 TESTING ST-CONNECTIVITY IN THE ORIENTATION MODEL . . 28
3.1 Introduction 28
3.2 Preliminaries 32
3.2.1 Notations 32
3.2.2 Orientation distance, properties and testers 33
3.2.3 Connectivity Programs and Branching Programs 33
3.3 The main result 36
3.3.1 Proof overview 36
3.4 Reducing general graphs to connectivity programs 38
3.4.1 Reducibility between st-connectivity instances 38
3.4.2 Reduction to graphs having high-diameter subgraphs 39
3.4.3 Properties of ϵ-long graphs 40
3.4.4 Bounding the number of edges within wide edge-layers . . . 41
3.4.5 Reduction to bounded width graphs 43
3.4.6 Reducing bounded width graphs to st-connectivity programs 44
3.5 Reducing st-connectivity programs to branching programs 45
3.6 Converting clustered branching programs to non-clustered ones . . . 48
3.7 Wrapping up – Proof of the Main Theorem 50
3.8 Bounded Expansion Lemma 50

4 TESTING EQUIVALENCE UNDER A TRANSITIVE GROUP ACTION 57
4.1 Introduction 57
4.2 Preliminaries 60
4.2.1 Definitions 60
4.2.2 Previous Results 62
4.2.3 Chernoff bounds 63
4.3 Query Complexity for 1-sided-error Testing of Equivalence under some Primitive Group Action 63
4.3.1 Structure of Primitive Groups 63
4.3.2 G-Agreeability Lemma for G Primitive 66
4.3.3 Lower Bounds for 1-sided error Testing 67
4.3.4 Proof of the G-Agreeability Lemma for Primitive Groups . . 68
4.4 Upper bounds for Transitive groups 71
4.5 Lower bounds for Transitive Groups 75
4.6 Tight bounds and comparisons 77
4.7 Future Work 78

IV Quantum Query Complexity **80**

5 QUANTUM QUERY COMPLEXITY FOR DATABASE SEARCH . . . 81
5.1 Introduction . . . 81
5.2 Background, definitions and results . . . 83
5.2.1 Our contributions . . . 85
5.3 Upper bounds: quantum algorithms . . . 87
5.3.1 Alternative algorithms using the standard oracle . . . 87
5.3.2 Algorithms with restrictions on ϵ . . . 91
5.4 Lower bounds . . . 92
5.4.1 Analysis of low degree polynomials . . . 93
5.4.2 Proof that the error function is a low degree polynomial . . 95

REFERENCES . . . 98

LIST OF FIGURES

5.1 The one-query algorithm 88

LIST OF TABLES

2.1 Current knowledge about sensitivity of some classes of Boolean functions 15

4.1 Bounds on the query complexity of Testing of Equivalence under G-isomorphism. 60

4.2 The results of Fischer and Matsliah for Graph Isomorphism. . . . 62

4.3 Corollaries of our results to Graph Isomorphism. 78

Part I

Introduction

CHAPTER 1
MODELS OF QUERY COMPLEXITY

Let $f : \{0,1\}^n \rightarrow \{0,1\}^k$ be the function we are interested in, that is, given an input $x \in \{0,1\}^n$ we want to compute $f(x)$. The restriction on our model of computation is that the bits of the inputs can be accessed only by querying. An algorithm that computes $f(x)$ by querying the bits of the input x is called a *query algorithm*. The kind of queries that can be made to the input and the kind of operations that the algorithm can perform and the kind of errors allowed depends upon the model of query algorithm. Given a model of query algorithm the query complexity for a function in that model is the number of queries the best query algorithm (in the given model) makes to compute the function.

The query complexity of a function for different models of query algorithm helps to understand the inherent difficulty in computing the function and also helps us understand the computational power of various models of computation. In this thesis we consider different models of query algorithms and study the query complexity in these models for some of the central functions in theoretical computer science, like st-connectivity in a graph, hypergraph isomorphism, and database search. Hopefully this thesis is a small step towards better understanding of the complexity of various functions and the computational power of different models of computation.

In this thesis we restrict our attention to query algorithms where the queries are probes to the bits of the input. We consider both classical and quantum queries. In the classical case the queries are of form "what is the i-th bit of the input." In the quantum case the probes to the bits of the input are made in superposition. In the query algorithms models that we consider in this thesis the queries can be adaptive, that is, which bit of the input to query can depend on what bits have already been queried and their outcomes.

In the classical case the output to each probe is either 0 or 1 indicating the queried bit of the input. Thus the algorithm can be described as a binary tree, called *Decision Tree*. The query complexity is thus called the *Decision Tree Complexity*, which is the height of the Decision tree. Decision tree is the backbone of the complexity of almost all classical query algorithms. Hence to have a better understanding of the computational complexity of a function it is crucial to have a good understanding of the decision tree complexity of the function.

In this thesis we consider three different models of query complexity, namely deterministic decision tree complexity, approximate decision tree complexity and quantum query complexity. In the remaining of this section we discuss these three models of query complexity and our results. In none of our related results other resources like time and space have been taken into consideration.

1.1 Deterministic Decision Tree Complexity

Deterministic Decision Tree Complexity of the function f (denoted $D(f)$) is the minimum number of bits of the input that the optimal deterministic classical algorithm has to query/probe to be able to compute the value of the function on that input.

The deterministic decision tree is well studied in the literature in many contexts. In particular it is known that it is closely related to other combinatorial and complexity measures. It is known that $\log D(f)$ is equal to, up to a constant factor, to the time needed to compute f on a CREW PRAM [Nis91]. The deterministic decision tree complexity is also known to be polynomially related to randomized and quantum query complexity and to other combinatorial measures of complexity like the certificate complexity, block sensitivity [Nis91] and degree of representing polynomial [NS94]. For a survey on this subject refer to [BdW00, Cha05b].

1.1.1 Deterministic Decision Tree Complexity vs. Sensitivity

Sensitivity (denoted $s(f)$) is another example of combinatorial measure for Boolean function. It is the number of bits of the input on which the function is sensitive. In other words it is the number of bits of an input such that if we change any one of those bits the function value changes. Cook, Dwork and Reischuk [CDR86] introduced sensitivity as a simple combinatorial complexity measure for Boolean functions providing lower bounds on the time needed on a CREW PRAM model.

The question whether sensitivity is polynomially related to deterministic decision tree complexity is an interesting open problem. It is believed that they are polynomially related.

Block sensitivity is another combinatorial measure of complexity of Boolean functions. The definition of block sensitivity [Nis91] (denoted $bs(f)$) is very similar to that of sensitivity. It is the number of disjoint sets of bits of the input on which the function is sensitive, that is, it is the number of disjoint sets of bits of an input such that if we change the value of all the bits in any of the sets, the function value changes. Clearly, block sensitivity is greater than sensitivity.

Block sensitivity is known to be polynomially related to deterministic decision tree complexity [Nis91]. Thus the question whether sensitivity is polynomially related to deterministic decision tree complexity translates to the question whether sensitivity and block sensitivity are polynomially related.

The big open problem is this area is

For any Boolean function f is $bs(f) = O(s(f)^2)$?

For general functions the best known relation between sensitivity and block sensitivity is exponential. On the other hand, the best known gap between sensitivity and block sensitivity is only quadratic. For functions that have lots of symmetry (like symmetric functions, monotone functions) or structure (like graph properties) the above question has been answered in the affirmative.

Our Results: In this thesis we try to answer the question in the case of cyclically invariant functions or more generally in the case of functions invariant under a transitive group action. It was a long standing belief that sensitivity of cyclically invariant functions is $\Omega(\sqrt{n})$ [Tur84, KK04]. We give a counter example to this belief by constructing a cyclically invariant Boolean function whose sensitivity is $\Theta(n^{1/3})$. We also prove a matching lower bound for a subclass of transitive invariant Boolean functions called minterm-transitive functions (that includes our example). Also for minterm-transitive function we prove that sensitivity and block sensitivity are quadratically related.

The above mentioned results are proved in Chapter 2. Some of these results appeared in [Cha05a].

1.2 Approximate Decision Tree Complexity (aka Property Testing)

Testing model for Boolean functions is a well-studied model of query algorithm. In this model the algorithm is allowed to use randomness. More importantly it is not necessary that the algorithm outputs the correct value for all inputs. Say $f : \{0,1\}^n \to \{0,1\}$ is the function we want to compute. Given an input $x \in \{0,1\}^n$ the algorithm should output correctly with high probability if $f(x) = 1$. Also if $f(x) = 0$ and for all binary string y that have Hamming distance less than ϵn with x we have $f(y) = 0$ then the algorithm should output correctly with high probability. On the rest of the input the algorithm can output anything. In other words: we are allowed to make a mistake if the input x is close to a string y and $f(x) = 0$ while $f(y) = 1$. Approximate decision tree complexity is the maximum number of bits queried by the best such algorithm.

A property is defined to be a subset of $\{0,1\}^n$. In case of the Boolean function f the relevant property is

$$\{x | f(x) = 1\}$$

Thus in the testing model the algorithm tests whether the given input is in the property or "far" from having the property. Thus it is commonly known as property testing.

This concept was introduced in the context of program checking by Blum, Luby and Rubinfeld [BLR93]. A central ingredient in the proof of the MIP=NEXP theorem [BFL91] was the proof that *mulitinearity* can be tested with a *polylogarithmic* number of queries. Rubinfeld and Sudan [RS96] formally defined property testing in the context of algebraic properties. Subsequently, the interest in property testing was extended to graph properties, with applications to learning and approximation [GGR98]. In recent years the field of combinatorial property testing has enjoyed a rapid growth (see, e. g., [AFKS00, AFNS06, AS03, AS05b, AS05a], cf. [Ron01, Fis04]).

In this thesis we study the query complexity for property testing of *st*-connectivity in a graph and for property testing of isomorphism under a transitive group action.

1.2.1 Testing of Graph Properties

The subject of property testing of graph properties started with the seminal paper by Goldreich, Goldwasser and Ron [GGR98]. The field has seen phenomenal growth in the last decade cf. [Ron01, Fis04]). The basic problem in this area is: given a graph we want to test whether the graph has a particular property or is far from it.

There are many different models of testing of graph properties depending on how the graph is represented and what information about the graph is known to the tester in advance. The two more most commonly used models are the "dense graph model" and the "bounded degree model."

In the dense graph model the input is the adjacency matrix of the graph and the entries of the matrix are queried. Thus the question is whether the graph has a particular property or we need to change more than ϵ fraction of the entries of the matrices to make the graph satisfy the property. Note that all non-dense graphs are ϵ close to the graph with no edges. So this model is not well suited for testing in sparse graphs. On the other hand, the "bounded degree graph" model (as the name suggests) is for testing properties in bounded degree graphs.

Recently Halevy, Lachish, Newman and Tsur [HLNT05] suggested a new model. Here the undirected graph is given to the tester in advance. Each edge of the graph has a label that has to be queried. The property is defined over the labels on the edges. As a special case of this model we can think of the labels on the edges as orientations. Thus the property we are testing is a property of the directed graph. The underlying undirected graph is given to the tester in advance and the orientations have to be queried. This model is thus called the "Orientation Model."

This is a model that combines information that has to be queried with information that is known in advance, and so does not readily yield to general techniques such as that of the regularity lemma used in [AFNS06] and [AS05a]. Fischer et all [MFNY08] showed that testing Eulerism in this model takes non-constant number of queries.

Our Results: In this thesis we consider the problem of testing st-connectivity in a graph. That is, we want to test whether there is a directed path from a given vertex s to another given vertex t in a graph. This problem of st-connectivity is a very basic problem in graph theory and has been studied extensively. Hence understanding this problem better is very important. Unfortunately neither the dense graph model nor the bounded degree model is suitable for testing st-connectivity is a directed graph. But testing of st-connectivity in the orientation model is a non-trivial problem.

We prove that given two vertices s and t we can test whether there is a directed path from s to t by using a constant number of queries in the orientation model. This also gives the first non-trivial graph property that can be tested in this model using constant number of queries. We construct a reduction of the st-connectivity problem to the problem of testing languages that are decidable by branching programs, which was solved in [New02]. The reduction combines combinatorial arguments with a concentration type lemma that is proven for this purpose. Unlike the case for many other property testing results, here the resulting testing algorithm is highly non-trivial itself, and not only its analysis.

The above result is proved in Chapter 3. A preliminary version of these results appeared in [CFL$^+$07].

1.2.2 Testing of Isomorphism under a transitive group action

Given two strings x and y in $\{0,1\}^n$ and a transitive group G which is a subgroup of the symmetric group on n points, $Sym(n)$, we want to test whether x and y are isomorphic to each other under the action of G on the indices. We say x is isomorphic to y under the action of G if there is a permutation $\sigma \in G$ such that if we permute the indices of x by σ we get y. This is a very important problem in computer science. The celebrated problem of graph isomorphism is a special case of this problem.

In the testing problem we have to access the bits of the input x and y by querying. We also consider the case when one of the strings is given to the tester in advance and only the bits of the other string have to be queried. So we want to output 1 if x is isomorphic to y and output 0 if we have to change at least ϵ fraction of the bits of x or y to make them isomorphic.

Testing of graph isomorphism in the dense graph model is a special case. Fischer and Matsliah [FM06] achieved almost tight bounds on the query complexity for this problem in the case when both the graphs have to be queried and also in the case when one graph is known in advance.

Our Resuts: In this thesis we generalize some of the the results from Fischer and Matsliah [FM06]. If the group under consideration is primitive then we obtain tight bounds for testing isomorphism under action of G. In the case when both the strings are unknown the query complexity for testing isomorphism is

$\widetilde{\Theta}(\sqrt{n \log |G|})$. And when one the strings is known in advance the query complexity is $\widetilde{\Theta}(\log |G|)$. Lots of commonly used permutation groups are primitive groups. For example, various finite geometries and groups corresponding to graph isomorphism and hyper-graph isomorphism. Thus, in particular, we generalize some of the results of [FM06] to testing hyper-graph isomorphism.

These results are proved in Chapter 4. These results appeared in [BC08].

1.3 Quantum Query Complexity

Quantum query algorithms are quantum algorithms that compute a function by making quantum queries, that is queries to the input in superposition. How powerful is quantum computer compared to classical computers is a natural and important question to ask. In light of this question we consider the problem of quantum database search problem.

Here we are given a function $f : [N] \to \{0, 1\}$, and are required to return an $x \in [N]$ (a target address) such that $f(x) = 1$. One classical query in this case is for a $x \in [N]$, "Is $f(x) = 1$?" Let $\epsilon = |f^{-1}(0)|/N$. If only one classical query is allowed then with probability at most $(1 - \epsilon^2)$ we can produce an x such that $f(x) = 1$. Recently, Grover [Gro05] showed that there is a quantum algorithm that after making one quantum query to the database, returns an $X \in [N]$ (a random variable) such that

$$\Pr[f(X) = 0] = \epsilon^3,$$

Using the same idea, Grover derived a t-query quantum algorithm (for infinitely many t) that errs with probability only ϵ^{2t+1}. Subsequently, Tulsi, Grover and Patel [TGP05] showed, using a different algorithm, that such a reduction can be achieved for all t. Comparing with classical algorithm we note that after t classical queries any algorithm outputs a $X \in f^{-1}(1)$ with error probability at least ϵ^{t+1}.

Error reduction in the form of amplitude amplification is one of the central tools in the design of efficient quantum search algorithms [Gro98a, Gro98b, BHMT02]. In fact, Grover's database search algorithm [Gro96, Gro97] can be thought of as amplitude amplification applied to the trivial algorithm that queries the database at a random location and succeeds with probability at least $\frac{1}{N}$. The key feature of quantum amplitude amplification is that it can boost the success probability from a small quantity δ to a constant in $O(1/\sqrt{\delta})$ steps, whereas, in general, a classical algorithm for this would require $\Omega(1/\delta)$ steps.

Our Results: In this thesis, we use polynomial methods we obtain *lower bounds* that show that the amplification achieved by the quantum algorithm in [TGP05] is essentially optimal. We also present simple alternative algorithms that achieve the same bound as those in Grover [Gro05], and have some other desirable properties. We then study the best reduction in error that can be achieved by a t-query quantum algorithm, when the initial error ϵ is known to lie in an interval of the form $[\ell, u]$. We generalize our basic algorithms and lower bounds, and obtain nearly tight bounds in this setting.

These results are proved in Chapter 5. The results also appeared in [CRR05]

Part II

Sensitivity and Block Sensitivity

CHAPTER 2
SENSITIVITY OF CYCLICALLY INVARIANT FUNCTIONS

2.1 Introduction

Cook, Dwork and Reischuk [CDR86] originally introduced sensitivity as a simple combinatorial complexity measure for Boolean functions providing lower bounds on the time needed by a CREW PRAM. Nisan [Nis91] introduced the concept of block sensitivity and demonstrated the remarkable fact that block sensitivity and CREW PRAM complexity are polynomially related.

Sensitivity is the number of bits of an input such that if we change any one of those bits the function value changes. Block sensitivity is the number of disjoint sets of bits of the input on which the function is sensitive, that is, it is the number of disjoint set of bits of an input such that if we change the value of all the bits in any of the sets the function value changes.

Block Sensitivity is known to be polynomially related to Deterministic, Randomized and Quantum Decision Tree Complexity and to other combinatorial measures of complexity like certificate complexity and degree of representing and approximate polynomials. A more detailed study on these topics are found in [Cha05b], [BdW00], [Weg87]. But whether sensitivity is polynomially related to the above measures is still an open problem.

The largest known gap between sensitivity and block sensitivity is quadratic, as shown by Rubinstein [Rub95]. But for an arbitrary Boolean function the best known upper bound on block sensitivity in terms of sensitivity is exponential. H.-U. Simon [Sim83] gave the best possible lower bound on sensitivity in terms of the number of effective variables. From that it follows that block sensitivity of a function f is $O(s(f)4^{s(f)})$, where $s(f)$ is the sensitivity of the function f. Kenyon and Kutin [KK04] gave the best known upper bound on block sensitivity in terms of sensitivity; their bound is $O\left(\frac{e}{\sqrt{2\pi}}e^{s(f)}\sqrt{s(f)}\right)$.

Nisan pointed out [Nis91] that for *monotone* Boolean functions sensitivity and block sensitivity are equal.

A natural direction in the study of the gap between sensitivity and block sensitivity is to restrict attention to Boolean functions with symmetry. We note that a slight modification of Rubinstein's construction (Example 2.15) gives a Boolean function, invariant under the cyclic shift of the variables, which still shows the quadratic gap between sensitivity and block sensitivity. Turán pointed out [Tur84] that for *symmetric* functions (functions invariant under all permutations of the variables), block sensitivity is within a factor of two of sensitivity. For any non-trivial *graph property* (the $n = \binom{V}{2}$ variables indicate the adjacency relation among the V vertices), Turán [Tur84] proved that sensitivity is at least $V = \Theta(\sqrt{n})$ and therefore the gap is at most quadratic. In the same paper he also asked the following question:

> **Problem (Turán, 1984):** Does a lower bound of similar order hold still if we generalize graph properties to Boolean functions invariant under a transitive group of permutations?

In Section 2.3 we give a cyclically invariant function with sensitivity $\Theta(n^{1/3})$. This example gives a negative answer to Turán's question.

Kenyon and Kutin [KK04] observed that for "nice" functions the product of 0-sensitivity and 1-sensitivity tends to be linear in the input length. Whether this observation extends to all "nice" functions was given as a (vaguely stated) open problem in that paper. In Section 2.3 we also construct a cyclically invariant Boolean function for which the product of 0-sensitivity and 1-sensitivity is $\Theta(\sqrt{n})$. Thus our function also gives a counterexample to Kenyon and Kutin's suggestion.

In Section 2.2.1 we define a class of Boolean functions called the minterm-transitive functions (Definition 2.14). This class of function is a subclass of the class of functions *invariant under some transitive group* (Definition 2.8) and contains our new functions (that we give in Section 2.3). In Section 2.4 we prove that for minterm-transitive functions sensitivity is $\Omega(n^{1/3})$ (where n is the input size) and the product of 0-sensitivity and 1-sensitivity is $\Omega(\sqrt{n})$. We also show that for this class of functions the sensitivity and block sensitivity is quadratically related. Thus for this class of function sensitivity is polynomially related to deterministic decision tree complexity. Our lower bound results generalizes to some bigger classes of Boolean functions (Section 2.5).

2.2 Preliminaries

2.2.1 Definitions

We use the notation $[n] = \{1, 2, 3, ..., n\}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. We call the elements of $\{0,1\}^n$ "words." For any word x and $1 \leq i \leq n$ we denote by x^i the word obtained by switching the ith bit of x. For a word x and $A \subseteq [n]$ we use x^A to denote the word obtained from x by switching all the bits in A. For a word $x = x_1, x_2, ..., x_n$ we define $\mathrm{supp}(x)$ as $\{i \,|\, x_i = 1\}$. Weight of x, denoted $\mathrm{wt}(x)$, is $|\,\mathrm{supp}(x)|$, *i. e.*, number of 1s in x.

Definition 2.1. The *sensitivity* of f on the word x is defines as the number of bits on which the function is sensitive: $s(f, x) = |\{i : f(x) \neq f(x^i)\}|$.

We define the *sensitivity* of f as $s(f) = \max\{s(f, x) : x \in \{0,1\}^n\}$

We define *0-sensitivity* of f as $s^0(f) = \max\{s(f, x) : x \in \{0,1\}^n, f(x) = 0\}$

We define *1-sensitivity* of f as $s^1(f) = \max\{s(f, x) : x \in \{0,1\}^n, f(x) = 1\}$.

Definition 2.2. The *block sensitivity* $bs(f, x)$ of a function f on an input x is the maximum number of disjoint subsets $B_1, B_2, ..., B_r$ of $[n]$ such that for all j, $f(x) \neq f(x^{B_j})$.

The *block sensitivity* of f, denoted $bs(f)$, is $\max_x bs(f, x)$.

	Best Known Lower Bound	Least sensitivity for which an example is known
General Functions	$(\frac{1}{2}\log n - \frac{1}{2}\log\log n + \frac{1}{2})$	$\frac{1}{2}\log m + \frac{1}{4}\log\log m + O(1)$
Symmetric Functions	$\lceil\frac{n+1}{2}\rceil$	$\lceil\frac{n+1}{2}\rceil$
Monotone Functions	$(\frac{1}{2}\log n - \frac{1}{2}\log\log n + \frac{1}{2})$	$\frac{1}{2}\log m + \frac{1}{4}\log\log m + O(1)$
Graph Properties	$\lfloor\frac{\|V\|}{2}\rfloor$	$(\|V\| - 1)$
Cyclically Invariant Functions	-	$\Theta(n^{1/3})$
Minterm Transitive Functions	$(2n)^{1/3}$	$\Theta(n^{1/3})$

Table 2.1: Current knowledge about sensitivity of some classes of Boolean functions

Definition 2.3. A *partial assignment* is a function $p : S \to \{0,1\}$ where $S \subseteq [n]$. We call S the support of this partial assignment. The weight of a partial assignment is the number of elements in S that are mapped to 1. We call x a (full) assignment if $x : [n] \to \{0,1\}$. (Note than any word $x \in \{0,1\}^n$ can be thought of as a full assignment.) We say $p \subseteq x$ if x is an extension of p, *i.e.*, the restriction of x to S denoted $x|_S = p$.

Definition 2.4. A *1-certificate* is a partial assignment, $p : S \to \{0,1\}$, which forces the value of the function to 1. Thus if $x|_S = p$ then $f(x) = 1$.

Definition 2.5. If $\mathcal{F}$ is a set of partial assignments then we define $m_{\mathcal{F}} : \{0,1\}^n \to \{0,1\}$ as $m_{\mathcal{F}}(x) = 1 \iff (\exists p \in \mathcal{F})$ such that $(p \subseteq x)$.

