Property Testing: Theory and Applications

by

Sofya Raskhodnikova

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author . . . .
Department of Electrical Engineering and Computer Science
May 22, 2003

Certified by . . . .
Michael Sipser
Professor of Mathematics
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Property Testing: Theory and Applications

by

Sofya Raskhodnikova

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2003, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Property testers are algorithms that distinguish inputs with a given property from those that are *far* from satisfying the property. *Far* means that many characters of the input must be changed before the property arises in it. Property testing was introduced by Rubinfeld and Sudan in the context of linearity testing and first studied in a variety of other contexts by Goldreich, Goldwasser and Ron. The query complexity of a property tester is the number of input characters it reads. This thesis is a detailed investigation of properties that are and are not testable with sublinear query complexity.

We begin by characterizing properties of strings over the binary alphabet in terms of their formula complexity. Every such property can be represented by a CNF formula. We show that properties of n-bit strings defined by 2CNF formulas are testable with $O(\sqrt{n})$ queries, whereas there are 3CNF formulas for which the corresponding properties require $\Omega(n)$ queries, even for adaptive tests.

We show that testing properties defined by 2CNF formulas is equivalent, with respect to the number of required queries, to several other function and graph testing problems. These problems include: testing whether Boolean functions over general partial orders are close to monotone, testing whether a set of vertices is close to one that is a vertex cover of a specific graph, and testing whether a set of vertices is close to a clique.

Testing properties that are defined in terms of monotonicity has been extensively investigated in the context of the monotonicity of a sequence of integers and the monotonicity of a function over the m-dimensional hypercube $\{1, \ldots, a\}^m$. We study the query complexity of monotonicity testing of both Boolean and integer functions over general partial orders. We show upper and lower bounds for the general problem and for specific partial orders.

A few of our intermediate results are of independent interest.

1. If strings with a property form a vector space, adaptive 2-sided error tests for the property have no more power than non-adaptive 1-sided error tests.
2. Random LDPC codes with linear distance and constant rate are not locally testable.
3. There exist graphs with many edge-disjoint induced matchings of linear size.

In the final part of the thesis, we initiate an investigation of property testing as applied to images. We study visual properties of discretized images represented by $n \times n$ matrices of binary pixel values. We obtain algorithms with the number of queries independent of n for several basic properties: being a half-plane, connectedness and convexity.

Thesis Supervisor: Michael Sipser
Title: Professor of Mathematics

Acknowledgments

The work included in this thesis was done in collaboration with Eli Ben-Sasson, Prahladh Harsha and Eric Lehman from MIT and Eldar Fischer, Ilan Newman, Ronitt Rubinfeld and Alex Samorodnitsky who were at NECI at the time. In particular, chapters 2–5 are co-authored with Eric, Eldar, Ilan, Ronitt and Alex, and chapters 6–7 are co-authored with Eli and Prahladh. I am indebted to all my other collaborators on research projects not included in this thesis and all colleagues at MIT, NECI and RSA Labs, who shared their knowledge, ideas and enthusiasm with me over the years.

I am grateful to my advisor, Michael Sipser, for indispensable discussions, advice, patience, and support. His wonderful course on theory of computation sparked my interest in theoretical computer science. Mike never stopped surprising me with his ability to make the most difficult concepts appear easy.

I would like to thank Shafi Goldwasser and Madhu Sudan for agreeing to be readers for this thesis. Conversations with Madhu Sudan helped tremendously with the material in this thesis. In particular, his suggestions lead to Theorem 6.5. His enthusiasm, the friendly and approachable atmosphere he creates around himself, and his readiness to help grad students in everything, from conference travel to job applications, made my life at MIT much easier.

Eric Lehman, who was at the time a fellow grad student at MIT, got me started on my first successful research project and got me interested in property testing.

I benefited greatly from an opportunity to learn from Piotr Indyk. I am grateful to him for his confidence in me and his advice on how to approach research.

I was extremely lucky to meet Ronitt Rubinfeld early in my graduate studies. She was my research advisor at NECI for three summers, and thanks to her mentoring, support, enthusiasm, valuable career and life advice, I have confidence to continue doing research.

Thank you to my family and friends who helped me throughout the years.