Note that each member of $\mathcal{F}$ is a 1-certificate for $m_{\mathcal{F}}$ and $m_{\mathcal{F}}$ is the unique smallest such function. (Here the ordering is pointwise, *i.e.*, $f \leq g$ if for all x we have $f(x) \leq g(x)$).

Definition 2.6. A *minterm* is a minimal 1-certificate, that is, no sub-assignment is a 1-certificate.

Definition 2.7. Let $S \subseteq [n]$ and let $\pi \in S_n$. Then we define S^π to be $\{\pi(i) \mid i \in S\}$.

Let G be a permutation group acting on $[n]$. Then the sets S^π, where $\pi \in G$, are called the *G-shifts* of S. If $p : S \to \{0,1\}$ is a partial assignment then we define $p^\pi : S^\pi \to \{0,1\}$ as $p^\pi(i) = p(\pi^{-1}i)$.

Definition 2.8. Let G be a subgroup of S_n, *i.e.*, a permutation group acting on $[n]$. A function $f : \{0,1\}^n \to \{0,1\}$ is said to be *invariant under the group G* if for all permutations $\pi \in G$ we have $f(x^\pi) = f(x)$ for all $x \in \{0,1\}^n$.

Definition 2.9. Let $x = x_1x_2...x_n \in \{0,1\}^n$ be a word. Then for $0 < \ell < n$, we denote by $cs_\ell(x)$ the word $x_{\ell+1}x_{\ell+2}...x_nx_1x_2...x_\ell$, *i.e.*, the *cyclic shift* of the variables of x by ℓ positions.

Definition 2.10. A function $f : \{0,1\}^n \to \{0,1\}$ is called *cyclically invariant* if $f(x) = f(cs_1(x))$ for all $x \in \{0,1\}^n$.

Note that a cyclically invariant function is invariant under the group of cyclic shifts.

Proposition 2.11. *Let G be a permutation group. Let $p : S \to \{0,1\}$ be a partial assignment and let $\mathcal{F} = \{p^\pi \mid \pi \in G\}$. Then p is a minterm for the function $m_{\mathcal{F}}$.*

The function $m_{\mathcal{F}}$ will be denoted p^G. Note that the function p^G is invariant under the group G. When G is the group of cyclic shifts we denote the function p^{cyc}. The function p^{cyc} is cyclically invariant.

Proof of Proposition 2.11. If p has k zeros then for any word x with fewer than k zeros $m_{\mathcal{F}}(x) = 0$, since all the element of $\mathcal{F}$ has same number of 1s and 0s. But if q is a 1-certificate with fewer than k zeros we can have a word x by extending q to a full assignment by filling the rest with 1s, satisfying $f(x) = 1$ (since $q \subseteq x$). But x contains fewer than k zeros, a contradiction. So no minterm of $m_{\mathcal{F}}$ has fewer than k zeros.

Similarly no minterm of $\mathcal{F}$ has weight less than p. So no proper sub-assignment of p can be a 1-certificate. Hence p is a minterm of $m_{\mathcal{F}}$. $\square$

Definition 2.12. Let G be a permutation group on $[n]$. G is called *transitive* if for all $1 \le i, j \le n$ there exists a $\pi \in G$ such that $\pi(i) = j$.

Definition 2.13. Let $C(n,k)$ be the set of Boolean functions f on n variables such that there exists a partial assignment $p : S \to \{0,1\}$ with support $k (\neq 0)$ for which $f = p^{cyc}$. Let $C(n) = \cup_{k=1}^{n} C(n,k)$. We will call the functions in $C(n)$ **minterm-cyclic**. These are the simplest cyclically invariant functions.

Definition 2.14. Let G be a permutation group on $[n]$. We define $D_G(n,k)$ (for $k \neq 0$) to be the set of Boolean functions f on n variables such that there exists a partial assignment $p : S \to \{0,1\}$ with support k for which $f = p^G$. We define $D_G(n)$ to be $\cup_{k=1}^{n} D_G(n,k)$. This is a class of simple G-invariant Boolean functions.

We define $D(n)$ to be $\cup_G D_G(n)$ where G ranges over all transitive groups. We call these functions **minterm-transitive**. Note that the class of minterm-cyclic functions is a subset of the class of minterm-transitive functions.

2.2.2 Previous Results

The largest known gap between sensitivity and block sensitivity is quadratic, given by Rubinstein [Rub95]. Although Rubinstein's example is not cyclically invariant, the following slight modification is cyclically invariant with a similar gap between sensitivity and block sensitivity.

Example 2.15. Let $g : \{0,1\}^k \to \{0,1\}$ be such that $g(x) = 1$ iff x contains two consecutive ones and the rest of the bits are 0. In function $f' : \{0,1\}^{k^2} \to \{0,1\}$ the variables are divided into groups $B_1, \ldots, B_k$ each containing k variables. $f'(x) = g(B_1) \vee g(B_2) \vee \cdots \vee g(B_k)$. Using f' we define the function $f : \{0,1\}^{k^2} \to \{0,1\}$ as $f(x) = 1$ iff $f(x') = 1$ for some x' which is a cyclic shift of x. The sensitivity of f is $2k$ while the block sensitivity is $\lfloor \frac{k^2}{2} \rfloor$.

Hans-Ulrich Simon [Sim83] proved that for any function f we have $s(f) \geq (\frac{1}{2}\log n - \frac{1}{2}\log\log n + \frac{1}{2})$, where n is the number of effective variables (the ith variable is effective if there exist some word x for which $f(x) \neq f(x^i)$). This bound is tight. Although for various restricted classes of functions better bounds are known.

Let $f : \{0,1\}^m \to \{0,1\}$ be a Boolean function that takes as input the adjacency matrix of a graph G and evaluates to 1 iff the graph G has a given property. So the input size m is $\binom{|V|}{2}$ where $|V|$ is the number of vertices in the graph G. Also $f(G) = f(H)$ whenever G and H are isomorphic as graphs. Such a function f is called a *graph property*. György Turán [Tur84] proved that non-trivial graph properties have sensitivity $\Omega(\sqrt{m})$.

A function f is called *monotone* if $f(x) \leq f(y)$ whenever $\mathrm{supp}(x) \subseteq \mathrm{supp}(y)$. Nisan[Nis91] pointed out that for monotone functions sensitivity and block sensitivity are the same.

In the definition of block sensitivity (Definition 2.2) if we restrict the block size to be at most ℓ then we obtain the concept of ℓ-block sensitivity of the function f, denoted $s_\ell(f)$. In [KK04] Kutin and Kenyon introduced this definition and proved that $bs_\ell(f) \leq c_\ell s(f)^\ell$ where c_ℓ is a constant depending on ℓ.

Recently Sun [Sun06] proved that for functions that are invariant under some transitive group action the block sensitivity is at least $n^{1/3}$ and they gave an example of a function that has block sensitivity $O(n^{3/7} \log n)$.

Table 2.1 gives a list of classes of Boolean functions and the current knowledge of sensitivity for these classes of functions.

2.3 Cyclically Invariant Function with Sensitivity $\Theta(n^{1/3})$

In this section we will construct a cyclically invariant Boolean function which has sensitivity $\Theta(n^{1/3})$. We will also construct a cyclically invariant function for which the product of 0-sensitivity and 1-sensitivity is $\Theta(\sqrt{n})$.

Theorem 2.16. *There is a cyclically invariant function, $f : \{0,1\}^n \to \{0,1\}$, such that, $s(f) = \Theta(n^{1/3})$.*

Theorem 2.17. *There is a cyclically invariant function, $f : \{0,1\}^n \to \{0,1\}$, such that, $s^0(f)s^1(f) = \Theta(\sqrt{n})$.*

For proving the above theorems we will first define an auxiliary function g on k^2 variables ($k^2 \leq n$). Then we use g to define our new minterm-cyclic function f on n variables. If we set $k = \lfloor n^{1/3} \rfloor$, Theorem 3.1 will follow. Theorem 3.2 follows by setting $k = \lfloor \sqrt{n} \rfloor$.

2.3.1 The auxiliary function

We first define $g : \{0,1\}^{k^2} \to \{0,1\}$ where $k^2 \leq n$. We divide the input into k blocks of size k each. We define g by a regular expression.

$$g(z) = 1$$

$$\Updownarrow$$

$$z \in \underbrace{110^{k-2}}_{k}(\underbrace{11111(0+1)^{k-5}}_{k})^{k-2}\,\underbrace{11111(0+1)^{k-8}111}_{k}$$

We call this regular expression $\mathcal{R}$.

In other words, let $z \in \{0,1\}^{k^2}$ and let $z = z_1 z_2 ... z_k$, where each $z_i \in \{0,1\}^k$ for all $1 \leq i \leq k$, *i. e.*, z is broken up into k blocks of size k each. Then $g(z) = 1$ iff $z_1 = (11\underbrace{00...0}_{k-2})$ and for all $2 \leq j \leq k$ the first five bits of z_j are 1 and also the last 3 bits of z_k are 1. Note that g does not depend on the rest of the bits.

2.3.2 The new function

Now we define the function f using the auxiliary function g. Let $x|_{[m]}$ denote the word formed by the first m bits of x. Let us set

$$f(x) = 1 \iff \exists \ell \text{ such that } g\left(cs_\ell(x)|_{[k^2]}\right) = 1.$$

In other words, viewing x as laid out on a cycle, $f(x) = 1$ iff x contains a contiguous substring y of length k^2 on which $g(y) = 1$.

2.3.3 Properties of the new function

It follows directly from the definition that f is a cyclically invariant Boolean function.

It is important to note that the function g is so defined that the value of g on input z depends only on $(6k - 2)$ bits of z.

Also note that the pattern defining g is so chosen that if $g(z) = 1$ then there is exactly one set of consecutive $(k-2)$ zeros in z and no other set of consecutive $(k-4)$ zeros.

Claim 2.18. *The function f has (a) 1-sensitivity $\Theta(k)$ and (b) 0-sensitivity $\Theta(\frac{n}{k^2})$*

Proof of Claim 2.18. (a) Let x be the following word:

$$(110^{k-2}(111110^{k-5})^{k-2}111110^{k-8}111)0^{n-k^2}$$

Note that $f(x) = 1$. Also it is easy to see that on this input x 1-sensitivity of f is at least $(6k-2)$.

Now let $x \in \{0,1\}^n$ be such that $f(x) = 1$ and there exists $1 \leq i \leq n$ such that $f(x^i) = 0$. But $f(x) = 1$ implies that some cyclic shift of x contains a contiguous substring z of length k^2 such that $g(z) = 1$. But since g depends only on the values of $(6k-2)$ positions so one of those bits has to be switched so that f evaluates to 0. Thus $s^1(f) \leq 6k-2$.

Combined with the lower bound $s^1(f) \geq 6k-2$ we conclude $s^1(f) = \Theta(k)$.

(b) Let $\lfloor \frac{n}{k^2} \rfloor = m$ and $r = (n - k^2 m)$. Let x be

$$(1\underline{0}0^{k-2}(111110^{k-5})^{k-2}111110^{k-8}111)^m 0^r$$

Note that $f(x) = 0$. But if we switch any of the underlined zero the function evaluates to 1. Note that the function is not sensitive on any other bit. So on this input x the 0-sensitivity of f is $m = \lfloor \frac{n}{k^2} \rfloor$ and therefore $s^0(f) \geq \frac{n}{k^2}$.

Now let $x \in \{0,1\}^n$ and assume $f(x) = 0$ while $f(x^i) = 1$ for some $1 \leq i \leq n$. By definition, the 0-sensitivity of f is the number of such i. For each such i there exists a contiguous substring z_i of length k^2 of some cyclic shift of x^i such that $g(z_i) = 1$. Now consider the $z_i^i \subseteq x$ (recall z_i^i denotes the partial assignment obtained by switching the ith bit of z_i). Due to the structure of the pattern $\mathcal{R}$ we note that z_i has exactly one set of consecutive $(k-2)$ zeros. So z_i^i either has exactly one set of consecutive $(k-1)$ zeros or exactly one set of consecutive $(k-2)$ zeros or exactly one set of consecutive $(k-2)$ bits with at most one of the bits being 1 while the remaining bits are zero. So the supports of any two z_i^i either have at least (k^2-1) positions in common or they have at most three positions in common (since the pattern $\mathcal{R}$ begins and ends with 11). Hence the number of distinct z_i^i is at most $\Theta(\frac{n}{k^2})$. Hence we have $s^0(f) = O(\frac{n}{k^2})$.

Combined with $s^0(f) \geq \frac{n}{k^2}$ we conclude that $s^0(f) = \Theta(\frac{n}{k^2})$. □

Proof of Theorem 2.16: From Claim 2.18 it follows $s(f) = \max\left\{\Theta(k), \Theta(\frac{n}{k^2})\right\}$ (since $s(f) = \max s^0(f), s^1(f))$. So if we set $k = \lfloor n^{1/3} \rfloor$ we obtain $s(f) = \Theta(n^{1/3})$. □

Proof of Theorem 2.17: From Claim 2.18 we obtain $s^0(f)s^1(f) = \Theta(\frac{n}{k})$. So if we set $k = \lfloor \sqrt{n} \rfloor$ we have $s^0(f)s^1(f) = \Theta(\sqrt{n})$. □

Theorem 2.16 answers Turán's problem [Tur84] (see the Introduction) in the negative. In [KK04], Kenyon and Kutin asked whether $s^0(f)s^1(f) = \Omega(n)$ holds for all "nice" functions f. Although they do not define "nice," arguably our function in Theorem 2.17 is nice enough to answer the Kenyon-Kutin question in the negative.

In the next section we prove that for a minterm-transitive function, sensitivity is $\Omega(n^{1/3})$ and the product of 0-sensitivity and 1-sensitivity is $\Omega(\sqrt{n})$. Hence our examples are tight.

2.4 Lower Bound on Sensitivity for Some Classes of Boolean Functions

Theorem 2.19. *If f is a minterm-transitive function on n variables then $s(f) = \Omega(n^{1/3})$ and $s^0(f)s^1(f) = \Omega(\sqrt{n})$.*

To prove this theorem we will use the following three lemmas. Since f is a minterm-transitive function, *i.e.*, $f \in D(n)$, we can say $f \in D_G(n,k)$ for some transitive group G and some $k \neq 0$.

Lemma 2.20. *If $f \in D_G(n,k)$ then $s^1(f) \geq \frac{k}{2}$.*

Proof. Let y be the minterm defining f. Without loss of generality $\mathrm{wt}(y) \geq \frac{k}{2}$. Let us extend y to a full assignment x by assigning zeros everywhere outside the support of y. Then switching any 1 to 0 changes the value of the function from 1 to 0. So we obtain $s(f,x) \geq \frac{k}{2}$. Hence $s^1(f) \geq \frac{k}{2}$. □

Lemma 2.21. *If S is a subset of $[n]$, $|S| = k$ then there exist at least $\frac{n}{k^2}$ disjoint G-shifts of S.*

Proof. Let T be a maximal union of G-shifts of S. Since T is maximal T hits all G-shifts of S. Let H_{ij} be the subgroup of G that sends ith bit of the minterm to the bit position j. So the bit position j can at most hit $\sum_{i=1}^{k} |H_{ij}|$ many G-shifts of S. Since G is transitive for all i and j we have $|H_{ij}| \leq \frac{|G|}{n}$. Therefore $\forall j$, $\sum_{i=1}^{k} |H_{ij}| \leq k\frac{|G|}{n}$. That is each bit position hits at most $k\frac{|G|}{n}$ many G-shifts of S. But there are $|G|$ different G-shifts of S. So we must have $|T| \geq \frac{n}{k}$. Hence T must be a union of at least $\frac{n}{k^2}$ disjoint G-shifts of S. And this proves the lemma. □

Lemma 2.22. *Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant Boolean function invariant under a transitive group G. Let $S \subset \{0,1\}^n$ and let $p : S \to \{0,1\}$ be a 1-certificate of f. If the support of p has size k (i.e.$|S| = k$), then $s^0(f) \geq \frac{n}{k^2}$.*

Proof. By Lemma 2.21 we can have $\frac{n}{k^2}$ disjoint G-shifts of p. The union of these disjoint G-shifts of p defines a partial assignment. Let $\hat{S} = \{s_1, s_2, ..., s_r\}$ be the support of the partial assignment. And let Y_{s_i} be the value of the partial assignment in the s_i-th entry.

Since the function f is not a constant function, there exists a word z such that $f(z) = 0$. The i-th bit of z is denoted by z_i. We define,

$$T = \{j \mid z_j \neq Y_{s_m}, s_m = j\}$$

Now let $P \subseteq T$ be a maximal subset of T such that $f(z^P) = 0$. Since P is maximal, if we switch any other bit in $T \backslash P$ the value of the function f will change to 1.

So $s(f, z^P) \geq |(T \backslash P)|$. Now since $f(z^P) = 0$ we note that z^P does not contain any G-shift of p. But from Lemma 2.21 we know that z^T contains $\frac{n}{k^2}$ disjoint G-shifts of p. So $|(T \backslash P)|$ is $\frac{n}{k^2}$ and thus $s^0(f) \geq s(f, z^P) = \frac{n}{k^2}$. □

Corollary 2.23. *If $f \in D_G(n, k)$ then $s^0(f) \geq \frac{n}{k^2}$.*

Proof of Theorem 2.19. From the Lemma 2.20 and Corollary 2.23 we obtain,

$$s(f) = \max\{s^0(f), s^1(f)\} = \max\left\{\frac{n}{k^2}, \frac{k}{2}\right\}.$$

This implies $s(f) \geq (2n)^{1/3}$.

Now since $s^0(f)$ and $s^1(f)$ cannot be smaller than 1, it follows from the Lemma 2.20 and Lemma 2.23 that

$$s^0(f)s^1(f) = \max\left\{\frac{n}{2k}, \frac{k}{2}\right\}.$$

So $s^0(f)s^1(f) \geq \sqrt{n}$. □

The new function we looked at in Theorem 2.16 is minterm-transitive and has sensitivity $\Theta(n^{\frac{1}{3}})$. Thus this lower bound on sensitivity is tight for minterm-transitive functions. Similarly for the function in Theorem 2.17 the product of 0-sensitivity and 1-sensitivity is tight.

We can even give tight upper bound on block sensitivity for minterm-transitive functions in terms of sensitivity.

Theorem 2.24. *For a minterm-transitive function $f : \{0,1\}^n \to \{0,1\}$ we have $s(f) \geq \frac{bs(f)}{k}$, where k is the size of the minterm.*

Proof. Let x is the word for which the block sensitivity is $bs(f)$. Now let the minimal blocks be $B_1, B_2, \cdots, B_{bs(f)}$. By definition $f(x^{B_i}) \neq f(x)$. That means

x^{B_i} has at least one minterm. Choose any minterm. Now that minterm can intersect at most $(k-1)$ other blocks. By Turan's theorem of independence number of sparse graphs we obtain $\frac{bs(f)}{k}$ blocks such that if we switch any block a minterm will be formed that does not intersects any other block. Let the union of these blocks be B.

Now let $A \subset B$ be the maximal set such that $f(x^A) = f(x)$. So x^A has sensitivity more than $B \backslash A$. And $B \backslash A$ must have have at least one bit from all the blocks because if any block is switched fully then a minterm is formed because the minterm does not intersect any other block. So $|B \backslash A| \geq \frac{(bs(f)}{k}$. Hence $s(f, x^A) \geq \frac{bs(f)}{k}$. □

Corollary 2.25. *For miniterm-transitive function, $bs(f) \leq s(f)^2$.*

Proof. Follows from Theorem 2.24 and Lemma 2.20. □

Hence for minterm-transitive functions, sensitivity and block sensitivity does not have a gap of more than quadratic. And this is tight.

2.5 Generalization of the Results and Open Problems

Note that Lemma 2.22 hold for any function invariant under some transitive group. But unfortunately the proof of Lemma 2.20 does not generalizes for all functions closed under some transitive group action. But the proof of Lemma 2.20 uses the fact that all the minterms of the Boolean function have more than $k/2$ number of 0 or all of them more than $k/2$ number of 1. Thus for functions having that property we can prove a similar lemma and hence for these kind of functions the sensitivity is also $\Omega(n^{1/3})$.

For the proof of Theorem 2.24 the fact that is used is that all the minterms are of support less than k. So if a Boolean function, f, that is invariant under some transitive group, has all minterms of size k and weight at least (or at most) k/c for some constant c then the sensitivity of the function is $\Omega(n^{1/3})$ and $bs(f) = O(s(f)^2)$.

The main question in this field is still open: Are sensitivity and block sensitivity polynomially related? Can the gap between them be more than quadratic? The following variant of Turán's question remains open:

Problem: If f is a Boolean function invariant under a transitive group of permutations then is it true that $s(f) = \Omega(n^c)$ for some constant $c > 0$?

Part III

Approximate Decision Tree Complexity

CHAPTER 3
TESTING ST-CONNECTIVITY IN THE ORIENTATION MODEL

3.1 Introduction

In this chapter we continue the study, started in [HLNT05], of property testing of graphs in the orientation model. This is a model that combines information that has to be queried with information that is known in advance, and so does not readily yield to general techniques such as that of the regularity lemma used in [AFNS06] and [AS05a].

Specifically, the information given in advance is an underlying undirected graph $G = (V, E)$ (that may have parallel edges). The input is then constrained to be an orientation of G, and the distances are measured relative to $|E|$ and not to any function of $|V|$. An orientation of G is simply an orientation of its edges. That is, for every edge $e = u, v$ in $E(G)$ an orientation of G specifies which of u and v is the source vertex of e, and which is the target vertex. Thus an orientation defines a directed graph $\vec{G}$ whose undirected skeleton is G. Given the undirected graph G, a property of orientations is just a subset set of all orientations of G.

We study orientation properties in the framework of *property testing*. Here the relevant combinatorial structure is an orientation $\vec{G}$ of the underlying graph G, and the distance between two orientations $\vec{G}_1$, $\vec{G}_2$ is the number of edges that are oriented differently in $\vec{G}_1$ and $\vec{G}_2$. Thus an orientation $\vec{G}$ is ϵ*-far* from a given property P if at least $\epsilon|E(G)|$ edges have to be redirected in $\vec{G}$ to make it satisfy P. Ideally the number of queries that the tester makes depends only on ϵ and on nothing else (neither $|E|$ nor the specific undirected graph G itself).

A major question that has remained open in [HLNT05] is whether connectivity properties admit such a test. For a fixed $s, t \in V(G)$, an orientation $\vec{G}$ is *st*-connected if there is a directed path from s to t in it. Connectivity and in particular *st*-connectivity is a very basic problem in graph theory which has been extensively studied in various models of computation.

Our main result is that the property of being *st*-connected is testable by a one-sided error algorithm with a number of queries depending only on ϵ. That is, we construct a randomized algorithm that for any underlying graph G, on input of an unknown orientation the algorithm queries only $O(1)$ edges for their orientation and based on this decides with success probability $\frac{2}{3}$ (this of course could be amplified to any number smaller than 1) between the case that the orientation is *st*-connected and the case that it is ϵ-far from being *st*-connected. Our algorithm additionally has one-sided error, meaning that *st*-connected orientations are accepted with probability 1. Note that the algorithm knows the underlying graph G in advance and G is neither alterable nor part of the input to be queried. The dependence of the number of queries in our test on ϵ is triply exponential, but it is independent of the size of the graph.

To put our result in context with previous works in the area of property testing, we note that graph properties were extensively studied since the early beginning in the defining paper of Goldreich, Goldwasser and Ron [GGR98]. The model that was mainly studied is the *dense graphs* model in which an input is a graph represented as a subgraph of the complete graph. As such, for n-vertex graphs, the input representation size is $\binom{n}{2}$ which is the number of all possible unordered pairs. Thus, any property that has $o(n^2)$ witness size, and in particular the property of undirected *st*-connectivity, is trivially testable as every input is close to the property. Similarly, properties of directed graphs were studied in the same context mostly by [AFNS06, AS05a]. Inputs in this model are subgraphs of the complete directed graph (with or without anti parallel edges). In this case, directed *st*-connectivity is again trivial.