Contents

Chapter 1

Introduction

1.1 Property Testing

Practical computations are typically associated with polynomial time algorithms. However, with the emergence of the Internet, we are often faced with massive datasets which are so huge that in some scenarios it is impractical to read every single bit of them. Examples of such large datasets include all documents in the World Wide Web, measurements from scientific experiments, the human DNA sequence of the 3.2 billion base pairs recently completed by the genome project, high-resolution images and census data. Given that reading some data takes too long, it is natural to ask what properties of the data can be detected by *sublinear* algorithms that read only a small portion of the data.

Since, in general, most problems are not solvable exactly and deterministically with this restriction, the question becomes more interesting when sublinear algorithms considered are allowed to be probabilistic and approximate. The traditional notion of approximation requires that the output should be close to the desired value. This thesis studies *decision problems*, where the desired answer is either *accept* or *reject*. For such problems, the traditional notion of approximation is meaningless because there are only two possible answers. A notion of approximation tailored for decision problems, called *property testing*, was introduced by Rubinfeld and Sudan [RS96] in the context of testing of linearity of functions. It was first applied to combinatorial objects, such as graphs, by Goldreich, Goldwasser and Ron [GGR98]. Property testing approaches approximation of decision problems by transforming languages into *promise problems* [ESY84] where certain inputs are excluded from consideration.

A standard probabilistic algorithm accepts positive instances (that have the property) and rejects negative instances (that do not have the property) with high probability. A property testing algorithm is still required to accept positive instances with high probability[1]. However, only those negative instances that are *far* away from every positive instance should be rejected with high probability[1]. How *far* is dictated by a *distance parameter* ε, which plays a role similar to that of an *approximation factor* in standard approximation algorithms. Two instances are ε*-far* if they differ on an ε-fraction of their characters.[2] Thus, we exclude from consideration negative instances which are *borderline*, that is, close to positive instances. We cannot hope that a sublinear algorithm will find the few places on which borderline instances differ from positive instances. Allowing arbitrary errors on borderline instances enables us to design much more efficient tests.

Property testing algorithms, as we defined them, can in general have *2-sided error* and be *adaptive*, namely, they can make input queries that depend on answers to previous queries. An algorithm has *1-sided error* if it always accepts an input that has the property. An algorithm is *non-adaptive* if it makes all input queries in advance, before getting the answers. In addition to possible practical significance, 1-sided error and non-adaptivity are useful theoretical tools for obtaining hardness results.

Property testing algorithms offer several benefits: they save time, are good in settings where some errors are tolerable and where the data is constantly changing, and can also provide a fast sanity check to rule out bad inputs. They are also useful as a theoretical tool: for example, linearity testers have been used in PCP constructions [RS96]. An additional motivation for studying property testing is that this area is abundant with fascinating combinatorial problems. Property testing has recently become an active research area. An interested reader is referred to surveys [Ron01, Fis01] on the topic.

1.2 Characterizing Testable Properties

One of the important questions in property testing is characterizing properties that can be tested with a sublinear number of queries into the input. A series of works identified classes of properties testable with constant query complexity (dependent only on distance

[1] In this paragraph, "high probability" means "probability at least 2/3". However, it can always be amplified to $1-\mu$ by running $O(\log(1/\mu))$ iterations of the algorithm and outputting the majority function of its answers.

[2] This corresponds to the Hamming distance.

parameter ε). Goldreich et al. [GGR98] identified many such graph properties. Examples include being k-colorable and having a clique with ρ fraction of the nodes. Alon et al. [AKNS01] proved that all regular languages are testable with constant complexity. Newman [New02] extended their result to properties that can be computed by oblivious read-once constant-width branching programs. Fischer and Newman [FN02] demonstrated a property computable by a read-twice constant-width branching program which required super-constant query complexity, thus showing that Newman's result does not generalize to read-twice branching programs. Several papers [AFKS00, Fis01a] worked on the logical characterization of graph properties testable with a constant number of queries.

One of the goals of this thesis is to characterize properties testable with a sublinear number of input queries. We first attempt characterization of properties over the binary alphabet, $\{0, 1\}$, in terms of formula complexity. For every fixed property, the set of strings of length n with that property (as any set of binary strings of length n) can be represented by a truth table on n variables, where each variable corresponds to a position in the string. We can write out a Boolean formula corresponding to the truth table and convert it to the conjunctive normal form. Thus, every property over the binary alphabet can be represented by a CNF formula. Testing that property can be viewed as testing whether a given assignment to Boolean variables of the corresponding CNF is close to one that satisfies the formula. Goldreich et al. [GGR98] prove that there exist properties over the binary alphabet which require testing algorithms to read a linear portion of the input. This implies that testing assignments to general CNF formulas is hard. A natural question is whether restricting CNF formulas to have a constant number of variables k per clause allows for faster testers. At first sight, there is hope for obtaining good testers for every fixed k because for any assignment that does not satisfy the formula there exists a set of k queries that witnesses this fact. Moreover, the exact version of the problem is easy. However, we will show that already for $k = 3$ testing whether an input assignment is close to satisfying a fixed kCNF formula might require a linear number of queries.