Other models in which graph properties were studied are the *bounded-degree* graph model of [GR02] in which a sparse representation of sparse graphs is considered (instead of the adjacency matrix as in the dense model), and the *general density* model (also called the *mixed* model) of [PR02] and [KKR04]. In those models edges can be added as well, and so st-connectivity (directed or not) is again trivial as the single edge (s, t) can always be added and thus every graph is close to having the property. Testing general (all-pairs) connectivity is somewhat harder, and for this constant query testers are generally known, e.g. the one in [GR02] for the undirected bounded degree case.

Apart from [HLNT05], the most related work is that of [HLNT07] in which a graph $G = (V, E)$ is given and the properties are properties of boolean functions $f : E(G) \to \{0, 1\}$. In [HLNT07] the interpretation of such a function is as an assignment to certain formulae that are associated with the underlying graph G, and in particular can viewed as properties of orientations (although the results in [HLNT07] concentrate on properties that are somewhat more "local" than our "global" property of being st-connected). Hence, the results here should be viewed as moving along the lines of the investigation that was started in [HLNT05] and [HLNT07]. A common feature of the current work with this previous one, which distinguishes these results from results in many other areas of property testing and in particular the dense graph models, is that the algorithms in themselves are rather non-trivial in construction, and not just in their analysis.

The algorithm that we present here for st-connectivity involves a preprocessing stage that is meant to reduce the problem to that of testing a branching program of bounded width. Once this is achieved, a randomized algorithm simulating the test for the constant width branching program from [New02] is executed to conclude the result.

In general, the decision problem of st-connectivity of orientations of a given graph is not known to be reducible to constant width branching programs. In fact, this is not likely as st-connectivity is complete for NL (non-deterministic LOG space) while deciding constant width branching programs is in L.

In particular, it is not clear how to deal with high degree vertices or with large cuts. The purpose of the preprocessing stage is to get rid of these difficulties (here it would be crucial that we only want to distinguish between inputs that have the property and inputs that are quite far from the having the property). This is done in several steps that constitute the main part of the paper. In particular we have an interim result in which most but not all edges of the graph are partitioned into constant width layers. This is proved using a bounded expansion lemma for sequences of integers, which is formulated and proved for this purpose.

After the small portion of edges not in constant width layers is dealt with (using a reduction based on graph contraction), we can reduce the connectivity problem to a constant width read once branching program. Once such a branching program is obtained, the result of [New02] can be used essentially in a black box manner.

Some interesting related open problems still remain. We still do not know if the property of being *strongly st*-connected is testable with a constant number of queries. The orientation $\vec{G}$ is *strongly st*-connected if there is a directed path in $\vec{G}$ from s to t as well as a directed path from t to s. A more generalized problem is whether in this model we can test using a constant number of queries the property of being *all-pairs strongly-connected*. Another related property is the property that for a given $s \in V(G)$ every vertex is reachable by a directed path from s. We do not know yet what is the complexity of these problems, although there are some indications that similar methods as those used here may help in this regard.

The rest of this chapter is organized as follows. In Section 3.2 we introduce the needed notations and definitions. Section 3.3 contains the statement of the main result and an overview of the proof. In Section 3.4 we reduce the problem of testing *st*-connectivity in general graphs to the problem of testing *st*-connectivity in nicely structured bounded-width graphs (we later call them *st-connectivity programs*). In Section 3.5 we reduce from testing *st*-connectivity programs to testing *clustered* branching programs. In Section 3.6 we convert these clustered branching programs into regular ones, to which we can apply the testing algorithm from [New02]. Finally in Section 3.7 we combine these ingredients to wrap up the proof.

3.2 Preliminaries

3.2.1 Notations

In what follows, our graphs are going to possibly include parallel edges, so we use 'named' pairs for edges, i.e. we use $e = e\{u, v\}$ for an undirected edge named e whose end points are u and v. Similarly, we use $e = e(u, v)$ to denote the directed edge named e that is directed from u to v. Let $G = (V, E)$ be an undirected multi-graph (parallel edges are allowed), and denote by n the number of vertices in V. We say that a directed graph $\vec{G}$ is an *orientation* of the graph G, or in short a G-orientation, if we can derive $\vec{G}$ from G by replacing every undirected edge $e = e\{u, v\} \in E$ with either $e(u, v)$ or $e(v, u)$, but not both. We also call G the *underlying graph* of $\vec{G}$.

Given an undirected graph G and a subset $W \subset V$ of G's vertices, we denote by $G(W)$ the induced subgraph of G on W, and we denote by $E(W) = E(G(W))$ the edge set of $G(W)$. The distance between two vertices u, v in G is denoted by $\text{dist}_G(u, v)$ and is set to be the length of the shortest path between u and v. Similarly, for a directed graph $\vec{G}$, $\text{dist}_{\vec{G}}(u, v)$ denotes the length of the shortest *directed* path from u to v. The distance of a vertex from itself is $\text{dist}_{\vec{G}}(v, v) = \text{dist}_G(v, v) = 0$. In the case where there is no directed path from u to v in $\vec{G}$, we set $\text{dist}_{\vec{G}}(u, v) = \infty$. The diameter of an undirected graph G is defined as $\text{diam}(G) = \max_{u,v \in V}\{\text{dist}_G(u, v)\}$. Through this paper we always assume that the underlying graph G is connected, and therefore its diameter is finite.

For a graph $\vec{G}$ and a vertex $v \in V$, let $\Gamma_{in}(v) = \{u : \exists e(u, v) \in E\}$ and $\Gamma_{out}(v) = \{u : \exists e(v, u) \in E\}$ be the set of incoming and outgoing neighbors of v respectively, and let $\Gamma(v) = \Gamma_{in}(v) \cup \Gamma_{out}(v)$ be the set of neighbors in the underlying graph G. Let $\deg_{in}(v)$, $\deg_{out}(v)$ and $\deg(v)$ denote the sizes of $\Gamma_{in}(v)$, $\Gamma_{out}(v)$ and $\Gamma(v)$ respectively. We denote the i-neighborhood (in the underlying undirected graph G) of a vertex v by $N_i(v) = \{u : \text{dist}_G(u, v) \leq i\}$. For example, $N_1(v) = \{v\} \cup \Gamma(v)$, and for all $v \in V$, $V = N_{\text{diam}(G)}(v)$.

3.2.2 Orientation distance, properties and testers

Given two G-orientations $\vec{G}_1$ and $\vec{G}_2$, the distance between $\vec{G}_1$ and $\vec{G}_2$, denoted by $\Delta(\vec{G}_1, \vec{G}_2)$, is the number of edges in $E(G)$ having different directions in $\vec{G}_1$ and $\vec{G}_2$.

Given a graph G, a property $\mathcal{P}_G$ of G's orientations is a subset of all possible G-orientations. We say that an orientation $\vec{G}$ *satisfies* the property $\mathcal{P}_G$ if $\vec{G} \in \mathcal{P}_G$. The *distance* of $\vec{G}_1$ from the property $\mathcal{P}_G$ is defined by $\delta(\vec{G}_1, \mathcal{P}_G) = \min_{\vec{G}_2 \in \mathcal{P}_G} \frac{\Delta(\vec{G}_1, \vec{G}_2)}{|E(G)|}$. We say that $\vec{G}$ is ϵ-*far* from $\mathcal{P}_G$ if $\delta(\vec{G}, \mathcal{P}_G) \geq \epsilon$, and otherwise we say that $\vec{G}$ is ϵ-*close* to $\mathcal{P}_G$. We omit the subscript G when it is obvious from the context.

Definition 3.1. [(ϵ, q)**-tester**] Let G be a fixed undirected graph and let P be a property of G's orientations. An (ϵ, q)-tester T for the property P is a randomized algorithm, that for any $\vec{G}$ that is given via oracle access to the orientations of its edges operates as follows.

- The algorithm T makes at most q orientation queries to $\vec{G}$ (where on a query $e \in E(G)$ it receives as the answer the orientation of e in $\vec{G}$).
- If $\vec{G} \in P$, then T accepts it with probability 1 (here we define only *one-sided error* testers, since our testing algorithm will be one).
- If $\vec{G}$ is ϵ-far from P, then T rejects it with probability at least 2/3.

The query complexity of an (ϵ, q)-tester T is the maximal number of queries q that T makes on any input. We say that a property P is testable if for every $\epsilon > 0$ it has an $(\epsilon, q(\epsilon))$-test, where $q(\epsilon)$ is a function depending only on ϵ (and independent of the graph size n).

3.2.3 Connectivity Programs and Branching Programs

Our first sequence of reductions converts general *st*-connectivity instances to well structured bounded-width *st*-connectivity instances, as formalized in the next definition.

Definition 3.2 (*st*-Connectivity Program)**.** An *st-Connectivity Program* of width w and length m over n vertices (or $CP(w, m, n)$ in short), is a tuple $\langle G, \mathcal{L}\rangle$, where G is an *undirected* graph with n edges and $\mathcal{L}$ is a partition of G's vertices into layers $L_0, \ldots, L_m$. There are two special vertices in G: $s \in L_0$ and $t \in L_m$, and the edges of G are going only between vertices in consecutive layers, or between the vertices of the same layer, i.e. for each $e = e\{u, v\} \in E(G)$ there exists $i \in [m]$ such that $u \in L_{i-1}$ and $v \in L_i$, or $u, v \in L_i$. The partition $\mathcal{L}$ induces a partition $E_1, \ldots, E_m$ of $E(G)$, where E_i is the set of edges that have both vertices in L_i, or one vertex in L_{i-1} and another in L_i. In this partition of the edges the following holds: $\max_i\{|E_i|\} \leq w$.

Any orientation of G's edges (that maps every edge $e\{u, v\} \in E(G)$ to either $e(u, v)$ or $e(v, u)$) defines a directed graph $\vec{G}$ in the natural way. An *st*-connectivity program $C = \langle G, \mathcal{L}, \rangle$ defines a property P_C of G's orientations in the following way: $\vec{G} \in P_C$ if and only if in the directed graph $\vec{G}$ there is a directed path from s to t.

Next we define branching programs. These are the objects to which we can apply the testing algorithm of [New02].

Definition 3.3 (Branching Program)**.** A *Read Once Branching Program* of width w over an input of n bits (or $BP(w, n)$ in short), is a tuple $\langle G, \mathcal{L}, X\rangle$, where G is a *directed* graph with 0/1-labeled edges, $\mathcal{L}$ is a partition of G's vertices into layers $L_0, \ldots, L_n$ such that $\max_i\{|L_i|\} \leq w$, and $X = \{x_0, \ldots, x_{n-1}\}$ is a set of n Boolean variables. In the graph G there is one special vertex s belonging to L_0, and a subset $T \subset L_n$ of *accepting* vertices. The edges of G are going only between vertices in consecutive layers, i.e. for each $e = e(u, v) \in E(G)$ there is $i \in [n]$ such that $u \in L_{i-1}$ and $v \in L_i$. Each vertex in G has at most two outgoing edges, one of which is labeled by '0' and the other is labeled by '1'. In addition, all edges between two consecutive layers are associated with one distinct member of $X = \{x_0, \ldots, x_{n-1}\}$. An assignment $\sigma : X \to \{0, 1\}$ to X defines a subgraph G_σ of G, which has the same vertex set as G, and for every layer L_{i-1} (whose outgoing edges are associated with the variable x_{j_i}), the subgraph G_σ has only the outgoing edges labeled by $\sigma(x_{j_i})$. A read once branching program $B = \langle G, \mathcal{L}, X\rangle$ defines a property $P_B \subset \{0,1\}^n$ in the following way: $\sigma \in P_B$ if and only if in the subgraph G_σ there is a directed path from the starting vertex s to any of the accepting vertices in T.

Branching programs that comply with the above definition can be tested by the algorithm of [New02]. However, as we will see in Section 3.5, the branching programs resulting by the reduction from our st-connectivity programs have a feature that they require reading more than one bit at a time to move between layers. The next definition describes these special branching programs formally.

Definition 3.4 (Clustered Branching Program)**.** A *c-clustered Read Once Branching Program* of width w and length m over an input of n bits, denoted $BP_c(w, m, n)$, is a tuple $\langle G, \mathcal{L}, X, \mathcal{I}\rangle$, where similarly to the previous definition, G is a directed graph with labeled edges (see below for the set of labels), $\mathcal{L} = (L_0, \dots, L_m)$ is a partition of G's vertices into m layers such that $\max_i\{|L_i|\} \le w$, and $X = \{x_0, \dots, x_{n-1}\}$ is a set of n Boolean variables. Here too, G has one special vertex s belonging to L_0, and a subset $T \subset L_n$ of *accepting* vertices. The additional element $\mathcal{I}$ is a partition $(I_1, \dots, I_m)$ of X into m components, such that $\max_i\{|I_i|\} \le c$.

All edges in between two consecutive layers L_{i-1} and L_i are associated with the component I_i of $\mathcal{I}$. Each vertex in L_{i-1} has $2^{|I_i|}$ outgoing edges, each of them labeled by a distinct $\alpha \in \{0,1\}^{|I_i|}$.

An assignment $\sigma : X \to \{0,1\}$ to X defines a subgraph G_σ of G, which has the same vertex set as G, and for every layer L_i (whose outgoing edges are associated with the component I_i), the subgraph G_σ has only the edges labeled by $\Big(\sigma(x_i)\Big)_{i \in I_i}$. A c-clustered read once branching program $B = \langle G, \mathcal{L}, X, \mathcal{I}\rangle$ defines a property $P_B \subset \{0,1\}^n$ in the following way: $\sigma \in P_B$ if and only if in the subgraph G_σ there is a directed path from the starting vertex s to one of the accepting vertices in T.

Observe that $BP(w, m)$ is equivalent to $BP_1(w, m, m)$.

3.3 The main result

For an undirected graph G and a pair $s, t \in V(G)$ of distinct vertices, let P_G^{st} be a set of G-orientations under which there is a directed path from s to t. Formally, $P_G^{st} = \{\vec{G} : \mathrm{dist}_{\vec{G}}(s,t) < \infty\}$.

Theorem 3.5. *The property P_G^{st} is testable. In particular, for any undirected graph G, two vertices $s, t \in V(G)$ and every $\epsilon > 0$, there is an (ϵ, q)-tester T for P_G^{st} with query complexity $q = 2^{2^{2^{O(\epsilon^{-1}\log^2(1/\epsilon))}}}$*

3.3.1 Proof overview

The main idea of the proof is to reduce the problem of testing st-connectivity in the orientation model to the problem of testing a Boolean function that is represented by a *small width read once branching program.* For the latter we have the result of [New02] asserting that each such Boolean function is testable.

Theorem 3.6 ([New02])**.** *Let $P \subseteq \{0,1\}^n$ be the language accepted by a read-once branching program of width w. Then testing P requires at most $\left(\frac{2^w}{\epsilon}\right)^{O(w)}$ queries.*

By the definition above of $BP(w,n)$, one could already notice that testing the acceptance of a branching program resembles testing st-connectivity, and that the two problems seem quite close. However, there are several significant differences, as noted here.

1. In branching programs, querying a single variable reveals all the edges of its associated layer, while in our case, we need to query each edge of the st-connectivity instance separately.

2. The length of the input in branching programs is the number of layers rather than the total number of edges.

3. The edges in branching program graphs are always directed from L_{i-1} to L_i for some $i \in [n]$. In our case, the graph is not layered, and a pair u, v of vertices might have any number of edges in both directions.

4. In branching programs the graphs have out-degree exactly 2, while an input graph of the *st*-connectivity problem might have vertices with unbounded out-degree.

5. The most significant difference is that the input graphs of the *st*-connectivity problem may have unbounded width. This means that the naive reduction to branching programs may result in an unbounded width BPs, which we cannot test with a constant number of queries.

We resolve these points in several steps. First, given an input graph G, we reduce it to a graph $G^{(1)}$ which has the following property: for every induced subgraph W of $G^{(1)}$, the diameter of W is larger than ϵ times the number of edges in W. Then we prove that $G^{(1)}$ can be layered such that most of its edges lie within bounded-width layers. In particular, the total number of edges in the "wide" layers is bounded by $\frac{\epsilon}{2}E(G^{(1)})$. For this we need a bounded expansion lemma which is stated and proven here for this purpose. Next we reduce $G^{(1)}$ to a graph $G^{(2)}$, which can be layered as above, but without wide layers at all. This in particular means that the number of edges in $G^{(2)}$ is of the order of the number of layers, similarly to bounded width branching programs. In addition, in this layering of $G^{(2)}$ the vertex s is in the first layer, and the vertex t is in the last layer. Then we reduce $G^{(2)}$ (which might be thought of as the *undirected* analogue of a bounded width branching program) to a clustered read once bounded width branching program. Finally we show that these clustered branching programs can be converted into non-clustered branching programs, to which we can apply the test from [New02].

3.4 Reducing general graphs to connectivity programs

In this section we prove our first step towards proving Theorem 3.5. We reduce the problem of testing st-connectivity in a general graph to the problem of testing st-connectivity on an st-Connectivity Program. First we define the notion of reducibility in our context, and then we describe a sequence of reductions that will eventually lead us to the problem of testing read once bounded width BPs.

3.4.1 Reducibility between st-connectivity instances

Let $\mathcal{G}^{st}$ denote the class of undirected graphs having two distinct vertices s and t, and let $G, G' \in \mathcal{G}^{st}$. We say that G is (ϵ, η)-*reducible* to G' if there is a function ρ that maps orientations of G to orientations of G' (from now on we denote by $\vec{G}'$ the orientation $\rho(\vec{G})$) such that the following holds.

- If $\vec{G} \in P^{st}_{G}$ then $\vec{G}' \in P^{st}_{G'}$
- If $\delta(\vec{G}, P^{st}_{G}) \geq \epsilon$ then $\delta(\vec{G}', P^{st}_{G'}) \geq \eta$
- Any orientation query to $\vec{G}'$ can be simulated by a single orientation query to $\vec{G}$.

We say that G is (ϵ)-*reducible* to G' if it is (ϵ, ϵ)-reducible to G'. Notice that whenever G is (ϵ, η)-reducible to G', any (η, q)-tester T' for $P^{st}_{G'}$ can be converted into an (ϵ, q)-tester T for P^{st}_{G}. Or in other words, (ϵ, q)-testing P^{st}_{G} is reducible to (η, q)-testing $P^{st}_{G'}$.

In the following section we introduce our first reduction, which is referred to as the reduction from G to $G^{(1)}$ in the proof overview.

3.4.2 Reduction to graphs having high-diameter subgraphs

An undirected graph G is called *ϵ-long* if for every subset $W \subset V(G)$, we have

$$\mathrm{diam}(G(W)) > \epsilon |E(W)|.$$

Lemma 3.7. *Any graph $G \in \mathcal{G}^{st}$ is (ϵ)-reducible to a graph $G' \in \mathcal{G}^{st}$ which is ϵ -long.*

We first define a general contraction operator for graphs, and then use it for the proof. Given a graph G let $W \subset V$ be a subset of its vertices, and let $C_1, \ldots, C_t$ be the vertex sets of the connected components of $G(W)$. Namely, for each $C_i \subset W$, the induced subgraph $G(C_i)$ of the underlying graph G is connected, and for any pair $u \in C_i, v \in C_j$ $(i \neq j)$ of vertices, $e\{u, v\} \notin E(G)$. We define a graph G/W as follows. The graph G/W has the vertex set $V/W = (V \setminus W) \cup \{c_1, \ldots, c_t\}$ and its edges are E/W which is

$$\Big\{e\{u, v\} : (u, v \in V \backslash W) \wedge (e \in E)\Big\} \cup \Big\{e\{c_i, v\} : (v \in V \backslash W) \wedge \exists_{u \in C_i}(e\{u, v\} \in E)\Big\}$$

Intuitively, in G/W we contract every connected component C_i of $G(W)$ into a single (new) vertex c_i without changing the rest of the graph. Note that such a contraction might create parallel edges, but loops are removed. Whenever $s \in C_i$ for some $i \in [t]$, we rename the vertex c_i by s, and similarly if $t \in C_j$ then we rename c_j by t. In the following a connected component containing both s and t will not be contracted, so we can assume that the distinguished vertices s and t remain in the new graph G/W. Given an orientation $\vec{G}$ of G we define the orientation $\vec{G}/W = \rho(\vec{G})$ of G/W as the orientation induced from $\vec{G}$ in the natural way (note that there are no "new" edges in G/W). We will now need the following lemma.

Lemma 3.8. *Let $W \subset V$ be a subset of G's vertices, such that* $\mathrm{diam}(G(W)) \leq \epsilon |E(W)|$. *Then G is (ϵ)-reducible to the graph G/W.*

Proof of Lemma 3.8. Fix an orientation $\vec{G}$ of G. It is clear that if $\vec{G} \in P_G^{st}$ then $\vec{G}/W \in P_{G/W}^{st}$. Now assume that $\delta(\vec{G}, P_G^{st}) \geq \epsilon$. Let d and d' denote $\delta(\vec{G}, P_G^{st}) \cdot |E(G)|$ and $\delta(\vec{G}/W, P_{G/W}^{st}) \cdot |E(G/W)|$ respectively. From the definition of the graph G/W it follows that $d' \geq d - \mathrm{diam}(G(W))$. This is true since any st-path in $\vec{G}/W$ can be extended to an st-path in $\vec{G}$ by reorienting at most $\mathrm{diam}(G(W))$ edges in W (by definition, $\mathrm{diam}(G(W))$ is an upper bound on the undirected distance from any "entry" vertex to any "exit" vertex in $G(W)$). From the condition on W we have $|E(\vec{G}/W)| = |E| - |E(W)| \leq |E| - \frac{\mathrm{diam}(G(W))}{\epsilon}$. Combining these two together we have

$$\delta(\vec{G}/W, P_{G/W}^{st}) = \frac{d'}{|E(\vec{G}/W)|} \geq \frac{d - \mathrm{diam}(G(W))}{|E| - \mathrm{diam}(G(W))/\epsilon} \geq \frac{d - \mathrm{diam}(G(W))}{d/\epsilon - \mathrm{diam}(G(W))/\epsilon} = \epsilon$$

In addition, it is clear that we can simulate each query to $\vec{G}/W$ by making at most one query to $\vec{G}$. □

As a special case we have the following lemma.

Lemma 3.9. *Let $v \in V$ be a vertex of $\vec{G}$, such that* $\deg(v) \geq 2/\epsilon$. *Then G is (ϵ)-reducible to the graph $G/N_1(v)$.*

Now the proof of Lemma 3.7 follows by applying this contraction (iteratively) for each "bad" subgraph W, until eventually we get a graph G' in which all vertex subsets W satisfy $\mathrm{diam}(G(W)) > \frac{|E(W)|}{\epsilon}$. If in some stage we have both s and t contained in the contracted subgraph W, then we terminate the whole algorithm by accepting (since in this case all orientations are ϵ-close to being st-connected). Note that this process may result in several different graphs (depending on choices of the set W in each iteration), but we are only interested in one (arbitrary) graph G'. □

3.4.3 Properties of ϵ-long graphs

Next we show that an ϵ-long graph G can be "layered" so that the total number of edges in the "wide" layers is bounded by $\frac{\epsilon}{2}E(G)$. We first define graph layering.

Definition 3.10 (Graph layering and width)**.** Given a graph G and a vertex $s \in V(G)$, let m denote the maximal (undirected) distance from s to any other vertex in G. We define a *layering* $\mathcal{L} = (L_0, L_1, \ldots, L_m)$ of G's vertices as follows. $L_0 = \{s\}$ and for every $i > 0$, $L_i = N_i(s) \setminus N_{i-1}(s)$. Namely L_i is the set of vertices which are at (undirected) distance exactly i from s. Note that for every edge $e\{u, v\}$ of G either both u and v are in the same layer, or $u \in L_{i-1}$ and $v \in L_i$ for some $i \in [m]$.