Observe that in our testing question a CNF formula is *not* part of the input; instead, it describes the property. Our problem is different from testing whether an input kCNF formula is satisfiable. The exact version of this problem is a classical NP-complete problem and the property testing version was studied by Alon and Shapira [AS02]. They showed that testing satisfiability of kCNF formulas can be done with complexity independent of

Problem	Exact	Testing
Is a 3-CNF satisfiable? (3SAT)	NP-complete	easy
Does an assignment satisfy a fixed 3-CNF?	easy	hard

Figure 1-1: The world of property testing is different from the world of exact problems.

the input size. By contrast, our problem is very easy in its exact version, but hard in its property testing version for $k \geq 3$. Figure 1-1 contrasts our 3CNF testing question with testing satisfiability of 3CNF formulas [AS02].

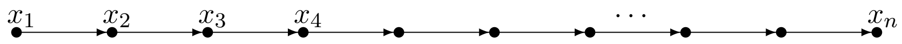
1.3 Testing 2CNF Properties

A *property* is a collection of n-character strings. Informally, one can think of this collection as strings that satisfy some property. A natural place to start investigating testability of properties corresponding to kCNF formulas is $k = 1$. This turns out to be very easy as 1CNFs can only describe sets of strings with certain positions fixed. Testing whether an input string is in such a set or is ε-far from it can be done by querying the input on $1/\varepsilon$ random positions fixed by the 1CNF formula and rejecting if one of them is not set to the right value.

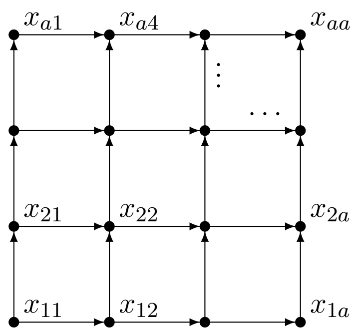
2CNFs offer a richer set of properties. Chapter 3 describes some natural properties which are equivalent to properties describable by 2CNF formulas with respect to the number of queries required for testing. These results provide additional motivation for studying 2CNF properties. The equivalent properties include being a vertex cover in a graph fixed in advance, being a clique in a graph fixed in advance and monotonicity of graph labelings over the binary alphabet. In the latter problem, each character of the input is viewed as a label of the corresponding node of a fixed directed n-node graph. A labeling of the graph is *monotone* if labels are non-decreasing along every edge. In general, labelings can come from an arbitrary alphabet. The monotonicity property for a given graph is a collection of all monotone labelings for that graph. Every directed graph[3] defines a property. Conversely, every problem, where we have to determine whether the input satisfies some partial ordering,

[3] The graph does not have to be acyclic. If a labeling is monotone for a cyclic graph, all nodes in a cycle must have the same labels.

corresponds to a graph. For example, a line where all edges are pointing in the same direction defines a property of sorted labels:



A two-dimensional grid with edges pointing up and to the right defines a property where all labels are sorted along all rows and columns:



In fact, every partially-ordered set (poset) can be represented by a directed graph. Monotone functions on a poset correspond to monotone labelings for the corresponding graph.

In the context of property testing, monotonicity of functions was first considered by Goldreich et al. [GGL$^+$00]. Before that, a special case of this problem, testing whether a list is sorted, was studied by Ergün et al. [EKK$^+$98]. Along with linearity and low-degree testing, monotonicity is one of the more studied properties in the context of property testing (see also [DGL$^+$99, BRW99, Ras99, FN01, Fis]).

After proving that monotonicity of graph labelings over the binary alphabet is equivalent to 2CNF properties with respect to the number of queries required for testing, we proceed to analyze testability of monotonicity properties. We choose to work with monotonicity properties instead of equivalent 2CNF properties for two reasons. First, graphs provide a useful pictorial tool and are easier to work with. Second, monotonicity is a more general property, as it is not restricted to the binary alphabet.

All previous monotonicity results deal with posets that happen to be hypercubes of different sizes and dimensions. Chapter 4 gives an algorithm and a lower bound for general graphs. Our test for monotonicity of labelings on graphs with n nodes queries $O(\sqrt{n/\varepsilon})$ input characters, and works for all alphabets, addressing an open problem posed by [DGL$^+$99, Ras99]. This, in turn, yields $O(\sqrt{n/\varepsilon})$ query tests for all equivalent properties, including 2CNF properties. We then show that there are graphs for which no non-adaptive monotonicity test makes only a polylogarithmic number of queries. To be more precise, the non-adaptive lower bound is $n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}$ queries. This implies an adaptive lower bound of $\Omega\left(\frac{\log 1/\varepsilon \log n}{\log\log n}\right)$. The lower bounds are for the binary alphabet case, which is equivalent to 2CNF properties and other properties from Chapter 3.

To achieve our non-adaptive lower bound, we prove that on graphs with m edge-disjoint induced matchings of linear size, every non-adaptive monotonicity test makes $\Omega(\sqrt{m})$ queries. Then we prove that there are n-node graphs that can be partitioned into $n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}$ induced matchings of size εn.

We call graphs that can be partitioned into many large induced matchings *Ruzsá-Szemerédi graphs.* The reason for this name is that Ruzsá and Szemerédi [RS78] constructed such graphs for one extreme setting of parameters: with $n/3$ matchings of near-linear size. Their graphs are based on Behrend's construction [Beh46] of sets of integers that contain no three terms in arithmetic progression. Ruzsá and Szemerédi employed their construction in a lower bound proof for a Turán-like theorem. Recently, Håstad and Wigderson [HW01] used the graphs of Ruzsá and Szemerédi for improving linearity tests[4]. Motivated by conjectures in graph theory, Roy Meshulam [Personal communication] constructed similar graphs with different parameters.

Our approach is different from those of Ruzsá and Szemerédi and of Meshulam. We present four constructions, starting with a very intuitive one with relatively weak parameters and finishing with a technically difficult one with the desired parameters. The constructions improve on each other, gradually introducing new ideas. They all give bipartite Ruzsá-Szemerédi graphs. In the simplest construction, nodes in each layer are associated with binary strings of length $\log n$. A node from the first layer is connected to a node in the second layer if they differ in exactly one coordinate. This construction yields a logarithmic number of matchings of linear size. In the final construction, the strings associated with the nodes are over more general alphabets, and nodes are matched based on their values in a subset of coordinates. Improving our final construction of Ruzsá-Szemerédi graphs would lead to better lower bounds for monotonicity tests and, possibly, to improvements in the above-mentioned applications. Other applications are likely because Ruzsá-Szemerédi graphs are easily describable combinatorial objects.

Chapter 5 deals with algorithms and lower bounds for specific graphs, such as the

[4]Substituting the graphs we construct for the graphs of Ruzsá and Szemerédi in [HW01] yields a family of linearity tests. These tests are incomparable to these of Håstad and Wigderson, they could be better or worse depending on the distance of the tested function to the closest linear function.

hypercube. We give the first non-trivial lower bound for monotonicity testing on the well-studied hypercube. The question arises as to what graphs, besides the hypercube, can be tested more efficiently than the general lower bound. We find many families of graphs for which monotonicity is testable with a small number of queries. Different families use different combinatorial techniques.

1.4 Testing kCNF Properties For $k \geq 3$

Testing kCNF properties becomes hard for $k = 3$. We present a gap between 2CNFs and 3CNFs by showing the existence of families of 3CNF formulas which require a *linear* number of queries. Our lower bound applies to *adaptive* tests, i.e. tests where queries might depend on the answers to previous queries. This gives a class of properties which are easy to decide exactly (linear time), but are hard to test.

The hard 3CNF properties we find are vector spaces. As the first step towards proving the hardness result, Chapter 6 shows that every adaptive 2-sided error test for checking membership in a vector space can be converted to a non-adaptive 1-sided error test with the same query complexity and essentially identical parameters. This result applies to properties over arbitrary alphabets and is of independent interest. It allows us to consider only 1-sided error non-adaptive tests. Chapter 7 gives sufficient conditions for a vector space to be hard to test, and then proves that random Low Density Parity Check Codes (LDPC codes) satisfy these conditions. Finally, it shows how to convert the resultant codes to vector spaces expressible by 3CNF formulas.