We also denote by $E_i^{\mathcal{L}}$ the subset of G's edges that either have one vertex in L_i and the other in L_{i-1}, or edges that have both vertices in L_i. Alternatively, $E_i^{\mathcal{L}} = E(L_i \cup L_{i-1}) \setminus E(L_{i-1})$. We refer to the sets L_i and $E_i^{\mathcal{L}}$ as *vertex-layers* and *edge-layers* respectively. We might omit the superscript $\mathcal{L}$ from the edge-layers notation when it is clear from the context.

The *vertex-width* of a layering $\mathcal{L}$ is $\max_i\{|L_i|\}$, and the *edge-width* of $\mathcal{L}$ is $\max_i\{|E_i|\}$.

3.4.4 Bounding the number of edges within wide edge-layers

The following lemma states that in a layering of an ϵ-long graph most of the edges are captured in edge-layers of bounded width.

Lemma 3.11. *Consider the layering $\mathcal{L}$ of an ϵ-long graph G as defined above, and let $I = \{i : |E_i| > 2^{100/\epsilon^2}/\epsilon\}$ be the set of indices of the wide edge-layers. Then the following holds: $\sum_{i \in I} |E_i| \leq \frac{\epsilon}{2}|E|$.*

Before proving Lemma 3.11 we prove an auxiliary bounded expansion lemma. Denote by $\mathcal{A} = \langle a_0, a_1, \ldots, a_m \rangle$ a sequence of integers, where $a_0 = 1$ and for every $i \geq 1$, $a_i = |E_i|$.

Definition 3.12 (ϵ-good sequence)**.** Let $0 < \epsilon < 1$ be a positive constant. A sequence $1 = a_0, a_1, \ldots, a_m$ of positive natural numbers is *ϵ-good* if for every $0 \leq$

$k < m$ and $\ell \in [m-k]$ we have that

$$\sum_{i=k+1}^{k+\ell} a_i \leq \frac{\ell \cdot a_k}{\epsilon}.$$

Claim 3.13. *Let G be a graph in which for any induced subgraph W we have $|E(W)| < \mathrm{diam}(W)/\epsilon$. Then the sequence $\mathcal{A} = \langle a_0, a_1, \ldots, a_m \rangle$ defined above is $\epsilon/4$-good.*

Proof of Claim 3.13. Let us assume the contrary of the claim. Let k and l be such that

$$\sum_{i=k+1}^{k+\ell} a_i > \frac{4\ell \cdot a_k}{\epsilon}$$

Consider the subgraph W defined by the vertices $\cup_{i=k}^{k+\ell} L_i$ and the edges $\cup_{i=k+1}^{k+\ell} E_i$. Now the number of edges in W is clearly $\sum_{i=k+1}^{k+\ell} a_i$. For each vertex v in L_k consider the neighborhood of distance $\ell+1$ from v, $N_{\ell+1}(v)$, and denote the subgraph it spans by W_v. Notice that each W_v is of diameter at most $2(\ell+1) \leq 4\ell$, and that the union of the edge sets of the W_v includes the whole edge set of W, so we have $\sum_{v \in L_k} |E(W_v)| \geq \sum_{i=k+1}^{k+\ell} a_i$. The number of vertices in L_k is at most a_k, so by the pigeonhole principle we know that at least one of the vertices v in L_k has at least $\frac{\sum_{i=k+1}^{k+\ell} a_i}{a_k}$ edges in W_v. By our assumption on $\sum_{i=k+1}^{k+\ell} a_i$ we have $|E(W_v)| > \frac{4\ell}{\epsilon} \geq \mathrm{diam}(W_v)/\epsilon$, a contradiction. □

Proof of Lemma 3.11. The proof follows directly from Claim 3.13 and the following Bounded Expansion Lemma that we prove in Section 3.8. □

Lemma 3.14 (The Bounded Expansion Lemma)**.** *Let $\langle a_0, \ldots, a_m \rangle$ be an ϵ-good sequence and let $B = \{i \mid a_i > 2^{5/\epsilon^2}/\epsilon\}$. Then*

$$\sum_{i \in B} a_i \leq \epsilon \sum_{i=1}^{m} a_i.$$

3.4.5 Reduction to bounded width graphs

In this section we prove that ϵ-long graphs can be reduced to graphs that have bounded width. In terms of the proof overview, we are going to reduce the graph $G^{(1)}$ to the graph $G^{(2)}$.

Let $G = (V, E) \in \mathcal{G}^{st}$, and let $\mathcal{L} = (L_0, L_1, \ldots, L_m)$ be the layering of G as above. We call an edge-layer E_i *wide* if $|E_i| > \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$. Let $\mathcal{W}$ be the set of all wide edge-layers.

Lemma 3.15. *If $G \in \mathcal{G}^{st}$ satisfies $\sum_{E_i \in \mathcal{W}} |E_i| \leq \frac{\epsilon}{2}|E|$ then G is $(\epsilon, \epsilon/2)$-reducible to a graph G' which has no wide edge-layers at all.*

Proof of Lemma 3.15. Recall the definition of G/W (a graph in which a subset of vertices is contracted) from Section 3.4.2. Let E_i be a wide edge-layer. For every edge $e = e\{u, v\} \in E_i$ define $W_e = \{u, v\}$. We iteratively contract the subgraphs W_e in G, for all edges in all wide edge-layers. Denote the final graph as $G' = (V', E')$.

We claim that after this process G' has no wide edge-layers at all. Formally let $p(i)$ denote the size of the set $\{j | j \leq i \text{ and } E_j \text{ is wide}\}|$. Namely $p(i)$ is number of wide edge-layers in G preceding the vertex-layer L_i. Now we have the following claim:

Observation 3.16. *Let $v \in L_i$ be a vertex in the i'th vertex-layer of G. Then v's representing vertex v' in G' is in the vertex-layer $L'_{i-p(i)}$ of G'.*

Proof. The proof follows by induction on i. ☐

The mapping μ sending i to $i' = (i-p(i))$ is monotone non-decreasing, and onto the number of layers in G'. For an index i' of a vertex-layer in G', let $\ell(i')$ denote the largest index in the set $\mu^{-1}(i')$. Recall that E'_i is the set of edges from layer $L'_{i'-1}$ to layer $L'_{i'}$ and within layer $L'_{i'}$ in G'. Note that these edges are exactly the edges that correspond to the edges between layers $L_{\ell(i')}$ and $L_{\ell(i')-1}$ and within $L_{\ell(i')}$ in G. Thus assuming towards a contradiction that E'_i is wide in G', means that $E_{\ell(i)}$ is wide in G . But this is not possible, as $E_{\ell(i)}$ is in the set of edges that were not contracted. As a conclusion, G' cannot have any wide edge-layers.

It is clear that any orientation query to $\vec{G'}$ can be simulated by a single orientation query to $\vec{G}$, and since all we did was contracting edges, if there is a path from s to t in $\vec{G}$ then there is a path from s to t in $\vec{G'}$. Now we only need to show is that the second condition of reducibility holds.

We now show that by changing the direction of at most $\frac{\epsilon}{2}|E'|$ edges we can have a path from s to t in $\vec{G}$ if there was one in $\vec{G'}$. We can always assume that the path in $\vec{G'}$ is simple, i.e. it passes through each vertex at most once. Now each vertex in G' corresponds to a connected subgraph in G. We call a non-trivial subgraph of G *shrunk* if all its vertices, and only them, are represented by a single vertex in G'. Clearly we can extend an st-path in G' to an st-path in G by only reorienting the edges in the shrunk subgraphs of G. We only contracted edges in the wide edge-layers, so the total number of edges in the shrunk components is at most the total number of edges in the wide edge-layers. By the property of G we have that only $\frac{\epsilon}{2}|E|$ of the edges lie in the shrunk subgraphs of G. If by changing the orientation of only $\frac{\epsilon}{2}|E'|$ edges in $\vec{G'}$ we get a path from s to t in $\vec{G'}$, then by changing only $\frac{\epsilon}{2}|E'| + \frac{\epsilon}{2}|E| \leq \epsilon|E|$ edges in $\vec{G}$ we can get a path from s to t in $\vec{G}$, and the second reducibility condition is proved. □

3.4.6 Reducing bounded width graphs to st-connectivity programs

So far we reduced the original graph G to a graph G' which has a layering $\mathcal{L} = (L_0, L_1, \ldots, L_m)$ of edge-width at most $w = \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$, and in which the source vertex s belongs to layer L_0. The remaining difference between G' and an st-connectivity program is that in G' the target vertex t might not be in the last vertex-layer L_m. The following lemma states that we can overcome this difference by another reduction.

Lemma 3.17. *Let G' be a graph as described above. Then G' is $(\epsilon, \epsilon/2)$-reducible to an st-connectivity program S of width at most $w+1$.*

Proof of Lemma 3.17. Let L_r be the layer to which t belongs in the layering $\mathcal{L}$ of the graph G'. Let $P = \{p_1, p_2, \ldots, p_{m-r}\}$ be a set of $m-r$ new vertices. We define S as follows.

- $V(S) = V(G') \cup P$
- $E(S) = E(G') \cup \left(\bigcup_{i=1}^{m-r-1} \{e_i(p_i, p_{i+1})\}\right) \cup \{e_t(t, p_1)\}$

Any orientation $\vec{G}'$ of G' induces a natural orientation $\vec{S}$ of S; all edges $e \in E(S)$ that are also in $E(G')$ have the same orientation as in $\vec{G}'$, while the orientation of the new edges were defined explicitly above. We also rename t to p_0 and rename p_{m-r} to t in S. Basically we have added a sufficiently long path from the original target vertex to the new target vertex to get S. Now it is easy to verify that G' is indeed $(\epsilon, \epsilon/2)$-reducible to S (assuming that G has at least $1/\epsilon$ edges), and that the width of S is as required. □

3.5 Reducing *st*-connectivity programs to branching programs

We now show how to reduce an *st*-connectivity program to a clustered branching program (recall Definition 3.2 and Definition 3.4). First observe that we can assume without loss of generality that if an *st*-connectivity program has edge-width w, then its vertex-width is at most $2w$ (since removing vertices of degree 0 essentially does not affect the *st*-connectivity program, and a vertex in L_i with edges only between it and L_{i+1} can be safely moved to L_{i+1}).

Before moving to the formal description of the reduction, we start with a short intuitive overview. A branching program corresponds to a (space bounded) computation that moves from the start vertex s, which represents no information about the input at all, and proceeds (via the edges that are consistent with the input bits) along a path to one of the final vertices. Every vertex of the branching program represents some additional information gathered by reading more and more pieces of the input. Thus, the best way to understand the reduction is to understand the implicit meaning of each vertex in the constructed branching program.

Given a graph G of a bounded width st-connectivity program, and its layering $L_0, L_1, \ldots L_m$, we construct a graph G' (with layering $L'_0, L'_1, \ldots L'_m$) for the bounded-width branching program. The graph G' has the same number of layers as G. Each level L'_i in G' will represent the conditional connectivity of the vertices in the subgraph $G_i = G(\bigcup_{j=0}^{i} L_i)$ of G. To be specific, the knowledge we want to store in a typical vertex at layer L'_i of G' is the following.

- for every $u \in L_i$ whether it is reachable from s in G_i.
- for every $v, u \in L_i$ whether v is reachable from u in G_i.

Hence, the amount of information we store in each node $x \in L'_i$ has at most $2w + (2w)^2$ many bits, and so there will be at most 4^{2w^2+w} vertices in each L'_i, meaning that the graph G' of the branching program is of bounded width as well.

Lemma 3.18. *Let $\epsilon > 0$ be a positive constant. Given a $CP(w, m, n)$ instance $C = \langle G, \mathcal{L} \rangle$, we can construct a $BP_w(4^{2w^2+w}, m, n)$ instance $B = \langle G', \mathcal{L}', X', \mathcal{I}' \rangle$ and a mapping ρ from G-orientations to assignments on X such that the following holds,*

- *if $\vec{G}$ satisfies P_C then $\sigma = \rho(\vec{G})$ satisfies P_B.*
- *if $\vec{G}$ is ϵ-far from satisfying P_C then $\sigma = \rho(\vec{G})$ is ϵ-far from satisfying P_B.*
- *any assignment query to σ can be simulated using a single orientation query to $\vec{G}$.*

Proof. First we describe the construction, and then show that it satisfies the requirements above.

The vertices of G': We fix i and show, based on the layer L_i of G, how to construct the corresponding layer L'_i of G'. Each vertex in L'_i corresponds to a possible value of a pair $(S_i,\, R_i)$ of relations. The first relation $S_i \subset L_i$ is a unary relation (predicate) over L_i indicating for each $v \in L_i$ whether there is a directed path from s to v in the subgraph of G induced on $\bigcup_{j=0}^{i} L_j$. The second relation $R_i \subset L_i \times L_i$ is a binary relation over L_i that for every ordered pair (u, v) of vertices in L_i indicates whether there is a directed path from u to v in the subgraph of G induced on $\bigcup_{j=0}^{i} L_j$ (the path can be of length 0, meaning that the R_i's are reflexive). Notice that $|L'_i| = 2^{|L_i|^2+|L_i|} \leq 4^{2w^2+w}$ for all i.

The edges of G': Now we construct the edges of G'. Recall that E'_{i+1} denotes the set of (labeled) edges having one vertex in L'_{i+1} and the other in L'_i. Fix i and a vertex $v \in L'_i$. Let (S, R) be the pair of relations that correspond to v. Let S'' be the set $L_{i+1} \cap \Gamma_{out}(S)$, namely the neighbors of vertices from S that are in L_{i+1}, and set $R'' = \{(v, v) : v \in L_{i+1}\} \cup \{(u, v) : \big(u, v \in L_{i+1}\big) \wedge \big(v \in \Gamma_{out}(u)\big)\} \;\cup\; \{(u, v) : \big(u, v \in L_{i+1}\big) \wedge \Big(\big(\Gamma_{out}(u) \times \Gamma_{in}(v)\big) \cap R \neq \emptyset\Big)\}$. Now define R' as the transitive closure of R'', and set $S' = S'' \cup \{v \in L_{i+1} : \exists_{u \in S''}(u, v) \in R'\}$. Let $v' \in L'_{i+1}$ be the vertex corresponding to the pair (S', R') that we defined above. Then the edges of E'_{i+1} are given by all such pairs of vertices (v, v').

The variables in X': Each variable $x'_i \in X'$ is associated with an edge $e_i \in E(G)$. This association is actually the mapping ρ above, i.e. every orientation $\vec{G}$ of G defines an assignment σ on X'.

The partition $\mathcal{I}'$ of X': Recall that E_i denotes the set of edges of G having either one vertex in L_{i-1} and the other in L_i, or both vertices in L_i. The partition $\mathcal{I}'$ of X' is induced by the partition $\mathcal{L}$ of $V(G)$. Namely, the component I_i of $\mathcal{I}$ contains the set of variables in X' that are associated with edges in E_i. Thus w is also a bound on the sizes of the components in $\mathcal{I}$.

The set $T' \subset L'_m$ of accepting vertices: The set T' is simply the subset of vertices in L'_m whose corresponding relation S contains the target vertex t of G.

Note that each value of a variable in X' corresponds exactly to an orientation of an edge in G. This immediately implies the third assertion in the statement of the lemma. Distances between inputs are clearly preserved, so to prove the other assertions it is enough to show that the branching program accepts exactly those assignments that correspond to orientations accepted by the connectivity program. It is straightforward (and left to the reader) to see that the vertex reached in each L'_i indeed fits the description of the relations R and S, and so an assignment is accepted if and only if it corresponds to a connecting orientation. □

The branching programs resulting from the above reduction have a feature that they require reading more than one bit at a time to move between layers. Specifically, they conform to Definition 3.4. The result in [New02], however, deals with standard branching programs (see Definition 3.3), which in relation to the above are a special case in which essentially $m = n$ and all the I_i's have size 1. Going from here to a standard (non-clustered) branching program (in which the edges between two layers depend on just one Boolean variable) is easy. The requires lemma is proved in the next section.

3.6 Converting clustered branching programs to non-clustered ones

Lemma 3.19. *Any $BP_c(w, m, n)$ instance can be converted to a $BP(w2^c, n)$ instance accepting the very same language.*

Proof. Throughout the proof it is convenient to refer to the vertices of a layer L_i in the clustered branching program as a set of *states*, and to the edges between L_{i-1} and L_i as a *transition function* $f_i : L_{i-1} \times \{0,1\}^{|I_i|} \to L_i$. Namely, $f_i(v, b_1, \ldots, b_{|I_i|})$ is equal to the vertex w in L_i so that (v, w) is an edge labeled with $(b_1, \ldots, b_{|I_i|})$. We shall use an analogue notation for the transition functions in the constructed branching program, $f'_i : L'_{i-1} \times \{0,1\} \to L'_i$. The basic idea of the proof is that instead of reading the entire cluster I_i at once, we read it bit by bit, and use additional states in each layer to store the bits that we have read so far. We need to read at most c bits before we can use the original transition functions, which causes the blowup of up to 2^c in the number of states in each layer. We define the new layers $\{s\} = L'_0, \ldots, L'_n$ of the program inductively.

First we define $L_0 = L'_0 = \{s\}$. Now, assuming that we have already converted $L_0, \ldots, L_{i-1}$ into $L'_0, \ldots, L'_j$ such that $L'_j = L_{i-1}$, and calculated the corresponding transition functions $f_k : L'_{k-1} \times \{0,1\} \to L'_k$, we show how to convert L_i into layers $L'_{j+1}, \ldots, L'_{j+|I_i|}$ so that $L'_{j+|I_i|} = L_i$.

For $0 < k < |I_i|$ we set $L'_{j+k} = L_{i-1} \times \{0,1\}^k$, and set $L'_{j+|I_i|} = L_i$. Each layer L'_{j+k} will be associated with the bit corresponding to the k'th member of I_i, which to reduce indexes we shall re-label as y_{j+k}. In the following we denote members of a cross product $A \times B$ as tuples (a, b) with $a \in A$ and $b \in B$. The transition functions $f'_{j+1}, \ldots, f'_{j+|I_i|}$ are set as follows.

- For $k = 1$ we set
$$f'_{j+1}(v, y_{j+1}) = (v, y_{j+1}),$$
that is the transition is to the tuple resulting from pairing v with the bit y_{j+1}

- Accordingly, for $1 < k < |I_i|$ we set
$$f'_{j+k}(v, b_1, \ldots, b_{k-1}) = (v, b_1, \ldots, b_{k-1}, y_{j+k}),$$
i.e. we concatenate the value of y_{j+k} to the previously collected values.

- Finally, for $k = |I_i|$ we set
$$f'_{j+|I_i|}(v, b_1, \ldots, b_{|I_i|-1}) = f_i(v, b_1, \ldots, b_{|I_i|-1}, y_{j+|I_i|}),$$
i.e. we employ the function f_i on all the collected bit values including $y_{j+|I_i|}$.

The accepting subset T' of $L'_n = L_m$ remains the same as the original T. It is now not hard to see that both programs accept the exact same language over $x_1, \ldots, x_n$. □

The above means that the algorithm of [New02] can be employed over the new width $w2^c$ program with the same input.

3.7 Wrapping up – Proof of the Main Theorem

We started with a graph G and wanted to construct an (ϵ, q)-test for st-connectivity of orientations of G. In Section 3.4.1, Section 3.4.2 and Section 3.4.3 we constructed a graph G_1 such that if we have an (ϵ, q)-test for st-connectivity in G_1, then we have an (ϵ, q)-test for G. Additionally G_1 has the property that most of the edge-layers in G_1 are of size at most $w = \frac{1}{\epsilon} \cdot 2^{100/\epsilon^2}$. Then in Section 3.4.5 we constructed a graph G_2 such that if we have an $(\frac{\epsilon}{2}, q)$-test for st-connectivity in G_2 then we have an (ϵ, q)-test for G_1 and hence we have one for G. Moreover G_2 has all its edge-layers of size at most w. Finally in Section 3.4.6 we built a graph G_3 which has all its edge-layers of width at most $w + 1$, and in addition, the vertices s and t are in the first and the last vertex-layers of G_3 respectively. We also showed that having an $(\frac{\epsilon}{4}, q)$-test for st-connectivity in G_3 implies an $(\frac{\epsilon}{2}, q)$-test for G_2, and hence an (ϵ, q)-test for G. This ends the first part of the proof, which reduces general graphs to st-connectivity programs.

Then in Section 3.5 from G_3 we constructed a read once $(w + 1)$-clustered Branching Program that has width $4^{2(w+1)^2+w+1}$ so that an $(\frac{\epsilon}{4}, q)$-test for this BP gives an $(\frac{\epsilon}{4}, q)$-test for st-connectivity in G_3. Then we converted the $(w + 1)$-clustered Branching Program to a non-clustered Branching Program which has width $w_1 = 4^{2(w+1)^2+(w+1)2^{(w+1)}}$. Once we have our read once bounded width branching program then by applying the algorithm of [New02] for testing branching programs we get an $(\frac{\epsilon}{4}, q)$-test with $q = (\frac{2^{w_1}}{\epsilon/4})^{O(w_1)}$ queries for our problem. Hence by combining all of the above, we get an (ϵ, q) testing algorithm for our original st-connectivity problem, where $q = (2/\epsilon)^{2^{O((1/\epsilon)\cdot 2^{(100/\epsilon^2)})}}$ $\square$

3.8 Bounded Expansion Lemma

Recall the definition of ϵ-good sequence.

Definition 3.20 (ϵ-good sequence). Let $0 < \epsilon < 1$ be a positive constant. A sequence $1 = a_0, a_1, \ldots, a_m$ of positive natural numbers is ϵ-*good* if for every $0 \leq$

$k < m$ and $\ell \in [m-k]$ we have that

$$\sum_{i=k+1}^{k+\ell} a_i \leq \frac{\ell \cdot a_k}{\epsilon}.$$

The following is the statement of Lemma 3.14 that we prove in this section.

Lemma 3.21 (The Bounded Expansion Lemma)**.** *Let $\langle a_0, \ldots, a_m \rangle$ be an ϵ-good sequence and let $B = \{i \mid a_i > (2/\epsilon)^{1/\epsilon}\}$. Then*

$$\sum_{i \in B} a_i \leq \epsilon \sum_{i=1}^{m} a_i.$$

Proof. Let A be the sum of $a_1, \ldots, a_m$. According to Claim 3.23, that we state and prove further on, we may assume that $a_0, \ldots, a_m$ are monotone non-decreasing as a function of their index without loss of generality.

Note that a monotone non-decreasing sequence is ϵ-good if and only if for all $k \geq 0$ we have

$$\sum_{i=k+1}^{m} a_i \leq \frac{(m-k) \cdot a_k}{\epsilon} \tag{3.1}$$

So from the above condition and the fact that $a_0 = 1$ we have that

$$\sum_{i=1}^{m} a_i = A \leq \frac{m}{\epsilon}$$

Let $\eta = n/(\epsilon A)$. Consider the sequence $\langle a'_0, a'_1, \ldots, a'm \rangle$ where $a'_0 = a_0 = 1$ and for all $i \neq 0$ we have $a'_i = \eta \cdot a_i$. This sequence also satisfies Condition 3.1 and assume that $B' = \{i \mid a'_i > (2/\epsilon)^{1/\epsilon}\}$ and

$$\sum_{i \in B'} a'_i \leq \epsilon \sum_{i=1}^{m} a'_i.$$

Since $\eta \geq 1$ we have B is a subset of B' and hence

$$\sum_{i \in B} a_i \leq \sum_{i \in B'} a_i = \frac{1}{\eta} sum_{i \in B'} a'_i \leq \frac{1}{\eta} \epsilon \sum_{i=1}^{m} a'_i = \epsilon \sum_{i=1}^{m} a_i$$

So it is sufficient to prove the Bounded Expansion Lemma for the sequence $\langle a' \rangle$. So we can assume $\epsilon \cdot A = m$. Let C be defined as

$$C = min\{j | \sum_{i=j+1}^{m} a_i \leq m\} \tag{3.2}$$

Note that the Bounded Expansion Lemma is true if and only if the following inequality holds

$$a_{CUT-OFF} \leq \left(\frac{1}{\epsilon}\right)^{1/\epsilon} \tag{3.3}$$

To prove the Inequality 3.3 we will need the following claim that we prove further on.