Our results shed some light on the question of the existence of locally testable codes with linear distance and constant rate. An infinite family of codes $\{\mathcal{C}\}_n$ is called *locally testable* if the property $\mathcal{C}_n$ is testable with constant query complexity. Locally testable codes play a vital role in PCP constructions, and are of fundamental importance in theoretical computer science. Recently Ben-Sasson et al. [BSVW03], following the work of Goldreich and Sudan [GS02], gave an explicit construction of such codes which achieve linear distance and near constant rate, resulting in better PCP constructions.

The vector spaces we use (which are hard to test) are built upon random (c, d)-regular LDPC codes. These codes, introduced by Gallager [Gal63], are known to achieve linear distance and constant rate. We show that this important class of codes is not locally testable by a long shot. Moreover, the property that makes random codes so good in terms of minimal distance, namely expansion, is also behind the poor testability of these codes. Our techniques might be useful in proving that locally testable codes with linear distance and constant rate do not exist. Whether or not such codes exist remains an interesting open problem.

1.5 Testing Visual Properties

In the last chapter of the thesis, we propose to apply property testing to images. Image analysis is one area potentially well suited to the property testing paradigm. Images contain a large amount of data which must be analyzed quickly in a typical application. Some salient features of an image may be tested by examining only a small part thereof. Indeed, one motivation for this study is the observation that the eye focuses on relatively few places within an image during its analysis. The analogy is not perfect due to the eye's peripheral vision, but it suggests that property testing may give some insight into the visual system.

We present algorithms for a few properties of images. All our algorithms have complexity independent of the image size, and therefore work well even for huge images. We use an image representation popular in learning theory (see, e.g., [MT89]). Each image is represented by an $n \times n$ matrix of pixel values. We focus on black and white images given by matrices with entries in $\{0, 1\}$ where 0 denotes white and 1 denotes black.

We present tests for three visual properties: being a half-plane, convexity and connectedness. The algorithm for testing if the input is a half-plane is a 1-sided error test with $\frac{2\ln 3}{\varepsilon} + o(\frac{1}{\varepsilon})$ queries. The convexity test has 2-sided error and makes $O(1/\varepsilon^2)$ queries. Finally, the connectedness test has 1-sided error and makes $O\left(\frac{1}{\varepsilon^2}\log^2\frac{1}{\varepsilon}\right)$ queries.

In some sense, it is amazing that such global properties as connectedness can be tested with the number of queries independent of the size of the image. Marvin Minsky and Seymour Papert write in their book "Perceptrons" [MP69]:

> We chose to investigate *connectedness* because of a belief that this predicate is nonlocal in some very deep sense; therefore it should present a serious challenge to any basically local, parallel type of computation.

1.6 Organization of this thesis

The rest of the thesis is organized as follows.

Chapter 2 – Preliminaries. We introduce basic definitions and general tools.

Chapter 3 – Properties Equivalent to 2CNF Properties. We show that testing 2CNFs is equivalent to a few other testing problems, such as testing monotonicity of graph labelings over binary alphabet, testing whether a set of vertices is a vertex cover for a fixed graph and testing whether a set of vertices is a clique in a fixed graph.

Chapter 4 – Bounds on Query Complexity for Monotonicity Testing. We present a 1-sided error algorithm with an $O(\sqrt{n/\varepsilon})$ query complexity for testing monotonicity over general graphs with n nodes. This, in turn, yields 1-sided error $O(\sqrt{n/\varepsilon})$ query tests for all equivalent properties from Chapter 3. The corresponding hardness result is a lower bound of $n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}$ queries for general Boolean non-adaptive monotonicity testing. This implies an adaptive lower bound of $\Omega\left((\log 1/\varepsilon)\frac{\log n}{\log\log n}\right)$. The hardness proof shows that graphs with many edge-disjoint matchings of linear size are hard to test. We call these graphs *Ruzsá-Szemerédi* graphs. The last section is devoted to constructions of Ruzsá-Szemerédi graphs and a discussion of which parameters for those graphs are currently attainable.

Chapter 5 – Monotonicity Testing on Special Graphs. We present lower bounds for non-adaptive monotonicity tests over the (Boolean) m-dimensional hypercube. We prove an $\Omega(\sqrt{m})$ lower bound for 1-sided error algorithms, and an $\Omega(\log m)$ lower bound for 2-sided error algorithms. These results imply the corresponding adaptive lower bounds of $\Omega(\log m)$ and $\Omega(\log\log m)$, respectively.

This Chapter also contains efficient algorithms for testing monotonicity on several special classes of graphs. We show that certain types of graphs have monotonicity tests with a number of queries that is independent of the size of the graph. For Boolean labelings, this includes graphs whose undirected versions are trees, graphs of constant width, and what we call *top-parallel* graphs. For labelings with arbitrary ranges, this applies to graphs with a linear number of edges in the transitive closure. We also prove that for graphs with bounded separators, monotonicity testing of labelings with arbitrary ranges requires only a logarithmic number of queries.

Chapter 6 – Testing Vector Spaces. A property is *linear* if it forms a vector space.

We prove that in the context of linear property testing, adaptive 2-sided error tests have no more power than non-adaptive 1-sided error tests. The reduction to simpler tests proceeds in two stages. First, we convert adaptive 2-sided error tests for linear properties to adaptive 1-sided error ones by shifting all error probability to one side and preserving the query complexity of the test. Then we convert adaptive 1-sided error tests for linear properties to non-adaptive 1-sided error tests, keeping all parameters unchanged.

Chapter 7 – Some 3CNF Properties Require a Linear Number of Queries. We prove that there are 3CNF properties that require a linear number of queries, even for adaptive tests. We work with linear properties and later show how to represent the hard properties we find by 3CNF formulas. Working with linear properties allows us to use the reduction from Chapter 6 and concentrate on non-adaptive 1-sided error tests. We provide sufficient conditions for linear properties to be hard to test, and prove that random linear LDPC codes with linear distance and constant rate satisfy these conditions. Then we show how to convert them to properties representable with 3CNF formulas.

Chapter 8 – Testing Visual Properties. We propose to explore property testing as applied to visual properties of images. We study visual properties of discretized images represented by $n \times n$ matrices of 0–1 pixel values and obtain algorithms with the number of queries independent of n for several basic properties: being a half-plane, connectedness and convexity.

Chapter 9 – Conclusion and Open Problems. We make some concluding remarks and discuss directions for further research.

Bibliographic note. The results on monotonicity testing appearing in Chapters 2–5 were co-authored with Eldar Fischer, Eric Lehman, Ilan Newman, Ronitt Rubinfeld and Alex Samorodnitsky [FLN$^+$02]. The material on testing vector spaces and testing 3CNF formulas in Chapters 6 and 7 is a result of collaboration with Eli Ben-Sasson and Prahladh Harsha [BHR03]. The second reduction in Chapter 6 was suggested by Madhu Sudan. The chapter on testing visual properties will also appear as a separate paper [Ras]. Looking at visual properties in the context of property testing was suggested by Michael Sipser.

Chapter 2

Preliminaries

Property testing

A *property* is a collection of strings of a fixed size n over a fixed alphabet L. The distance $dist(x, \mathcal{P})$ of a string x to a property $\mathcal{P}$ is $\min_{x' \in \mathcal{P}} dist(x, x')$, where $dist(x, x')$ denotes the Hamming distance between the two strings. The *relative distance* of x to $\mathcal{P}$ is its distance to $\mathcal{P}$ divided by n. A string is ε-far from $\mathcal{P}$ if its relative distance to $\mathcal{P}$ is at least ε.

A *test for property $\mathcal{P}$ with distance parameter ε, positive error μ_+, negative error μ_- and query complexity q* is a probabilistic algorithm that queries at most q characters of the input, rejects strings in $\mathcal{P}$ with probability at most μ_+ and accepts strings that are ε-far from $\mathcal{P}$ with probability at most μ_-, for some $0 \leq \mu_+, \mu_- \leq 1$. A test is said to have *error* μ if $\mu_+ \leq \mu$ and $\mu_- \leq \mu$ (for $\mu < \frac{1}{2}$). If a test T accepts input x, we say $T(x) = 1$. Otherwise, we say $T(x) = 0$. A test with distance parameter ε, positive error μ_+, negative error μ_- and query complexity q is referred to as an $(\varepsilon, \mu_+, \mu_-, q)$*-test.* Often, we want to bound the error of the test by a small constant $\mu < 1/2$. The exact value of the constant does not matter, as the error can always be made smaller without affecting the asymptotics of other parameters by standard probability amplification techniques. When μ is not explicitly specified, we mean $\mu = 1/3$. In particular, (ε, q)-test is a test with distance parameter ε, query complexity q and error $2/3$, unless specified otherwise. An ε-test denotes a test with distance parameter ε. A property is (ε, q)*-testable* if it has an (ε, q)-test; $(\varepsilon, \mu_+, \mu_-, q)$*-testable* is defined analogously.