Claim 3.22. *If $\langle a_0, \ldots, a_m \rangle$ is a non-decreasing ϵ-good sequence, with $a_0 = 1$, and if C is as in Equation 3.2 then*

$$a_C \leq \left(\frac{n}{m - C}\right)^{1-\epsilon}$$

Since the sequence is an ϵ-good increasing sequence we know from Equation 3.1 that

$$(m - C)\left(\frac{a_C}{\epsilon}\right) \geq \sum_{i=C+1}^{m} a_i \tag{3.4}$$

Now since C is defined as in Equation 3.2 we have that

$$\sum_{i=C}^{m} a_i > \epsilon A$$

From this and the fact that the sequence is increasing we obtain the following

$$\sum_{i=C+1}^{m} a_i > \epsilon A \left(\frac{m-C-1}{m-C} \right) \tag{3.5}$$

Combining Inequalities 3.4 and 3.5 and Claim 3.22 we obtain

$$\epsilon A \left(\frac{(m-C-1)}{(m-C)^2} \right) \leq \left(\frac{1}{\epsilon} \right) \left(\frac{\epsilon A}{m-C} \right)^{1-\epsilon}$$

Simplifying it we get

$$(n-C) \geq n \left(\frac{\epsilon^2 A}{2n} \right)^{1/\epsilon} = n \left(\frac{\epsilon}{2} \right)^{1/\epsilon}$$

Plugging it in Claim 3.22 we obtain that

$$a_C \leq \left(\frac{2}{\epsilon} \right)^{1/\epsilon}$$

□

Claim 3.23. *If $\langle a_0, \ldots, a_m \rangle$ is an ϵ-good sequence, then the sequence $\langle b_0, \ldots, b_m \rangle$ obtained by sorting $a_0, \ldots, a_m$ is also ϵ-good.*

Proof. As the b_i's are monotone increasing as a function of their index, in order to show that they are an ϵ-good sequence we only need to show that for any $k \in [m]$ we have

$$\frac{1}{m-k} \sum_{i=k+1}^{m} b_i \leq \frac{b_k}{\epsilon}$$

(as for monotone sequences the average only decreases if a subsequence is chopped off from the right).

Fix $k \leq m$ and consider the average $\frac{1}{m-k} \sum_{i=k+1}^{m} b_i$. We may assume that each b_i is a renaming of some a_j in the original sequence. Thus the subsequence

$B = \langle b_{k+1}, \ldots, b_m \rangle$ corresponds to members in the original subsequence:

$$\langle a_{i_1+1}, \ldots, a_{i_1+j_1} \rangle, \langle a_{i_2+1}, \ldots, a_{i_2+j_2} \rangle, \ldots, \langle a_{i_t+1}, \ldots, a_{i_t+j_t} \rangle$$

where each subsequence $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ is a maximal contiguous subsequence in the original sequence whose members were renamed to members of B, and hence their value is at least b_k (as all members in B are such).

On the other hand, the values of $a_{i_1}, a_{i_2}, \ldots, a_{i_t}$ are all bounded by b_k as their renaming does not put them in B. Since $\langle a_1, \ldots, a_m \rangle$ is ϵ-good, this means that for every $1 \leq r \leq t$, the average of the subsequence $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ is at most b_k/ϵ. Note that it is safe to assume that b_0 is the renaming of a_0 (which for a good sequence is equal to the minimum 1) and hence $i_1 \geq 0$. Finally, as the average of the subsequence B is clearly a weighted average of the averages of the $\langle a_{i_r+1}, \ldots a_{i_r+j_r} \rangle$ subsequences, it is bounded by b_k/ϵ as well. □

Proof of Claim 3.22. We define a new sequence $\langle b_0, b_1, \ldots, b_m \rangle$ where for all $i > C$ we have $b_i = a_i$. For all other we define recursively from C to 0 as

$$b_j = \frac{\epsilon}{m-j} \sum_{i=j+1}^{m} b_i \tag{3.6}$$

Note that for all i we have $b_i \leq a_i \leq b_i/b_0$. So now we will give an upper bound on b_C/b_0 and this will imply an upper bound on a_C. The nice recursive pattern of the sequence $\langle b \rangle$ (Condition 3.6) will help us derive the upper bound.

From the definition of the sequence $\langle b \rangle$ (Condition 3.6) we obtain

$$b_j = \frac{(m-j+1)}{(m-j+\epsilon)} b_{j-1}$$

So we can write

$$b_C = \frac{m}{(m - C + \epsilon)\Pi_{i=1}^{C-1}(1 + \epsilon/(m - i))} b_0 \tag{3.7}$$

Now let $f(i) = (1 + \epsilon/(m - i))$. We want a lower bound on $\Pi_{i=1}^{C-1} f(i)$. Note that $f(i)$ is an increasing function. So we have

$$\log(\Pi_{i=1}^{C-1} f(i)) = \sum_{i=i}^{C-1} \log(f(i)) > \int_0^{C-1} \log(f(x))\, dx$$

From Claim 3.24 we have that $\log(\Pi_{i=1}^{C-1} f(i))$ is more than

$$\left[(C-1)\log(1 + \tfrac{\epsilon}{m-C+1}) - \epsilon \log\left(\tfrac{m-C+1}{\epsilon} + 1\right) - m \log\left(\tfrac{1}{\epsilon} + \tfrac{1}{m-C+1}\right)\right] - \left[-\epsilon \log\left(\tfrac{m}{\epsilon} + 1\right) - m \log\left(\tfrac{1}{\epsilon} + \tfrac{1}{m}\right)\right]$$

After doing the algebra we obtain, $\Pi_{i=1}^{C-1} f(i)$ is more than

$$\left(\frac{m + \epsilon}{(m - C + 1) + \epsilon}\right)^{\epsilon} \left(1 + \frac{\epsilon}{m}\right)^{m} \left(1 + \frac{\epsilon}{(m - C + 1)}\right)^{(C-1-m)}$$

which is greater than

$$\left(\frac{m}{(m - C + \epsilon}\right)^{\epsilon}$$

So plugging it back to Equation 3.7 we obtain our desired upper bound on b_C/b_0,

$$b_C < \left(\frac{m}{m - C}\right)^{1-\epsilon} b_0$$

$\square$

Claim 3.24. *If* $f(x) = (1 + \epsilon/(m - x))$, *then*

$$\int \log(f(x))\, dx = \left(x \log(f(x)) - \epsilon \log\left(\frac{(m - x)}{\epsilon} + 1\right) - nlog\left(\frac{1}{\epsilon} + \frac{1}{m - x}\right)\right)$$

Proof of Claim 3.24. First doing integration by parts

$$\begin{aligned}\int \log(1+\tfrac{\epsilon}{(m-x)}) &= x\log(1-\tfrac{\epsilon}{(m-x)}) - \int \tfrac{\epsilon x\,dx}{(m-x+\epsilon)(m-x)}\\ &= xf(x) + \int \tfrac{\epsilon(m-x)\,dx}{(m-x+\epsilon)(m-x)} - m\int \tfrac{\epsilon dx}{(m-x+\epsilon)(m-x)}\\ &= xf(x) - \epsilon\log(\tfrac{(m-x)}{\epsilon}+1) - m\int \tfrac{\epsilon dx}{(m-x+\epsilon)(m-x)}\end{aligned}$$

Now let us consider the $\int \frac{\epsilon dx}{(m-x+\epsilon)(m-x)}$. If we substitute $y=(m-x)$, then $dx=-dy$. So we obtain

$$\int \frac{\epsilon dx}{(m-x+\epsilon)(m-x)} = -\int \frac{\epsilon dy}{(y+\epsilon)y}$$

Now if we substitute $z=1/y$ we get $dz=-dy/y^2$. Hence,

$$-\int \frac{\epsilon dy}{(y+\epsilon)y} = \int \frac{\epsilon dz}{1+\epsilon z} = \log(\frac{1}{\epsilon}+z) = \log(\frac{1}{\epsilon}+\frac{1}{y}) = \log(\frac{1}{\epsilon}+\frac{1}{m-x})$$

So we finally have,

$$\int \log(f(x))\,dx = xf(x) - \epsilon\log(\frac{(m-x)}{\epsilon}+1) - m\log(\frac{1}{\epsilon}+\frac{1}{m-x})$$

□

CHAPTER 4
TESTING EQUIVALENCE UNDER A TRANSITIVE GROUP ACTION

4.1 Introduction

The immediate motivation of our work in this chapter comes from papers by Fischer [Fis05] and Fischer and Matsliah [FM06] who consider the Graph Isomorphism problem in the property testing model. Here two graphs are given as inputs and we have to test whether they are isomorphic or "far" from being isomorphic.

In this chapter we consider a generalization of graph isomorphism. Let us fix a permutation group G acting on the set Ω. Given two input strings $x, y : \Omega \to \{0,1\}$, we say x is "G-isomorphic" to y if y is a a π-shift of x for some $\pi \in G$. We want to test the property "x is G-isomorphic to y," that is, we want to distinguish the case when x and y are G-isomorphic from the case when every string that is G-isomorphic to y is far from x. [Formal definitions are given in Section 4.2.]

Graph Isomorphism is a special case: we need to choose Ω to be the set of unordered pairs of the set V of vertices; and $G = \mathrm{Sym}^{(2)}(V)$ the induced action on Ω of $\mathrm{Sym}(V)$, the symmetric group acting on V (so $n = \binom{|V|}{2}$). We note that the induced symmetric group action on pairs is primitive (does not admit nontrivial invariant partitions of the permutation domain). This fact defines the direction in which we extend results on Graph Isomorphism. We note that by considering the induced symmetric group action on k-tuples, another primitive action, we also cover the case of k-uniform hyper-graphs. Here k need not be a constant. Various finite geometries also correspond to primitive groups, so G-isomorphism includes equivalence under geometric transformations (projective, orthogonal, symplectic, etc.).

Besides the fact that the case of primitive groups includes Graph Isomorphism and its immediate generalizations (hyper-graphs) as well as geometric equivalence, primitive permutation groups are significant because they form the "building blocks" of all permutations groups in the sense that a "structure tree" can be built of which the leaves constitute the permutation domain and the action of G extends to the tree in such a way that the action of the stabilizer of any node in the tree is primitive on the children of the node (cf. [BLS87]). This structure tree formalizes the natural divide-and-conquer approach successfully exploited in algorithm design [BL83, BLS87, Luk82].

In "property testing" we want to output 1 if the inputs are G-isomorphic and 0 if they are "far" from being G-isomorphic. The complexity is the number of queries made to the input. We consider two models depending on whether we have to query both x and y or we have to query only one of them (the other is known). We call the models query-2 and query-1, respectively. A property test can have 1-sided or 2-sided-error.

In this chapter we focus mainly on property testing of G-isomorphism when the group is primitive. Our main results are the tight bounds on the query complexity when we are allowed only 1-sided error, that is, the algorithm has to output 1 with probability 1 when the two inputs are G-isomorphic and we have to output 0 with high probability when the inputs are "far" from being isomorphic. The main results are the following.

Theorem 4.1. *[Tight bounds for primitive groups] If G is a primitive group then*

1. *The 1-sided-error query complexity for testing G-isomorphism in the query-2 model is $\widetilde{\Theta}(\sqrt{n \log |G|})$.*

2. *The 1-sided-error query complexity for testing G-isomorphism in the query-1 model is $\widetilde{\Theta}(\log |G|)$.*

Theorem 4.1 generalizes a result of Fischer and Matsliah [FM06] on Graph Isomorphism. The lower bound parts of this result is the main technical contribution of this paper and is proved in Section 4.3. For the lower bound proofs we crucially use a classification of primitive groups based on the O'Nan–Scott Theorem (see [Cam81]).

We also prove some upper and lower bounds for the other cases. But in most of these cases, a significant gap remains between the upper and lower bounds. We present these results in the appendix. The following is the list of results we prove in the appendix. The tilde in the asymptotic notation indicates polylog(n) factors.

Proposition 4.2. *[Upper bound]*

1. *The query-1 complexity of 1-sided and 2-sided error G-isomorphism testing is $O(1 + \log |G|)$.*

2. *The query-2 complexity of 1-sided and 2-sided error G-isomorphism testing is $O(\sqrt{n(1 + \log |G|)})$.*

In Table 1, we abbreviated the expression $1 + \log |G|$ to $\log |G|$ for better typography. The only case where this makes a difference is when $|G| = 1$ so the results as stated in the Table 1 assume $|G| \geq 2$.

Theorem 4.3. *[Lower bound] Let G be a transitive group of order $2^{O(n^{1-\epsilon})}$. Then the 2-sided-error query-1 complexity of the property testing of G-isomorphism is $\Omega(\log n)$.*

Note that we have tighter lower bound for the same case when G is primitive.

In Section 4.2 we give the formal definitions. In Sections 4.3, 4.4 and 4.5 we give the proofs of the above three results. In Section 4.6 we state further nearly tight bounds that follow from our results (in addition to Theorem 4.1).

Table 4.1 summarizes our results on G-isomorphism. Table 2 gives the results of Fischer and Matsliah on Graph Isomorphism. In Table 3 we specialize our results to the case of Graph Isomorphism for comparison with the results of Fischer and Matsliah.

	Query-1 Complexity	**Query-2 Complexity**
1-sided-error testing	$\widetilde{\Theta}(\log \lvert G\rvert)^{\ddagger}$, $\Omega(\log n)^{\dagger}$	$\widetilde{\Theta}(\sqrt{n \log \lvert G\rvert})^{\ddagger}$
2-sided-error testing	$O(\log \lvert G\rvert)$, $\Omega(\log n)^{\dagger}$	$O(\sqrt{n \log \lvert G\rvert})$

† The lower bound holds when G is transitive and $|G| = 2^{O(n^{1-\epsilon})}$.
‡ The lower bound is for primitive G and the upper bound has no tilde.

Table 4.1: Bounds on the query complexity of Testing of Equivalence under G-isomorphism.

4.2 Preliminaries

4.2.1 Definitions

Let Ω be a set of size n. The permutations of Ω form the **symmetric group** $\mathrm{Sym}(\Omega)$ of order $n!$. We write the action of $\pi \in \mathrm{Sym}(\Omega)$ as $i \mapsto i^\pi$. For a subset $S \subseteq \Omega$ we set $S^\pi = \{i^\pi : i \in S\}$.

A subgroup G of $\mathrm{Sym}(\Omega)$ is a **permutation group**; Ω is the **permutation domain** on which G acts. G has **order** $|G|$ and **degree** n.

G is **transitive** if $(\forall i, j \in \Omega)(\exists \pi \in G)(i^\pi = j)$. A partition $\Omega = \Omega_1 \dot\cup \ldots \dot\cup \Omega_m$ is **invariant** under $\pi \in \mathrm{Sym}(\Omega)$ if $(\forall i)(\exists j)(\Omega_i^\pi = \Omega_j)$. The partition is invariant under G if it is invariant under every $\pi \in G$. The trivial partitions correspond to $m = 1$ or $m = n$; these are always invariant. If G is transitive and does not admit any nontrivial invariant partition then G is **primitive**. The largest primitive permutation groups of degree n other than the symmetric and the alternating groups (groups of even permutations) have order $\exp(O(\sqrt{n}\log^2 n))$ ([Bab81, Bab82]) so except for the two classes of "giants" of order $n!$ and $n!/2$, resp., $\log(|G|) = \widetilde{O}(\sqrt{n})$ for all primitive groups of degree n.

We use the notation $[n] = \{1, 2, 3, ..., n\}$. Most often we take $\Omega = [n]$ and write S_n for $\mathrm{Sym}([n])$.

Definition 4.4. A *partial assignment* is a function $p : S \to \{0, 1\}$ where $S \subseteq [n]$. We call S the support of this partial assignment and often denote $|S|$ as $|p|$. We call x a (full) assignment if $x : [n] \to \{0, 1\}$. (Note than a string $x \in \{0, 1\}^n$ can be thought of as a full assignment.) We say $p \subseteq x$ if x is an extension of p, *i. e.*, if $p = x|_S$ (the restriction of x to S).

$\mathrm{Ham}(x, y)$ will denote the Hamming distance of the strings (full assignments) x and y.

Definition 4.5. Let $T \subseteq [n]$ and let $\pi \in S_n$.

Let G be a permutation group acting on $[n]$. Then the sets T^π, where $\pi \in G$, are called the *G-shifts* of T. If $p : T \to \{0, 1\}$ is a partial assignment then we define $p^\pi : T^\pi \to \{0, 1\}$ as $p^\pi(i) = p(i^{\pi^{-1}})$.

Given two full assignments x and y and a permutation group G we denote by $d_G(x, y)$ the minimum distance between the G-shifts of x and y. That is,

$$d_G(x, y) = \min_{\pi_1, \pi_2 \in G} \mathrm{Ham}(x^{\pi_1}, y^{\pi_2}). \tag{4.1}$$

Since G is a group, we have

$$d_G(x, y) = \min_{\pi \in G} \mathrm{Ham}(x, y^\pi) = \min_{\pi \in G} \mathrm{Ham}(x^\pi, y). \tag{4.2}$$

If $d_G(x, y) = 0$ then we say "x is G**-isomorphic** to y."

A **2-sided property tester** for G-isomorphism is a probabilistic decision tree, say $\mathcal{A}$, such that given $x, y \in \{0, 1\}^n$

if $d_G(x, y) = 0$ then with probability $> \frac{2}{3}$ we have $\mathcal{A}(x, y) = 1$, and,

if $d_G(x, y) \geq \delta n$ then with probability $> \frac{2}{3}$ we have $\mathcal{A}(x, y) = 0$.

	Query-1 Complexity	Query-2 Complexity
1-sided-error testing	$\widetilde{\Theta}(\lvert V\rvert)$	$\widetilde{\Theta}(\lvert V\rvert^{3/2})$
2-sided-error testing	$\widetilde{\Theta}(\sqrt{\lvert V\rvert})$	$\Omega(\lvert V\rvert)$, $\widetilde{O}(\lvert V\rvert^{5/4})$

Table 4.2: The results of Fischer and Matsliah for Graph Isomorphism.

An **1-sided error property tester** is one which makes no mistake if $d_G(x, y) = 0$.

The complexity of a property tester is the maximum (over all possible inputs) of the minimum number of bits that need to be queried. If neither x nor y is given (so both need to be queried) then we speak of a *query-2 tester* and correspondingly of **query-2 complexity**. If one of them is given (we always assume y is given) and only the other (that is x) needs to be queried then we speak of a *query-1 tester* and **query-1 complexity**.

The trivial upper bound on the complexity of query-2 testers is $2n$ and of query-1 testers is n.

All our upper bound results hold for any permutation group G. But for our lower bound results we need some more structure on G. In Theorem 4.3 we assume that the group is transitive while Theorem 4.1 holds for primitive groups. Our main tool for primitive groups is the O'Nan–Scott Theorem (see Section 4.3).

4.2.2 Previous Results

The query complexity of the property testing version of graph isomorphism has been well studied. Fischer and Matsliah [FM06] gave some tight bounds. In case of graph isomorphism the group that acts is $S^{(2)}_{|V|}$, where V is the vertex set of the graph. Hence the order of the group is $|V|!$. Table 2 shows the main results of [FM06].

4.2.3 Chernoff bounds

We shall repeatedly use the following version of the Chernoff bounds, as presented by N. Alon and J. Spencer [AS92, Corollary A.14].

Let $X_1, X_2, \ldots, X_k$ be mutually independent indicator random variables and $Y = \sum_{i=1}^{k} X_i$. Let the expected value of Y be $\mu = E[Y]$. For all $\alpha > 0$,

$$\Pr[|Y - \mu| > \alpha\mu] < 2e^{-c_\alpha \mu},$$

where $c_\alpha > 0$ depends only on α.

4.3 Query Complexity for 1-sided-error Testing of Equivalence under some Primitive Group Action

4.3.1 Structure of Primitive Groups

Definition 4.6. Let G be a permutation group acting on a set A and H a permutation group acting on a set B. The *wreath product* $G \wr H$ is the split extension of the base group G^B (the cartesian product of $|B|$ copies of G) by H, where H acts on G^B by permuting the factors as it does the elements of B. Identifying G^B with the set of functions $f : B \to G$ we have $h^{-1} f h(b) = f(h^{-1}(b))$ for $h \in H$, $b \in B$.

There are two natural actions of $G \wr H$.

1. The imprimitive action on $A \times B$. The base group acts in the first coordinate by the rule $f(a, b) = (f(b)(a), b)$ and H acts on the second coordinate in the usual way.

2. The product action on the set A^B of $B \to A$ functions, where the base group acts coordinatewise (that is, if $p \in A^B$, $f \in G^B$, then $(fp)(b) = f(b)(p(b))$ and H acts by permuting the coordinates $((hp)(b) = p(h^{-1}(b))$ for $g \in G^B, h \in H)$.

Note that these are two permutation representations of the same group. Note also that $G \wr H$ has $G^{|B|}$ as a normal subgroup with H as the quotient.

The structure of primitive permutation groups is described by the O'Nan–Scott Theorem. A useful consequence of that theorem is given by Cameron.

Theorem 4.7. *[Cam81] There is a (computable) constant c with the property that, if G is a primitive permutation group of degree n, then one of the following holds:*

1. $|G| \leq n^{c \log n}$.

2. *G is a subgroup of $Aut(A_m^{(k)}) \wr S_\ell$ (product action) containing $(A_m^{(k)})^\ell$, where $A_m^{(k)}$ is the alternating group A_m acting on k-element subsets. [We can assume without loss of generality that $1 \leq k \leq \frac{m}{2}$].*

So in the case $|G| > n^{c \log n}$ the degree of G is given by

$$n = \binom{m}{k}^\ell \text{ and therefore } n \geq m^\ell. \tag{4.3}$$

It follows that $\ell \leq \log_2 n$. Also since we can assume $k \leq \frac{m}{2}$, so

$$\binom{m}{k} \geq \left(\frac{m}{k}\right)^k \geq 2^k \text{ and therefore } k \leq \log_2 n. \tag{4.4}$$

In fact if $|G| > n^{c \log n}$ then we obtain the bound on the size of G as

$$|G| \leq (m!)^\ell (\ell!) < m^{m\ell} \ell^\ell \leq n^m \ell^\ell \text{ [From Equation 4.3]} \tag{4.5}$$

Since $\ell \leq \log_2 n$ we have from Equation 4.5,

$$c(\log n)^2 < \log(|G|) < (m \log n + \ell \log \ell) \sim m \log n. \tag{4.6}$$

The last asymptotic equality holds because $\ell < \log n$ and therefore $\ell \log \ell = o(\log^2 n)$.

Therefore,

$$\log |G| \lesssim m \log n \text{ and } m \gtrsim c \log n. \tag{4.7}$$

It follows in particular that $m \geq 7$ (for sufficiently large n). The significance of this is in the known fact that for $m \geq 7$ we have

$$Aut(A_m) = S_m, \tag{4.8}$$

and therefore $Aut(A_m^{(k)}) = S_m^{(k)}$.

Observation 4.8. *If* $k = O(\sqrt{m})$ *then*

$$\binom{m}{k} = \Theta\left(\frac{m^k}{k!}\right).$$

Corollary 4.9. *Either* $\sqrt{n \log |G|} = \widetilde{O}(\sqrt{n})$ *or*

$$m\binom{m}{k}^{\ell/2} = \widetilde{O}(\sqrt{n \log |G|})$$

.

Proof. Let $k > \sqrt{m}$. Then

$$n = \binom{m}{k} > \left(\frac{m}{k}\right)^k > 2^k > 2^{\sqrt{m}}.$$

Therefore $m = O((\log n)^2)$ which implies from Equation 4.6 $\log |G| < (\log n)^3$. Hence if $k > \sqrt{m}$ we have $\sqrt{n \log |G|} = \widetilde{O}(\sqrt{n})$. The corollary now follows from Observation 4.8. ☐

Definition 4.10. Let $A, B \subset [n]$ and $p : A \to \{0,1\}$ and $q : B \to \{0,1\}$ be two partial assignments. Let G be a permutation group on $[n]$. Then p and q are said to be **G-agreeable** if there exists a full assignment x on $[n]$ and two elements $\pi_1, \pi_2 \in G$ such that x is an extension of both p^{π_1} and q^{π_2}. Since G is a group this is same as saying p and q are G-agreeable if there exists an element $\pi \in G$ and a full assignment x such that x is an extension of both p^{π} and q. We say that p and q are agreeable through π.

We say that the partial assignments p and q are compatible if there is a full assignment x on $[n]$ which is an extension of both p and q.

Definition 4.11. Let G be a permutation group on $[n]$. Let x and y be two full assignments on $[n]$. Then x and y are called k-G-agreeable if for any sets $A, B \subset [n]$ with $|A|, |B| \leq k$, the partial assignments $x|_A$ and $y|_B$ are G-agreeable.

4.3.2 G-Agreeability Lemma for G Primitive

The following proposition is folklore.

Proposition 4.12. *Let G be a transitive group on $[n]$. Let us fix $A, B \subset [n]$ and let us select $\pi \in G$ uniformly at random. Then*

$$E(|A^\pi \cap B|) = \frac{|A||B|}{n}. \tag{4.9}$$

Proof. By G-symmetry, for each $b \in B$ we have $\Pr(b \in A^\pi) = \frac{|A|}{n}$. Now the linearity of expectation yields the result. ☐

Corollary 4.13. *Let G be a transitive group on $[n]$. Let $A, B \subset [n]$ with $|A|, |B| \leq \epsilon\sqrt{n}$. Then,*

$$\Pr_{\pi \in G}[A^\pi \cap B = \phi] > (1 - \epsilon^2)$$

In particular if A and B are the support of the partial functions p and q, respectively, then p and q are G-agreeable.

Proof. Immediate from Proposition 4.12 by Markov's inequality. ☐

A simple consequence of Corollary 4.13 is that if G is a transitive group then any two full assignments x and y on $[n]$ are $\sqrt{n}$-G-agreeable.

Next we state the most technical lemma of this chapter - the G-Agreeability Lemma for primitive groups.

Lemma 4.14 (G-Agreeability Lemma)**.** *Let G be a primitive group. Then there exist two full assignments x and y on $[n]$ such that $d_G(x, y) \geq n/6$ and x and y are $\widetilde{O}(\sqrt{n \log |G|})$-$G$-agreeable.*

4.3.3 Lower Bounds for 1-sided error Testing

Proof of Part 1 of Theorem 4.1

Let $\mathcal{A}$ be a 1-sided-error query-2 property tester for G-isomorphism. Let the inputs be x and y. After the queries are made we get two partial functions $x|_{Q_x}$ and $y|_{Q_y}$. Now if $x|_{Q_x}$ and $y|_{Q_y}$ are G-agreeable then we have no proof that $d_G(x, y) \neq 0$. Since $\mathcal{A}$ is a 1-sided-error tester, it has to output 1. So by Lemma 4.14 we see that there exists x and y such that $d_G(x, y) \geq \frac{1}{6}n$ and $\mathcal{A}(x, y)$ has to be 1 if the query size is $\widetilde{O}(\sqrt{n \log |G|})$. So the result follows from the lemma.

Proof of Part 2 of Theorem 4.1

We recall the example for lower bound of 1-sided query-1 complexity of graph isomorphism given by Fischer and Matsliah [FM06]. The unknown graph is the complete graph on n vertices while the known graph is the union of $n/2$ isolated vertices and a complete graph on $n/2$ vertices. Note that without querying at least $n/4$ pairs of vertices it is impossible to give a certificate of non-isomorphism. This gives the lower bound of $n/4$ for the graph isomorphism case.

A similar example can be given in case of isomorphism under primitive group action. First of all we assume that the primitive group is of size more than $n^{c \log n}$ where the c is same as in Cameron's Theorem 4.7. Now we use the structure of the primitive group given by Cameron. We continue with the same notation as in Section 4.3.4. We partition V_1 into three disjoint parts, namely $V_a, V_b,$ and V_c, where $|V_a| = |V_b| = |V_c| = \frac{m}{3}$. The known input is

$$x(W) = 1 \text{ iff } W \in \binom{V_a, V_c, V_2 \ldots, V_\ell}{1, k-1, k, \ldots, k}$$

The unknown input is

$$y(W) = 1 \text{ iff } W \in \binom{(V_a \cup V_b), V_c, V_2, \ldots, V_\ell}{1, k-1, k, \ldots, k}$$

Note that one need to make at least $m/6$ queries to give a certificate of non-isomorphism between the two inputs. Now from Equation 4.7 we get a lower bound of $\Omega(\frac{\log(|G|)}{\log n})$.

4.3.4 Proof of the G-Agreeability Lemma for Primitive Groups

Proof of Lemma 4.14. If $|G| \leq n^{c \log n}$ then $\sqrt{n \log |G|} = \widetilde{O}(\sqrt{n})$ and the result follows from Corollary 4.13. Therefore from Theorem 4.7 and Corollary 4.9 we may assume that G is a subgroup of $S_m^{(k)}) \wr S_\ell$ (product action) containing $(A_m^{(k)})^\ell$, and $k < \sqrt{m}$. Hence in rest of the proof we will use from Lemma 4.8 that

$$\binom{m}{k} = \Theta\left(\frac{m^k}{k!}\right)$$

where the impied constant is absolute.

If $\ell = 1$ and $G = S_m^{(2)}$ then G is the group of automorphisms of the complete graph on m vertices. This case was settled by Fischer and Matsliah [FM06]. We generalize their technique.

For our convenience we have the following definition.

Definition 4.15. Let $T_1, T_2, \ldots, T_s$ be disjoint sets and $r_1, r_2, \ldots, r_s$ be positive integers satisfying $\sum_{i=1}^{s} r_i = R$. Then by $\binom{T_1, T_2, \ldots, T_s}{r_1, r_2, \ldots, r_s}$ we mean the set of R-tuples formed by r_i distinct elements from the set T_i for all $1 \leq i \leq s$. That is,

$$\binom{T_1, T_2, \ldots, T_s}{r_1, r_2, \ldots, r_s} = \left\{ \bigcup_{i=1}^{s} S_i \middle| S_i \subseteq T_i, |S_i| = r_i \right\}$$

G is a subgroup of $S_m^{(k)} \wr S_\ell$ (product action) containing $(A_m^{(k)})^\ell$. G is naturally isomorphic to a subgroup of $S_m \wr S_\ell$, acting in its imprimitive action on $\mathcal{V} = \cup_{i=1}^{\ell} V_i$, where $|V_i| = m$ and the V_i are all disjoint. Then any full assignment is a function from the set $\binom{V_1, \ldots, V_\ell}{k, k, \ldots, k}$ to $\{0, 1\}$.

We will first have to define two full assignments, x and y, on n bits. The group G is a map from $\mathcal{V}$ to $\mathcal{V}$. The rest of our proof has the following two parts:

- Define the full assignments x and y and prove that $d_G(x, y) > \delta n$ for some constant δ.
- Let Q_x and Q_y be two query sets for x and y, respectively such that both $|Q_x|$ and $|Q_y|$ is $\widetilde{O}(\sqrt{n \log |G|})$. Then we prove that there exist a permutation $\pi = \pi_1 \times \pi_2 \times \cdots \times \pi_\ell \in (A_m^{(k)})^\ell$ such that Q_x^π and Q_y are compatible.

We start with defining x.

Definition of the full assignments x and y

We partition V_1 into three disjoints parts U_1, U_2 and U_3 such that

$$|U_3| = m\left(1 - \frac{1}{k}\right), \ |U_1| = m\left(\frac{1}{2k} + \epsilon\right) \text{ and } |U_2| = m\left(\frac{1}{2k} - \epsilon\right)$$

We define x and y as

$$x(W) = 1 \text{ iff } W \in \binom{U_1, U_3, V_2, \ldots, V_\ell}{1, k-1, k, k, \ldots, k}$$

$$y(W) = 1 \text{ iff } W \in \binom{U_2, U_3, V_2, \ldots, V_\ell}{1, k-1, k, k, \ldots, k}$$

Note that a map from $\mathcal{V}$ to $\mathcal{V}$ gives a reordering of the bits is x.

Now note that number of 1s in x and y is $m\binom{m(1-\frac{1}{k})}{k-1}(\frac{1}{2k} + \epsilon)\binom{m}{k}^{\ell-1}$ and $m\binom{m(1-\frac{1}{k})}{k-1}(\frac{1}{2k} - \epsilon)\binom{m}{k}^{\ell-1}$ respectively. So from the difference in number of 1s in x and y we see that

$$d_G(x, y) \geq 2\epsilon m \binom{m(1-\frac{1}{k})}{k-1}\binom{m}{k}^{\ell-1}$$

For $k = 1$ the right-hand side is $2\epsilon m^{\ell} = 2\epsilon n$. If $k \neq 1$ then from Lemma 4.8 and the fact that $\left(1 - \frac{1}{k}\right)^{k-1} \geq \frac{1}{e}$ we obtain

$$2\epsilon m\binom{m(1-\frac{1}{k})}{k-1} \sim 2\epsilon\frac{m^k(1-\frac{1}{k}))^{k-1}}{(k-1)!} \geq \epsilon k\frac{2m^k}{ek!} = \Theta\left(\epsilon k\binom{m}{k}\right).$$

So if we choose $\epsilon = \frac{1}{6ck}$ where c is the constant implied in the Θ notation then we get that

$$d_G(x, y) \geq \frac{1}{6}\binom{m}{k}^{\ell} = \frac{1}{6}n.$$

Now we give the second part of the proof. Let Q_x and Q_y be query sets for x and y, respectively, such that $|Q_x|, |Q_y| \leq M$ where $M = \frac{m}{18k}\sqrt{\binom{m(1-\frac{1}{k})}{k-1}\binom{m}{k}^{\ell-1}}$.

To prove that x and y are M-G-agreeable, we have to give a $\pi \in (A_m^{(k)})^{\ell} \subseteq G$ that maps $\mathcal{V}$ to $\mathcal{V}$ such that Q_x^{π} and Q_y agrees.

If $a \in U_1$ then we define

$$q_x(a) = \left\{w \in \binom{U_1, U_3, V_2, \ldots, V_\ell}{1, k-1, k, \ldots, k} \middle| w \in Q_x \text{ and } a \in w\right\}$$

Similarly if $b \in U_2$ let

$$q_y(b) = \left\{w \in \binom{U_1, U_3, V_2, \ldots, V_\ell}{1, k-1, k, \ldots, k} \middle| w \in Q_y \text{ and } b \in w\right\}$$

Now by an averaging argument there exist sets $A \subset U_1$ and $B \subset U_2$ such that $|A|, |B| > \frac{2m}{9k}$ and for all $a \in A$ and $b \in B$ we have

$$|q_x(a)|, |q_y(b)| \leq \frac{9k}{m}M.$$

Let $H = A^{(k-1)}_{m(1-\frac{1}{k})} \times (A_m^{(k)})^{\ell-1}$ acting on the set $\binom{U_3, V_2, \ldots, V_\ell}{k-1, k, k, \ldots, k}$. Pick a random element $\pi' \in H$. Note that H acts transitively on the set $\binom{U_3, V_2, \ldots, V_\ell}{k-1, k, k, \ldots, k}$.

Fix an arbitrary even bijection from A to B, i.e., an even permutation of $[n]$ which maps A to B. Let $a \in A$ be mapped to $b \in B$. We call a pair (a, b)

acceptable if $q_x(a)^{\pi'} \cap q_y(b) = \phi$. We want to calculate the probability of a pair (a, b) being acceptable.

Note that q_a and q_b are two subsets of $\binom{U_3, V_2, \ldots, V_\ell}{k-1, k, \ldots, k}$. So from Lemma 4.13 we get that probability that a and b are compatible is more that $\frac{3}{4}$.

So the expected number of (a, b) pairs that are acceptable is $\geq \frac{3}{4}\frac{2m}{9k} = \frac{m}{6k} = \epsilon m$. So there exist a permutation $\pi' \in H$ such that ϵm of the (a, b) pairs are acceptable. These acceptable pairs along with the permutation π' give a map from a set $A' \subset A \subset U_1$ to $B' \subset B \subset U_2$ such that Q_x and Q_y are compatible. Now we have

$$|U_1 \backslash A'| = |U_1|$$

and

$$|U_2| = |U_2 \backslash B'|.$$

Hence π' and the map from the acceptable pairs can be extended to a mapping π from $\mathcal{V}$ to $\mathcal{V}$ by mapping $U_1 \backslash A'$ and U_2 to U_1 and $U_2 \backslash B'$ respectively, such that Q_x^π and Q_y are compatible.

Finally from Corollary 4.9 we have $M = \widetilde{O}(\sqrt{n \log |G|})$. □

4.4 Upper bounds for Transitive groups

Proof of Proposition 4.2

Definition 4.16. *We define **query sequence** as the sequence of elements of $[n]$ consisting of the positions of the bits of the input that will be queried. Repetition is permitted.*

The proofs of both parts of Proposition 4.2 are rather simple applications of the Chernoff bound; we describe the proofs for completeness.

Proof of Part 1 of Proposition 4.2. In this part we only have to query bits of x. Let us choose a real number p, $0 < p < 1$, appropriately (see below). The length of the query sequence Q will be $m = pn$. We say that two partial functions

p, q *contradict* at i if both $p(i)$ and $q(i)$ are defined and $p(i) \neq q(i)$. The following is the test:

1. Construct the query sequence $Q = (a_1, \ldots, a_m)$ by choosing pn elements of $[n]$ independently at random. (So there is a small chance that the same element is chosen twice.)

2. Query the bits of x corresponding to Q. So we obtain the partial function $x|_Q$.

3. If for some $\pi \in G$ the partial function $x|_Q^\pi$ and y contradict in fewer than $\delta pn/2$ places then output 1. Otherwise output 0.

Now to prove that the above test works we have to show that the test outputs the correct answer with probability at least $\frac{2}{3}$.

Given a permutation $\pi \in G$, we say that the ith bit queried contradicts y along π if $x(a_i) \neq y(a_i^\pi)$. We define the (0,1)-variable X_i^π by

$$X_i^\pi = 1 \text{ if the } i\text{th bit queried contradicts } y \text{ along } \pi.$$

$X^\pi = \sum X_i^\pi$ is the number of places the partial information of the two strings contradicts along π. Since the members of the query sequence are chosen independently, the X_i^π are mutually independent indicator random variables ($i = 1, \ldots, m$; π is fixed). So we can use the Chernoff bound to estimate the value of X^π.

Suppose that one of the following conditions holds for a given $\pi \in G$:

(i) x^π and y agrees completely;

(ii) x^π and y differ in more than δn places.

It follows that the expected value of X^π is less than 0 in Case (i) and greater than $pn\delta$ in Case (ii). Let $\eta = \delta/2$.

So, using the Chernoff bound we obtain,

$$\Pr\left[|X_\pi - E(X_\pi)| > \eta np\right] \le 2\exp\left(-c_{\eta/\delta} np\right). \tag{4.10}$$

If there exists $\pi \in G$ satisfying condition (i) (so the correct answer is 1), the probability we err is less than the right-hand side of this inequality.

If every $\pi \in G$ satisfies (ii) (so the correct answer is 0), the probability we err is less than $|G|$ times the right-hand side by the union bound. So in any case, the probability of error is less than $|G|\exp\left(-c_{\eta/\delta} np\right)$.

If we take $p = \frac{2+\log(|G|)}{c_{\eta/\delta} n}$ then the probability of error is less than $\frac{1}{3}$.

Note that this is an 1-sided error algorithm. So the query-1 complexity for the 1-sided error G-isomorphism testing is $O(pn) = O(1 + \log|G|)$. $\square$

Proof of Part 2 of Proposition 4.2. In this part we have to query both x and y. Again we choose a real number p, $0 < p < 1$, appropriately (see below). The total length of the query sequence will be $2m = 2pn$. The following is the test:

1. Construct two query sequences $Q_1 = (a_1, \ldots, a_m)$ and $Q_2 = (b_1, \ldots, b_m)$, by choosing these $2m$ elements of $[n]$ independently at random.

2. Query the bits of x and y corresponding to Q_1 and Q_2 respectively. So we obtain the partial functions $x|_{Q_1}$ and $y|_{Q_2}$.

3. If for some group element $\pi \in G$, the partial function $x|_{Q_1}^{\pi}$ contradicts the partial function $y|_{Q_2}$ in fewer than $p^2 n\delta/2$ places then output 1. Otherwise output 0.

To prove that the above test works we have to show that the test outputs the correct answer with probability at least $\frac{2}{3}$.

Given a permutation $\pi \in G$, we say that the ith bit queried in x contradicts $y|Q_2$ along π if $a_i^\pi \in Q_2$ and $x(a_i) \neq y(a_i^\pi)$. We define the $(0,1)$-random variable X_i^π by

$$X_i^\pi = 1 \text{ if the } i\text{th bits queried contradict along } \pi.$$

$X^\pi = \sum X_i^\pi$ is the number of places the queried information about the two strings contradicts along π. Since the members of the query sequences are chosen independently, the X_i^π are mutually independent indicator random variables ($i = 1, \ldots, m$; π is fixed). So we can use the Chernoff bound to estimate the value of X^π.

For any group element π, let D_π be the set of positions of the bits of x^π that differ from y. The expected number number of bits in D_π that are queried is $p|D_\pi|$. Now for X_i^π to be 1 we must also have $a_i^\pi \in Q_2$. Now the expected number of bits in D_π that are queried in both x and y is $p^2|D_\pi|$. So $E[X^\pi] = p^2|D_\pi|$.

Suppose that one of the following conditions holds for a given $\pi \in G$:

(i) x^π and y agrees completely;

(ii) x^π and y differ in more than δn places.

If condition (i) holds then $|D_\pi|$ is less than 0 and hence $E[X^\pi]$ is less than $\epsilon p^2 n$. In case of condition (ii), $|D_\pi|$ is greater than δn and hence $E[X^\pi]$ is greater than $\delta p^2 n$. Let $\eta = \delta/2$. From the Chernoff bound we get,

$$\Pr\left[|X^\pi - E[X^\pi]| > \eta p^2 n\right] \leq 2\exp\left(-c_{\eta/\delta} p^2 n\right) \tag{4.11}$$

If there exists $\pi \in G$ satisfying condition (i) (so the correct answer is 1), the probability we err is less than the right-hand side of this inequality.

If for every π condition (ii) is satisfied then the probability we err is less than $|G|$ times the right-hand side by the union bound.

If we take $p = \sqrt{\frac{2+\log|G|}{c_{\eta/\delta} n}}$ the error is less than $\frac{1}{3}$.

Again note that this algorithm is also 1-sided. So the query-2 complexity of 1-sided error G-isomorphism testing is $O(\sqrt{n(1+\log|G|)})$. $\square$

4.5 Lower bounds for Transitive Groups

Proof of Theorem 4.3

We will use the following lemma and the Theorem.

Lemma 4.17. *If G is a transitive permutation group and S is a subset of $[n]$, $|S| = k$ then there exist at least $\frac{n}{k^2}$ pairwise disjoint G-shifts of S.*

Theorem 4.18. ([*Fis04, FNS04*]) *Let $x \in \{0,1\}^n$. Suppose that there exists a distribution D_P on inputs $y \in \{0,1\}^n$ such that $f(x,y) = 1$, and a distribution D_N on inputs $z \in \{0,1\}^n$ such that x and z are ϵ-far from satisfying the f. Suppose further that for any $\mathcal{Q} \subset [n]$ of size q, and any $g : \mathcal{Q} \to \{0,1\}$, we have*

$$\frac{2}{3} \Pr_{D_P|\mathcal{Q}}(g) < \Pr_{D_N|\mathcal{Q}}(g).$$

Then any 2-sided-error property test for f requires at least q queries.

Proof of Theorem 4.3. Let x be a full assignment. For any subset $P \subseteq [n]$ of size k and any $Q \in \{0,1\}^k$ let $p_Q = |\{\pi \in G : x^\pi|_P = Q\}|$. We call x "almost universal" if for all $Q \in \{0,1\}^k$ and for all subset P of size k, we have

$$|p_Q - \frac{n}{2^k}| \leq \frac{n}{5(2^k)}.$$

Note that this means that if we pick $\pi \in G$ at random then for all $Q \in \{0,1\}^k$ and for all subset P we have

$$|\Pr[x^\pi|_P = Q] - \mu| \leq \mu/5$$

where $\mu = 1/2^k$.

We prove the existence of an almost universal string using the probabilistic method. Pick a random full assignment x. Fix a subset $P \subset [n]$ of size k and queries the bits of x corresponding to the indices in P. For a fixed $Q \in \{0,1\}^k$ we will estimate p_Q. Using Lemma 4.17 we can place $\frac{n}{k^2}$ disjoint G-copies of the subset P in $[n]$. Let $\mathcal{S}$ denote the set of disjoint copies of P. Let $v_i^Q(x)$ be the $(0,1)$-indicator variable indicating whether the i-th G-copy of P in $\mathcal{S}$ is same as Q. Since x is chosen randomly these random variables are independent. Let

$$v_Q(x) = \sum v_i^Q(x).$$

So $v_Q(x)$ be the number of times Q occurs in $\mathcal{S}$. The expected value of $v_Q(x)$ is $\frac{n}{k^2 2^k}$. So using the Chernoff bound

$$\Pr\left[\left|v_Q(x) - \frac{n}{k^2 2^k}\right| > \frac{n}{5k^2 2^k}\right] \leq 2\exp\left(-\frac{c_{1/5}n}{k^2 2^k}\right).$$

So using the union bound we get

$$\Pr\left[\forall \pi \in G, \forall Q \in \{0,1\}^k, \forall P, \left|v_Q(x^\pi) - \tfrac{n}{k^2 2^k}\right| \leq \tfrac{n}{5k^2 2^k}\right]$$
$$\geq 1 - 2\exp\left(-\tfrac{c_{1/5}n}{k^2 2^k}\right)|G|\tbinom{n}{k}2^k.$$

If $|G| = 2^{O(n^{1-\epsilon})}$ for any positive ϵ and $k \leq (1-\gamma)(\log n)$ (where $\gamma > 0$), this probability is non-zero. Now since we had exactly (n/k^2) times of disjoint copies of P, so there is a string x such that

$$\forall \pi \in G, \forall Q \in \{0,1\}^n, \forall P, |\Pr[x^\pi|_P = Q] - \mu| \leq \mu/5$$

where $\mu = 1/2^k$.

Similarly we can show that existence of a full assignment such that it is $\frac{1}{3}$-far from x and still "almost universal." The argument is similar. Now probability that a random string is $\frac{1}{3}$-close to x is less than $\frac{1}{2^{o(n)}}$. Using the same argument as above we can say that the probability that a random string is $\frac{1}{3}$-far from x and is an "almost universal" string is more than

$$\left(1 - \frac{1}{2^{o(n)}} - 2\exp\left(-\frac{c_{1/5}n}{k^2 2^k}\right)|G|\binom{n}{k}2^k\right).$$

This is also positive for $k \leq \frac{\log n}{2}$ if $|G| = 2^{o(n)}$. Hence there exists a full assignment $y \in \{0,1\}^n$ which is $\frac{1}{3}$-far from x and is "almost universal."

Now let x be the string that to which we have full access. The unknown string is chosen from the following two distributions.

- D_P: Uniform random G-shift of x.
- D_N: Uniform random G-shift of y.

Now we now that x and y are $\frac{1}{3}$-far. And since both are "almost universal," for all subset $P \subset [n]$ of size $(1-\gamma)\log n$ and all $Q \in \{0,1\}^k$,

$$2/3 \Pr_{\pi \in G}\left[x^{\pi}|_P = Q\right] \leq \Pr_{\pi}\left[y^{\pi}|_P = Q\right].$$

Now by Theorem 3.5 we can say that it will be impossible to test G-isomorphism with less than $(1-\gamma)\log n$ queries. So the query-1 complexity of any property tester of G-isomorphism is $\Omega(\log n)$. $\square$

4.6 Tight bounds and comparisons

In addition to our main result, Theorem 4.1, we obtain tight bounds for polynomial-sized groups. Note that these include all linear groups of bounded dimension. These in turn include most finite simple groups: all the classical finite simple groups of bounded dimension (linear, symplectic, orthogonal, and unitary groups) and all exceptional simple groups of Lie type, not only in their "natural" representations but in any representation (cf. [Cam81]).

Corollary 4.19. *Let G be a transitive permutation group and $|G| = n^{O(1)}$. Then the query-1 complexity of 1-sided-error and 2-sided-error property testing of G-isomorphism is $\Theta(\log n)$.*

The next corollary gives an essentially tight bound for all groups of order $\exp(\text{polylog}(n))$. This includes all linear groups, and also includes all finite simple groups in any representation, except for the alternating groups.

	Query-1 Complexity	**Query-2 Complexity**
1-sided-error testing	$\widetilde{\Theta}(\|V\|)^{\dagger}$	$\widetilde{\Theta}(\|V\|^{3/2})^{\dagger}$
2-sided-error testing	$\widetilde{O}(\|V\|)$, $\Omega(\log(\|V\|))$	$\widetilde{O}(\|V\|^{3/2})$

† Matches the Fischer–Matsliah bounds.
‡ New results.

Table 4.3: Corollaries of our results to Graph Isomorphism.

Corollary 4.20. *Let G be a permutation group and assume* $\log(|G|) = (\log n)^{O(1)}$. *Then the query-2 complexity of 1-sided error property testing of G-isomorphism is* $\widetilde{\Theta}(\sqrt{n})$.

For comparison with the results of Fischer amd Matsliah, we also include a table of corollaries of our results when specialized to Graph Isomorphism. In this case we take $n = \binom{|V|}{2}$ and define G to be $G = S_{|V|}^{(2)}$ (the induced symmetric group action) and hence

$$\log(|G|) = \log(|V|!) \sim \sqrt{n/2}\log n.$$

In the case of 1-sided error, query-2 complexity, the special cases of our general bounds (for primitive groups) match the Fischer–Matsliah bounds.

4.7 Future Work

We obtain tight bounds for the 1-sided error query complexity when the group is primitive. Obtaining tight bound for the 2-sided error query complexity would be the obvious next step. Also we want to obtain tight bounds in the case when the group is transitive and not just primitive. In the tight bounds for primitive groups that we obtain, we use the classification of primitive groups. But is the special structure of primitive groups essential or are there similar bounds for the general transitive groups?

A test case would be the automorphism group of a complete binary tree in its action on the leaves. We have reason to believe that a solution to this case would bring us close to solving the general case of transitive groups. Let T_N denote the complete binary tree with $N = 2^h$ leaves. Let G be the action of the automorphism group of the tree on the leaves. Given a string x of length N we place the bits of x on the leaves of the tree T_N. Then G permutes the bits of x. For this particular transitive group, the query-1 and query-2 complexities of testing G-isomorphism are wide open both in the 1-sided error and 2-sided error cases.

Part IV

Quantum Query Complexity

CHAPTER 5
QUANTUM QUERY COMPLEXITY FOR DATABASE SEARCH

5.1 Introduction

In this chapter, we consider the problem of reducing the error in quantum search algorithms by making a small number of queries to the database. Error reduction in the form of amplitude amplification is one of the central tools in the design of efficient quantum search algorithms [Gro98a, Gro98b, BHMT02]. In fact, Grover's database search algorithm [Gro96, Gro97] can be thought of as amplitude amplification applied to the trivial algorithm that queries the database at a random location and succeeds with probability at least $\frac{1}{N}$. The key feature of quantum amplitude amplification is that it can boost the success probability from a small quantity δ to a constant in $O(1/\sqrt{\delta})$ steps, whereas, in general a classical algorithm for this would require $\Omega(1/\delta)$ steps. This basic algorithm has been refined, taking into account the number of solutions and the desired final success probability $1-\epsilon$. For example, Buhrman, Cleve, de Wolf and Zalka [BCdWZ99] obtained the following:

> **Theorem [BCdWZ99]:** Fix $\eta \in (0,1)$, and let $N > 0$, $\epsilon \geq 2^{-N}$, and $t \leq \eta N$. Let T be the optimal number of queries a quantum computer needs to search with error $\leq \epsilon$ through an unordered list of N items containing at least t solutions. Then $\log 1/\epsilon \in \Theta(T^2/N + T\sqrt{t/N})$ (Note that the constant implicit in the Θ notation can depend on η).

Recently, Grover [Gro05] considered error reduction for algorithms that err with small probability. The results were subsequently refined and extended by

Tulsi, Grover and Patel [TGP05]. Let us describe their results in the setting of the database search problem, where, given a database $f : [N] \to \{0, 1\}$, we are asked to determine an $x \in f^{-1}(1)$. If $|f^{-1}(0)| = \epsilon N$, then choosing x uniformly at random will meet the requirements with probability at least $1 - \epsilon$. This method makes no queries to the database. If one is allowed one classical query, the error can be reduced to ϵ^2 and, in general, with t classical queries one can reduce the probability of error to ϵ^{t+1}. It can be shown that no classical t-query randomized algorithm for the problem can reduce the probability of error significantly below ϵ^{t+1}, even if the value of ϵ is known in advance. Grover [Gro05] presented an interesting algorithm that makes one quantum query and returns an x that is in $f^{-1}(1)$ with probability $1-\epsilon^3$. Tulsi, Grover and Patel [TGP05] showed an iteration where one makes just one query to the database and performs a measurement, so that after t iterations of this operator the error is reduced to ϵ^{2t+1}. This algorithm works for all ϵ and is not based on knowing ϵ in advance. Thus this iteration can be said to exhibit a "fixed point" behavior [Gro05, TGP05], in that the state approaches the target state (or subspace) closer with each iteration, just as it does in randomized classical search. The iteration used in the usual Grover search algorithm [Gro98a, Gro98b] does not have this property. Note, however, that if the initial success probability is $\frac{1}{N}$, these new algorithms make $\Omega(N)$ queries to the database, whereas the original algorithm makes just $O(\sqrt{N})$ queries.

In [Gro05], the database is assumed to be presented by means of an oracle of the form $|x\rangle \to \exp(f(x)\pi i/3))|x\rangle$. The standard oracle for a function f used in earlier works on quantum search is $|x\rangle|b\rangle \mapsto |x\rangle|b \oplus f(x)\rangle$, where $\oplus$ is addition modulo two. It can be shown that the oracle assumed in [Gro05] cannot be implemented by using just one query to the standard oracle. In Tulsi, Grover and Patel [TGP05] the basic iteration uses the controlled version of the oracle, namely $|x\rangle|b\rangle|c\rangle \mapsto |x\rangle|b \oplus c \cdot f(x)\rangle|c\rangle$.

In this paper, we present a version of the algorithm that achieves the same reduction in error as in [Gro05], but uses the standard oracle. In fact, our basic one-query algorithm has the following natural interpretation. First, we note that for $\delta \leq \frac{3}{4}$, there is a one-query quantum algorithm $\mathcal{A}_\delta$ that makes no error if $|f^{-1}(0)| = \delta N$. Then, using a simple computation, one can show that the one-query algorithm corresponding to $\delta = \frac{1}{N}$ errs with probability less than ϵ^3 when $|f^{-1}(0)| = \epsilon N$. One can place these algorithms in a more general framework, just as later works due to Grover [Gro98a, Gro98b] and Brassard, Hoyer, Mosca and Tapp [BHMT02] placed Grover's original database search algorithm [Gro96, Gro97] in the general amplitude amplification framework. The framework is as follows: Suppose there is an algorithm G that guesses a solution to a problem along with a witness, which can be checked by another algorithm T. If the guess returned by G is correct with probability $1-\epsilon$, then there is another algorithm that uses G, G^{-1}, makes t queries to T, and guesses correctly with probability $1-\epsilon^{2t+1}$.

These algorithms show that, in general, a t-query quantum algorithm can match the error reduction obtainable by any $2t$-query randomized algorithm. Can one do even better? The main contribution of this paper are the *lower bounds* on the error probability of t-query algorithms. We show that the amplification achieved by these algorithms is essentially optimal (see Section 5.2.1 for the precise statement). Our result does not follow immediately from the result of Buhrman, Cleve, de Wolf and Zalka [BCdWZ99] cited above because of the constants implicit in the θ notation, but with a slight modification of their proofs one can derive a result similar to ours (see Section 5.2.1). Our lower bound result uses the polynomial method of Beals, Cleve, Buhrman, Mosca and de Wolf [BBC+98] combined with an elementary analysis based on the roots of low degree polynomials, but unlike previous proofs using this method, we do not rely on any special tools for bounding the rate of growth of low degree polynomials.

5.2 Background, definitions and results

We first review the standard framework for quantum search. We assume that the reader is familiar with the basics of quantum circuits, especially the quantum database search algorithm of Grover [Gro96, Gro97] (see, for example, Nielsen and

Chuang [NC00, Chapter 6]). The database is modelled as a function $f : [N] \to S$, where $[N] \stackrel{\Delta}{=} \{0, 1, 2, \ldots, N-1\}$ is the set of addresses and S is the set of possible items to be stored in the database. For our purposes, we can take S to be $\{0, 1\}$. When thinking of bits we identify $[N]$ with $\{0, 1\}^n$. Elements of $[N]$ will be called addresses, and addresses in $f^{-1}(1)$ will be referred to as targets. In the quantum model, the database is provided to us by means of an oracle unitary transformation T_f, which acts on an $(n+1)$-qubit space by sending the basis vector $|x\rangle|b\rangle$ to $|x\rangle|b \oplus f(x)\rangle$. For a quantum circuit $\mathcal{A}$ that makes queries to a database oracle in order to determine a target, we denote by $\mathcal{A}(f)$ the random variable (taking values in $[N]$) returned by $\mathcal{A}$ when the database oracle is T_f.

Definition 5.1. Let $\mathcal{A}$ be a quantum circuit for searching databases of size N. For a database f of size N, let $\mathsf{err}_{\mathcal{A}}(f) = \Pr[\mathcal{A}(f) \text{ is not a target state}]$. When ϵN is an integer in $\{0, 1, 2, \ldots, N\}$, let $\mathsf{err}_{\mathcal{A}}(\epsilon) = \max_{f:|f^{-1}(0)|=\epsilon N} \mathsf{err}_{\mathcal{A}}(f)$.

Using this notation, we can state Grover's result as follows.

Theorem 5.2 (Grover [Gro05])**.** *For all N, there is a one-query algorithm $\mathcal{A}$, such that for all ϵ (assuming ϵN is an integer), $\mathsf{err}_{\mathcal{A}}(\epsilon) = \epsilon^3$.*

This error reduction works in a more general setting. Let $[N]$ represent the set of possible solutions to some problem, and let $f : [N] \to \{0, 1\}$ be the function that checks that the solution is correct; as before we will assume that we are provided access to this function via the oracle T_f. Let G be a unitary transform that guesses a solution in $[N]$ that is correct with probability $1-\epsilon$. Our goal is to devise another guessing algorithm $\mathcal{B}$ that using T_f, G and G^{-1} produces a guess that is correct with significantly better probability. Let $\mathcal{B}(T_f, G)$ be the answer returned by $\mathcal{B}$ when the checker is T_f and the guesser is G.

Theorem 5.3 (Grover [Gro05])**.** *There is an algorithm $\mathcal{B}$ that uses T_f once, G twice and G^{-1} once, such that $\Pr[f(\mathcal{B}(T_f, G)) = 0] = \epsilon^3$, where ϵ is the probability of error of the guessing algorithm G.*

Note that Theorem 5.2 follows from Theorem 5.3 by taking G to be the Hadamard transformation, which produces the uniform superposition on all N states when applied to the state $|0\rangle$.

Theorem 5.4 ([Gro05, TGP05]). *For all $t \geq 0$ and all N, there is a t-query quantum database search algorithm such that, for all ϵ (ϵN is an integer),* $\mathsf{err}_{\mathcal{A}}(\epsilon) = \epsilon^{2t+1}$.

In Grover [Gro05], this result was obtained by recursive application of Theorem 5.3, and worked only for infinitely many t. Tulsi, Grover and Patel [TGP05] rederived Theorems 5.2 and 5.3 using a different one-query algorithm, which could be applied iteratively to get Theorem 5.4.

From now on when we consider error reduction for searching a database f and use the notation $\mathsf{err}_{\mathcal{A}}(\epsilon)$, ϵ will refer to $|f^{-1}(0)|/N$; in particular, we assume that $\epsilon N \in \{0, 1, \ldots, N-1\}$. However, for the general framework, ϵ can be any real number in $[0, 1]$.

5.2.1 Our contributions

As stated earlier, in order to derive the above results, Grover [Gro05] and Tulsi, Grover and Patel [TGP05] assume that access to the database is available using certain special types of oracles. In the next section, we describe alternative algorithms that establish Theorem 5.2 while using only the standard oracle $T_f : |x\rangle|b\rangle \rightarrow |x\rangle|b \oplus f(x)\rangle$. The same idea can be used to obtain results analogous to Theorems 5.3. By recursively applying this algorithm we can derive a version of Theorem 5.4 for t of the form $\frac{3^i-1}{2}$ where i is the number of recursive applications. Our algorithms and those in Tulsi, Grover and Patel [TGP05] use similar ideas, but were obtained independently of each other.

We also consider error reduction when we are given a lower bound on the error probability ϵ, and obtain analogs of Theorems 5.2 and 5.3 in this setting.

Theorem 5.5 (Upper bound result)**.** *(a) For all N and $\delta \in [0, \frac{3}{4}]$, there is a one-query algorithm A_δ such that for all $\epsilon \geq \delta$,*

$$\mathsf{err}_{\mathcal{A}_\delta}(f) \leq \epsilon \left[\frac{\epsilon - \delta}{1 - \delta}\right]^2.$$

(b) For all $\delta \in [0, \frac{3}{4}]$, there is an algorithm $\mathcal{B}_\delta$ that uses T_f once and G twice and G^{-1} once, such that

$$\mathsf{err}_{\mathcal{B}_\delta}(T_f, G) \leq \epsilon \left[\frac{\epsilon - \delta}{1 - \delta}\right]^2.$$

The case $\epsilon = \delta$ corresponds to the fact that one can determine the target state with certainty if $|f^{-1}(0)|$ is known exactly and is at most $\frac{3N}{4}$. Furthermore, Theorems 5.2 and 5.3 can be thought of as special cases of the above Proposition corresponding to $\delta = 0$. In fact, by taking $\delta = \frac{1}{N}$ in the above proposition, we obtain the following slight improvement over Theorem 5.2.

Corollary 5.6. *For all N, there is a one-query database search algorithm $\mathcal{A}$ such that for all ϵ (where $\epsilon N \in \{0, 1, \ldots, N\}$), we have* $\mathsf{err}_{\mathcal{A}}(\epsilon) \leq \epsilon \left[\frac{\epsilon - \frac{1}{N}}{1 - \frac{1}{N}}\right]^2$.

Lower bounds: The main contribution of this work is our lower bound results. We show that the reduction in error obtained in Theorem 5.2 and 5.3 are essentially optimal.

Theorem 5.7 (Lower bound result)**.** *Let $0 < \ell \leq u < 1$ be such that ℓN and uN are integers.*

(a) For all one-query database search algorithms $\mathcal{A}$, for either $\epsilon = \ell$ or $\epsilon = u$,

$$\mathsf{err}_{\mathcal{A}}(\epsilon) \geq \epsilon^3 \left(\frac{u - \ell}{u + \ell - 2\ell u}\right)^2.$$

(b) *For all t-query database search algorithms $\mathcal{A}$, there is an $\epsilon \in [\ell, u]$ such that ϵN is an integer, and*

$$\mathsf{err}_{\mathcal{A}}(\epsilon) \geq \epsilon^{2t+1} \left(\frac{b-1}{b+1} - \frac{1}{N\ell(b+1)} \right)^{2t},$$

where $b = (\frac{u}{\ell})^{\frac{1}{t+1}}$, and we assume that $N\ell(b-1) > 1$.

In particular, this result shows that to achieve the same reduction in error, a quantum algorithm needs to make roughly at least half as many queries as a classical randomized algorithm. A similar result can be obtained by modifying the proof in Buhrman, Cleve, de Wolf and Zaka [BCdWZ99]: there is a constant $c > 0$, such that for all $u > 0$, all large enough N and all t-query quantum search algorithms for databases of size N, there is an $\epsilon \in (0, u]$ (ϵN is an integer) such that $\mathsf{err}_{\mathcal{A}}(\epsilon) \geq (cu)^{2t+1}$.

5.3 Upper bounds: quantum algorithms

In this section, we present algorithms that justify Theorems 5.2, 5.3 and 5.4, but by using the standard database oracle. We then modify these algorithms to generalize and slightly improve these theorems.

5.3.1 Alternative algorithms using the standard oracle

We first describe an alternative algorithm $\mathcal{A}_0$ to justify Theorem 5.2. This simple algorithm (see Figure 5.1) illustrates the main idea used in all our upper bounds. We will work with n qubits corresponding to addresses in $[N]$ and one ancilla qubit. Although we do not simulate Grover's oracle directly, using the ancilla, we can reproduce the effect the complex amplitude used there by real amplitudes. As we will see, the resulting algorithm has an intuitive explanation and also some additional properties not enjoyed by the original algorithm.

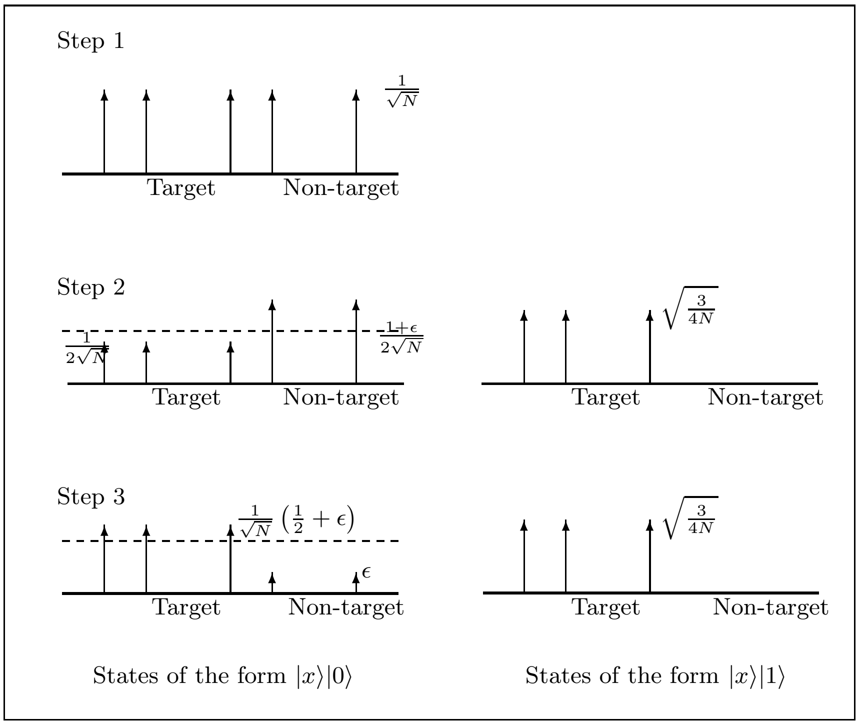

Figure 5.1: The one-query algorithm

Step 1: We start with the uniform superposition on $[N]$ with the ancilla bit in the state $|0\rangle$.

Step 2: For targets x, transform $|x\rangle|0\rangle$ to $\frac{1}{2}|x\rangle|0\rangle + \sqrt{\frac{3}{4}}|x\rangle|1\rangle$. The basis states $|x\rangle|0\rangle$ for $x \in f^{-1}(0)$, are not affected by this transformation.

Step 3: Perform an inversion about the average controlled on the ancilla bit being $|0\rangle$, and then measure the address registers.

Step 1 is straightforward to implement using the n-qubit Hadamard transform H_n. For Step 2, using one-query to T_f, we implement a unitary transformation U_f, which maps $|x\rangle|0\rangle$ to $|x\rangle|0\rangle$ if $f(x) = 0$ and to $|x\rangle\left(\frac{1}{2}|0\rangle + \sqrt{\frac{3}{4}}|1\rangle\right)$, if $f(x) = 1$. One such transformation is $U_f = (I_n \otimes R^{-1})T_f(I_n \otimes R)$, where I_n is the n-qubit identity operator and R is the one-qubit gate for rotation by $\frac{\pi}{12}$ (that is, $|0\rangle \overset{R}{\mapsto} \cos(\frac{\pi}{12})|0\rangle + \sin(\frac{\pi}{12})|1\rangle$ and $|1\rangle \overset{R}{\mapsto} \cos(\frac{\pi}{12})|1\rangle - \sin(\frac{\pi}{12})|0\rangle$). The inversion about the average is usually implemented as $A_n = -H_n(I_n - 2|\mathbf{0}\rangle\langle\mathbf{0}|)H_n$. The controlled version we need is then given by

$$A_{n,0} = A_n \otimes |0\rangle\langle 0| + I_n \otimes |1\rangle\langle 1|.$$

Let $H' = H_n \otimes I$. The final state is $|\phi_f\rangle = A_{n,0}U_f H'|\mathbf{0}\rangle|0\rangle$.

To see that the algorithm works as claimed, consider the state just before the operator $A_{n,0}$ is applied. This state is

$$\frac{1}{\sqrt{N}}\left[\sum_{x\in f^{-1}(1)} \frac{1}{2}|x\rangle|0\rangle \sum_{x\in f^{-1}(0)} |x\rangle|0\rangle\right] + \frac{1}{\sqrt{N}}\sum_{x\in f^{-1}(1)} \sqrt{\frac{3}{4}}|x\rangle|1\rangle.$$

Suppose $|f^{-1}(0)| = \epsilon N$. The "inversion about the average" is performed only on the first term, so the non-target states receive no amplitude from the second term. The average amplitude of the states in the first term is $\frac{1}{2\sqrt{N}}(1+\epsilon)$ and the amplitude of the states $|x\rangle|0\rangle$ for $x \in f^{-1}(0)$ is $\frac{1}{\sqrt{N}}$. Thus, after the inversion about the average the amplitude of $|x\rangle|0\rangle$ for $x \in f^{-1}(0)$ is $\frac{\epsilon}{\sqrt{N}}$. It follows that if we measure the address registers in the state $|\phi_f\rangle$, the probability of observing a non-target state is exactly

$$|f^{-1}(0)| \cdot \frac{\epsilon^2}{N} = \epsilon^3.$$

Remark: Note that this algorithm actually achieves more. Suppose we measure the ancilla bit in $|\phi_f\rangle$, and find a 1. Then, we are assured that we will find a target on measuring the address registers. Furthermore, the probability of the ancilla bit being 1 is exactly $\frac{3}{4}(1-\epsilon)$. One should compare this with the randomized one-query algorithm that with probability $1-\epsilon$ provides a guarantee that the solution it returns is correct. The algorithm in [Gro05] has no such guarantee associated with its solution. However, the algorithm obtained by Tulsi, Grover and Patel [TGP05] gives a guarantee with probability $\frac{1}{2}(1-\epsilon)$.

The general algorithm $\mathcal{B}_0$ needed to justify Theorem 5.3 is similar. We use G

instead of H_n; that is, we let $H' = G \otimes I$, $A_n = G(2|\mathbf{0}\rangle\langle\mathbf{0}| - I_n)G^{-1}$, and, as before,

$$A_{n,0} = A_n \otimes |0\rangle\langle 0| + I_n \otimes |1\rangle\langle 1|.$$

The final state is obtained in the same way as before $|\phi_f\rangle = A_{n,0}U_f H'|\mathbf{0}\rangle|0\rangle$.

Remark: As stated, we require the controlled version of G and G^{-1} to implement $A_{n,0}$. However, we can implement G with the uncontrolled versions themselves from the following alternative expression for $A_{n,0}$:

$$A_{n,0} = (G \otimes I)[(2|\mathbf{0}\rangle\langle\mathbf{0}| - I_n) \otimes |0\rangle\langle 0| + I_n \otimes |1\rangle\langle 1|](G^{-1} \otimes I).$$

We can estimate the error probability of this algorithm using the following standard calculation. Suppose the probability of obtaining a non-target state on measuring the address registers in the state $G|\mathbf{0}\rangle$ is ϵ. Let us formally verify that the probability of obtaining a non-target state on measuring the address registers in the state $|\phi_f\rangle$ is ϵ^3. This follows using the following routine calculation. We write

$$G|\mathbf{0}\rangle = \alpha|t\rangle + \beta|t'\rangle,$$

where $|t\rangle$ is a unit vector in the "target space" spanned by $\{|x\rangle : f(x) = 1\}$, and $|t'\rangle$ is a unit vector in the orthogonal complement of the target space. By scaling $|t\rangle$ and $|t'\rangle$ by suitable phase factors, we can assume that α and β are real numbers. Furthermore $\beta^2 = \epsilon$. The state after the application of U_f is then given by

$$\left(\frac{\alpha}{2}|t\rangle + \beta|t'\rangle\right)|0\rangle + \sqrt{\frac{3}{4}}\alpha|t\rangle|1\rangle. \tag{5.1}$$

Now, the second term is not affected by $A_{n,0}$, so the amplitude of states in the subspace of non-target states is derived entirely from the first term, which we denote by $|u\rangle$. To analyze this contribution we write $|u\rangle$, using the basis

$$|v\rangle = \alpha|t\rangle + \beta|t'\rangle; \tag{5.2}$$
$$|v'\rangle = \beta|t\rangle - \alpha|t'\rangle. \tag{5.3}$$

That is, $|u\rangle = \left(\frac{\alpha^2}{2} + \beta^2\right)|v\rangle|0\rangle - \frac{\alpha\beta}{2}|v'\rangle|0\rangle$.

Since $A_{n,0}|v\rangle = |v\rangle$ and $A_{n,0}|v'\rangle = -|v'\rangle$, we have

$$A_{n,0}|u\rangle = \left(\frac{\alpha^2}{2} + \beta^2\right)|v\rangle|0\rangle + \frac{\alpha\beta}{2}|v'\rangle|0\rangle.$$

Returning to the basis $|t\rangle$ and $|t'\rangle$ (using (5.2) and (5.3)), we see that the amplitude associated with $|t'\rangle$ in this state is β^3. Thus, the probability that the final measurement fails to deliver a target address is exactly $\beta^6 = \epsilon^3$.

Remark: The algorithm $\mathcal{B}_0$ can be used recursively to get a t-query algorithm that achieves the bound Theorem 5.4. Just as in the one-query algorithm, by measuring the ancilla bits we can obtain a guarantee; this time the solution is accompanied with guarantee with probability at least $(1 - \frac{1}{t} - \frac{6\log t}{t(\log\frac{1}{\epsilon})^{\log_3 4}})$. The t-query algorithm obtained by Tulsi, Grover and Patel [TGP05] has significantly better guarantees: it certifies that its answer is correct with probability at least $1 - \epsilon^{2t}$.

5.3.2 Algorithms with restrictions on ϵ

As stated above, for each $\delta \in [0, 1]$, there is a one-query quantum algorithm $\mathcal{A}_\delta$ that makes no error if the $|f^{-1}(0)| = \delta N$ (or, in the general setting, if G is known to err with probability at most $\frac{3}{4}$). Let us explicitly obtain such an algorithm $\mathcal{A}_\delta$ by slightly modifying the algorithm above. The idea is to ensure that the inversion about the average performed in Step 3 reduces the amplitude of the non-target states to zero. For this, we only need to replace U_f by $U_{f,\delta}$, which maps $|x\rangle|0\rangle$ to

$|x\rangle|0\rangle$ if $f(x) = 0$ and to $|x\rangle(\alpha|0\rangle + \beta|1\rangle)$, if $f(x) = 1$, where $\alpha = \dfrac{1-2\delta}{2(1-\delta)}$ and $\beta = \sqrt{1-\alpha^2}$.

Also, one can modify the implementation of U_f above, replacing $\frac{\pi}{12}$ by $\frac{\sin^{-1}(\alpha)}{2}$ (note that $\delta \leq \frac{3}{4}$ implies that $|\alpha| \leq 1$), and implement $U_{f,\delta}$ using just one-query to T_f.

Proposition 5.8. *Let* $|f^{-1}(0)| = \delta \leq \frac{3}{4}$. *Then,* $\mathsf{err}_{\mathcal{A}_\delta}(f) = 0$.

An analogous modification for the general search gives us an algorithm $\mathcal{B}_\delta(T, G)$ that has no error when G produces a target state for T with probability exactly $1-\delta$. We next observe that the algorithms $\mathcal{A}_\delta$ and $\mathcal{B}_\delta$ perform well not only when the original probability is known to be δ but also if the original probability is $\epsilon \geq \delta$. This justifies Theorem 5.5 claimed above.

Proof of Theorem 5.5: We will only sketch the calculations for part (a). The average amplitude of all the states of the form $|x\rangle|0\rangle$ is $(\frac{1}{\sqrt{N}})(1-2\delta+\epsilon)/(2(1-\delta))$. From this it follows that the amplitude of a non-target state after the inversion about the average is $(\frac{1}{\sqrt{N}})(\epsilon-\delta)/(1-\delta)$. Our claim follows from this by observing that there are exactly ϵN non-target states. □

5.4 Lower bounds

In this section, we show that the algorithms in the previous section are essentially optimal. For the rest of this section, we fix a t-query quantum search algorithm to search a database of size N. Using the polynomial method we will show that no such algorithm can have error probability significantly less than ϵ^{t+1}, for a large range of ϵ.

The proof has two parts. First, using standard arguments we observe that $\mathsf{err}_{\mathcal{A}}(\epsilon)$ is a polynomial of degree at most $2t+1$ in ϵ.

Lemma 5.9. *Let $\mathcal{A}$ be a t-query quantum search algorithm for databases of size N. Then, there is a univariate polynomial $r(Z)$ with real coefficients and degree at most $2t+1$, such that for all ϵ*

$$\mathsf{err}_{\mathcal{A}}(\epsilon) \geq r(\epsilon).$$

Furthermore, $r(x) \geq 0$ for all $x \in [0,1]$.

In the second part, we analyze such low degree polynomials to obtain our lower bounds. We present this analysis first, and return to the proof of Lemma 5.9 after that.

5.4.1 Analysis of low degree polynomials

Definition 5.10 (Error polynomial)**.** We say that a univariate polynomial $r(Z)$ is an *error polynomial* if (a) $r(z) \geq 0$ for all $z \in [0,1]$, (b) $r(0) = 0$, and (c) $r(1) = 1$.

Our goal is to show that an error polynomial of degree at most $2t+1$ cannot evaluate to significantly less than ϵ^{2t+1} for many values of ϵ. For our calculations, it will be convenient to ensure that all the roots of such a polynomial are in the interval $[0,1)$.

Lemma 5.11. *Let $r(Z)$ an error polynomial of degree $2t+1$ with $k < 2t+1$ roots in the interval $[0,1)$. Then, there is another error polynomial $q(Z)$ of degree at most $2t+1$ such that $q(z) \leq r(z)$ for all $z \in [0,1]$, and $q(Z)$ has at least $k+1$ roots in the interval $[0,1)$.*

Proof. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be the roots of $r(x)$ in the interval $[0,1)$. Hence we can write

$$r(Z) = \prod_{i=1}^{k}(Z - \alpha_i)r'(Z),$$

where $r'(Z)$ does not have any roots in $[0,1)$. Now, by substituting $Z = 1$, we conclude that $r'(1) \geq 1$. Since $r'(Z)$ does not have any roots in $[0,1)$, it follows that $r'(z) > 0$ for all $z \in [0,1)$.

The idea now, is to subtract a suitable multiple of the polynomial $1-Z$ from $r'(Z)$ and obtain another polynomial $r''(Z)$ which has a root in $[0,1)$. Since $1-Z$ is positive in $[0,1)$, $r''(Z)$ is at most $r'(Z)$ in this interval. The polynomial $q(Z)$ will be defined by $q(Z)=\prod_{\alpha\in R}(Z-\alpha)r''(Z)$. To determine the multiple of $1-Z$ we need to subtract, consider $\lambda(c)=\min_{z\in[0,1)} r'(Z)-c(1-Z)$. Since $\lambda(c)$ is continuous, $\lambda(0)>0$ and $\lambda(c)<0$ for large enough c, it follows that $\lambda(c_0)=0$ for some $c_0>0$. Now, let $r''(Z)=r(Z)-c_0(1-Z)$. $\square$

By repeatedly applying Lemma 5.11 we obtain the following.

Lemma 5.12. *Let $r(Z)$ be an error polynomial of degree at most $2t+1$. Then, there is an error polynomial $q(Z)$ of degree exactly $2t+1$ such that $q(z)\le r(z)$ for all $z\in[0,1]$, and $q(Z)$ has $2t+1$ roots in the interval $[0,1)$.*

We can now present the proof of Theorem 5.7, our main lower bound result.

Proof of Theorem 5.7: Consider the case $t=1$. By Lemma 5.9, it is enough to show that an error polynomial $r(Z)$ of degree at most three is bounded below as claimed. By Lemma 5.12, we may assume that all three roots of $r(Z)$ lie in $[0,1)$. Since $r(0)=0$ and $r(z)\ge 0$ in $[0,1)$, we may write $r(Z)=aZ(Z-\alpha)^2$ for some $\alpha\in[0,1)$ and some positive a; since $r(1)=1$, we conclude that $a=\frac{1}{(1-\alpha)^2}$. Thus, we need to determine the value of α so that $t(\alpha)=\max_{x\in\{\ell,u\}}\frac{r(x)}{x^3}$ is as small as possible. Consider the function $t_x(\alpha)=\frac{r(x)}{x^3}=\left(\frac{x-\alpha}{(1-\alpha)x}\right)^2$. Note that for all x, $t_x(\alpha)$ is monotonically increasing in $|x-\alpha|$. It follows that $t(\alpha)$ is minimum for some $\alpha\in[\ell,u]$. For α in this interval $t_\ell(\alpha)$ is an increasing function of α and $t_u(\alpha)$ is a decreasing function of α. So $t(\alpha)$ is minimum when $t_\ell(\alpha)=t_u(\alpha)$. It can be checked by direct computation that when $\alpha=\frac{2\ell u}{\ell+u}$,

$$t_\ell(\alpha)=t_u(\alpha)=\left(\frac{u-\ell}{u+\ell-2\ell u}\right)^2.$$

This establishes part (a) of Theorem 5.7.

To establish part (b), we show that an error polynomial of degree at most $2t+1$ satisfies the claim. As before, by Lemma 5.12, we may assume that $r(Z)$ has all its roots in $[0,1)$. Furthermore, since $r(Z) \geq 0$, we conclude that all roots in $(0,1)$ have even multiplicity. Thus we may write

$$r(Z) = \frac{Z(Z-\alpha_1)^2(Z-\alpha_2)^2\cdots(Z-\alpha_t)^2}{(1-\alpha_1)^2(1-\alpha_2)^2\cdots(1-\alpha_t)^2}.$$

Now, let $b = (\frac{u}{\ell})^{\frac{1}{t+1}}$. Consider subintervals $\{(\ell b^j, \ell b^{j+1}] : j = 0, 1, \ldots, t\}$. One of these intervals say $\ell b^{j_0}, \ell b^{j_0+1}$ has no roots at all. Let ϵ be the mid point of the interval, that is, $\epsilon = (\ell b^{j_0} + \ell b^{j_0+1})/2$. Then, we have

$$(\epsilon - \alpha_j)^2 \geq \left(\frac{\ell b^{j_0+1} - \ell b^{j_0}}{2}\right)^2$$

and since $(1-\alpha_j)^2 \leq 1$, we have

$$\frac{r(\epsilon)}{\epsilon^{2t+1}} \geq \left(\frac{b-1}{b+1}\right)^{2t}.$$

This establishes part (b). The term $-\frac{1}{N\ell(b+1)}$ appears in the statement of Theorem 5.7 because we need to ensure that ϵN is an integer. $\square$

5.4.2 Proof that the error function is a low degree polynomial

Proof of Lemma 5.9 We will use the following notation. Let $p(X_1, X_2, \ldots, X_N)$ be a polynomial in N variables $X_1, X_2, \ldots, X_N$ with real coefficients. For a database $f : \{0,1\}^N \rightarrow \{0,1\}$, let

$$p(f) \stackrel{\Delta}{=} p(f(1), f(2), \ldots, f(N)).$$

Also, in the following $\mathbf{X}$ denotes the sequence of variables $X_1, X_2, \ldots, X_N$.

The key fact we need is the following.

Theorem 5.13 ([BBC+98]). *Let $\mathcal{A}$ be a t-query quantum database search algorithm. Then, for $i = 1, 2, \ldots, N$, there is a multilinear polynomial $p_i(\mathbf{X})$ of degree at most $2t$, such that for all f.*

$$\Pr[\mathcal{A}(f) = i] = p_i(f).$$

Furthermore, $p_i(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in [0, 1]^N$.

Lemma 5.14. *Let $\mathcal{A}$ be a t-query quantum database search algorithm. Then, there is a multilinear polynomial $p(\mathbf{X})$ of degree at most $2t + 1$ such that for all f,*

$$\mathsf{err}_{\mathcal{A}}(f) = p_{\mathcal{A}}(f).$$

Proof. Using the polynomials $p_i(X)$ from Theorem 5.13, define

$$p_{\mathcal{A}}(\mathbf{X}) = \sum_{i=1}^{n} (1 - X_i) p_i(\mathbf{X}).$$

Clearly, $p(f) = \sum_{i=1}^{n} (1 - f(i)) p_i(f) = \sum_{i \in f^{-1}(0)} \Pr[\mathcal{A}(f) = i] = \mathsf{err}_{\mathcal{A}}(f)$. □

We can now prove Lemma 5.9. For a permutation σ of N and $f : [N] \to \{0, 1\}$, let σf be the function defined by $\sigma f(i) = f(\sigma(i))$.

Note that $|f^{-1}(0)| = |(\sigma f)^{-1}(0)|$. Now,

$$\frac{1}{N!} \sum_{\sigma} p_{\mathcal{A}}(\sigma f) = \mathop{\mathrm{E}}_{\sigma}[\mathsf{err}_{\mathcal{A}}(\sigma f)] \leq \max_{\sigma} \mathsf{err}_{\mathcal{A}}(\sigma f) \leq \mathsf{err}_{\mathcal{A}}(\epsilon), \tag{5.4}$$

where $|f^{-1}(0)| = \epsilon N$.

Let $\sigma \mathbf{X}$ be the sequence $X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(N)}$, and let

$$p^{\mathrm{sym}}_{\mathcal{A}}(\mathbf{X}) = \frac{1}{N!} \sum_{\sigma} p_{\mathcal{A}}(\sigma \mathbf{X}).$$

Then, by (5.4), we have $p_{\mathcal{A}}^{\text{sym}}(f) = \dfrac{1}{N!}\sum_{\sigma} p_{\mathcal{A}}(\sigma f) \le \mathsf{err}_{\mathcal{A}}(\epsilon)$.

Now, $p_{\mathcal{A}}^{\text{sym}}(\mathbf{X})$ is a symmetric multilinear polynomial in N variables of degree at most $2t+1$. For any such polynomial, there is a univariate polynomial $q(Z)$ of degree at most $2t+1$ such that if we let $\hat{p}(\mathbf{X}) = q(\sum_{i=1}^{N} X_i)/N)$, then for all f,

$$\hat{p}(f) = p_{\mathcal{A}}^{\text{sym}}(f) \le \mathsf{err}_{\mathcal{A}}(\epsilon).$$

(See Minsky and Papert [MP68].) Now, $\hat{p}(f) = q((f(1)+f(2)+\ldots+f(N))/N) = q(1-\epsilon)$. To complete the proof, we take $r(Z) = q(1-Z)$.

REFERENCES

[AFKS00] Noga Alon, Eldar Fischer, M. Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.

[AFNS06] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *Proceedings of the 38th ACM STOC*, pages 251–260, 2006.

[AS92] Noga Alon and J.H. Spencer. *The Probabilistic Method*. Wiley-Interscience (John Wiley & Sons), New York, 1992.

[AS03] Noga Alon and Asaf Shapira. Testing subgraphs in directed graphs. *STOC*, pages 700–709, 2003.

[AS05a] N. Alon and A. Shapira. A characterization of the (natural) graph properties testabale with one-sided error. *Proceedings of the 46th IEEE FOCS*, pages 429–438, 2005.

[AS05b] Noga Alon and Asaf Shapira. Linear equations, arithmetic progressions and hypergraph property testing. *SODA*, pages 708–717, 2005.

[Bab81] László Babai. On the order of uniprimitive permutation groups. *Annals of Mathematics*, 113:553–568, 1981.

[Bab82] László Babai. On the order of doubly transitive permutation groups. *Inventiones Math*, 65:473–484, 1982.

[BBC$^+$98] R. Beals, Harry Buhrman, R. Cleve, M. Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *FOCS*, pages 352–361, 1998.

[BC08] L. Babai and S. Chakraborty. Property testing of equivalence under a permutation group action. *Electronic Colloquium on Computational Complexity*, Report TR08-040, 2008.

[BCdWZ99] Harry Buhrman, R. Cleve, Ronald de Wolf, and C. Zalka. Bounds for small-error and zero-error quantum algorithms. *FOCS*, pages 358–368, 1999.

[BdW00] H. Burhman and Ronald de Wolf. Complexity measures and decision tree complexity: A survey. 2000.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BHMT02] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *S.J. Lomonaco and H.E. Brandt, editors, Quantum Computation and Information, AMS Contemporary mathematics Series*, 305:53–74, 2002.

[BL83] László Babai and E. M. Luks. Canonical labeling of graphs. *Proc. 15th STOC, ACM Press*, pages 171–183, 1983.

[BLR93] Manuel Blum, M. Luby, and Ronnit Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993.

[BLS87] László Babai, E.M. Luks, and Á. Seress. Permutation groups in nc. *Proc. 19th STOC, ACM Press*, pages 409–420, 1987.

[Cam81] Peter J. Cameron. Finite permutation groups and finite simple groups. *Bull. London Math. Soc.*, 13:1–22, 1981.

[CDR86] Stephen Cook, Cynthia Dwork, and Rudiger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J.Comput*, 15(1):87–97, 1986.

[CFL$^+$07] S. Chakraborty, E. Fischer, O. Lachish, A. Matsliah, and I. Newman. Testing *st*-connectivity. *11th International Workshop on Randomization and Computation*, 2007.

[Cha05a] S. Chakraborty. On the sensitivity of cyclically invariant boolean functions. *Computational Complexity Conference (CCC)*, 2005.

[Cha05b] Sourav Chakraborty. Sensitivity, block sensitivity and certificate complexity of boolean functions, masters thesis. Master's thesis, University of Chicago, 2005.

[CRR05] S. Chakraborty, J. Radhakrishnan, and N. Raghunathan. Bounds for error reduction with few quantum queries. *9th International Workshop on Randomization and Computation*, 2005.

[Fis04] Eldar Fischer. The art of uninformed decisions: A primer to property testing. *Current Trends in Theoretical Computer Science: The Challenge of the New Century, G. Paun, G. Rozenberg and A. Salomaa (editors), World Scientific Publishing*, I:229–264, 2004.

[Fis05] Eldar Fischer. The difficulty of testing for isomorphism against a graph that is given in advance. *SIAM Journal of Computing*, 34:1147–1158, 2005.

[FM06] Eldar Fischer and Arie Matsliah. Testing graph isomorphism. *SODA*, 2006.

[FNS04] Eldar Fischer, Ilan Newman, and J. Sgall. Functions that have read-twice constant width branching programs are not necessarily testable. *Random Structures and Algorithms*, 24:175–193, 2004.

[GGR98] Oded Goldreich, Saffi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[Gro96] L.K. Grover. A fast quantum mechanical algorithm for database search. *STOC*, pages 212–219, 1996.

[Gro97] L.K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325–328, 1997.

[Gro98a] L.K. Grover. A framework for fast quantum mechanical algorithms. *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, pages 53–63, 1998.

[Gro98b] L.K. Grover. Quantum computers can search rapidly by using almost any transformation. *Phy. Rev. Letters*, 80(19):4329–4332, 1998.

[Gro05] L.K. Grover. A different kind of quantum search. *quant-ph/0503205*, 2005.

[HLNT05] S. Halevy, Oded Lachish, Ilan Newman, and D. Tsur. Testing orientation properties. *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.

[HLNT07] S. Halevy, Oded Lachish, Ilan Newman, and D. Tsur. Testing properties of constraint-graphs. *Proceedings of the 22nd IEEE Annual Conference on Computational Complexity (CCC)*, 2007.

[KK04] Claire Kenyon and Samuel Kutin. Sensitivity, block sensitivity, and ℓ-block sensitivity of boolean functions. *Information and Computation*, 189(1):43–53, 2004.

[KKR04] Tali Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SICOMP*, 33(6):1441–1483, 2004.

[Luk82] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J.C.S.S.*, 25:42–65, 1982.

[MFNY08] Arie Matsliah, Eldar Fischer, Ilan Newman, and Orly Yahalom. On the query complexity of testing for eulerian orientations. *private communication*, 2008.

[MP68] M.L. Minsky and S.A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1968.

[NC00] M.A. Nielsen and I.L. Chuang. Quantum computation and quantum information. *Cambridge University Press*, 2000.

[New02] Ilan Newman. Testing membership in languages that have small width branching programs. *SIAM Journal on Computing*, 31(5):1557–1570, 2002.

[Nis91] Noam Nisan. Crew prams and decision trees. *SIAM J. Comput.*, 20(6):999–1070, 1991.

[NS94] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[PR02] M. Parnas and Dana Ron. Testing the diameter of graphs. *Random Structures and Algorithms*, 20(2):165–183, 2002.

[Ron01] Dana Ron. Property testing (a tutorial). *Handbook of Randomized Computing, (editors: S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim eds), Kluwer Press*, II, 2001.

[RS96] Ronnit Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996.

[Rub95] David Rubinstein. Sensitivity vs. block sensitivity of boolean functions. *Combinatorica*, 15(2):297–299, 1995.

[Sim83] Hans-Ulrich Simon. A tight $\omega(\log \log n)$-bound on the time for parallel ram's to compute nondegenerated boolean functions. *FCT, Lecture notes in Comp. Sci*, 4(158), 1983.

[Sun06] Xiaoming Sun. Block sensitivity of weakly symmetric functions. *Proceedings of Theory and Applications of Models of Computation*, pages 339–344, 2006.

[TGP05] T. Tulsi, L.K. Grover, and A. Patel. A new algorithm for directed quantum search. *quant-ph/0505007*, 2005.

[Tur84] Gyorgy Turán. The critical complexity of graph properties. *Inform. Process. Lett.*, 18:151–153, 1984.

[Weg87] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner Series in Computer Science (Wiley, New York), 1987.