A couple of special classes of tests are of interest. An algorithm is *non-adaptive* if it asks all queries in advance, before getting the answers. Namely, a query may not depend on the answers to previous queries. An algorithm has *1-sided error* if it always accepts an input that has the property (soundness $s = 1$).

CNF formulas

Recall that a Boolean formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, where every clause is a disjunction of literals. (A literal is a Boolean variable or a negated Boolean variable.) If all clauses contain at most three literals, the formula is a 3CNF.

Let φ be a formula on n variables. An n-bit string *satisfies* φ if it satisfies all clauses of the formula. An n-bit string is *ε-far* from satisfying φ if at least an ε fraction of the bits need to be changed to make the string satisfy φ. Each formula φ defines a property $SAT(\varphi) = \{x|\ x \text{ satisfies } \varphi\}$. For brevity, we refer to a test for this property as a test for φ.

Monotonicity

We defined *property* as a collection of n-character strings over alphabet L. Sometimes it is useful to think of these strings as functions over a fixed domain V with n elements. Each string $x \in L^n$ can be represented by a function $f : V \to L$ where x is the concatenation of evaluations of f on all elements of V. For example, an assignment $x_1, \ldots, x_n$ to variables $X_1, \ldots, X_n$ can be treated both as a string $x_1 \ldots x_n \in \{0,1\}^n$ and as a function $f : \{X_1, \ldots, X_n\} \to \{0,1\}$. Properties over $L = \{0,1\}$ alphabet are called Boolean because they correspond to Boolean functions.

Let $G = (V, E)$ be a directed graph. Let $f : V \longrightarrow L$ be a labeling of V with members of a linear order L. Then f is *monotone* on G if $f(v) \leq f(u)$ for all $(v, u) \in E$. The monotonicity property, denoted by $MON(G)$, is the set of monotone labelings of G. If there is a directed path from v to u in G, we say that v is *below* u (or u is *above* v) and denote it by $v \leq_G u$ (which is not an order relation in general). Every such pair of vertices of G imposes a constraint $f(v) \leq f(u)$. A pair of vertices (v, u) is *violated* if $v \leq_G u$ and $f(v) > f(u)$. Vertices v and u are *equivalent* in G if $v \leq_G u$ and $u \leq_G v$, namely, if both are in the same strongly connected component.

Note that monotonicity of labelings of acyclic graphs corresponds to monotonicity of functions on posets. We often consider a special case of monotonicity restricted to Boolean functions or labelings (namely, with $L = \{0,1\}$), which we call Boolean monotonicity.

Handy lemmas for monotonicity testing

A transitive closure of a graph $G = (V, E)$, denoted by $\mathrm{TC}(G)$, is a graph (V, E') where (v_1, v_2) is in E' if there is a directed path from v_1 to v_2 in G. Observe that two graphs with the same transitive closure give rise to the same monotonicity property $MON(G)$.

Lemma 2.1 *Let f be a labeling for a graph $G(V, E)$. If f is monotone on an induced subgraph $G' = (V', E')$ of* $\mathrm{TC}(G)$*, then f's distance to monotone is at most $|V - V'|$.*

Proof. Assuming that f is monotone on an induced subgraph $G' = (V', E')$ of $\mathrm{TC}(G)$, we make f monotone on $\mathrm{TC}(G)$ by relabeling only vertices in $V - V'$.

Indeed, fix V' and let $f|_{V'}$ be the partial labeling on V' that is monotone on $G'(V', E')$. We extend $f|_{V'}$ to V for one vertex $v \in V - V'$ at a time, always keeping the partial labeling monotone on the induced current graph. We now show how to extend the domain of f by one vertex. Let $v \in V - V'$ be a 'minimal' element in $V - V'$ (namely, v is unreachable from any other vertex $w \in V - V'$ that is not equivalent to it). Let $T = \{u \in V' |\ u \leq_G v\}$. We extend f to $V \cup \{v\}$ by letting $f(v)$ be $\max_{u \in T}\{f(u)\}$ if $T \neq \emptyset$ and the minimum value in the range otherwise. By transitivity, since f was monotone on V', the extended f is monotone on $V \cup \{v\}$. $\square$

Corollary 2.2 *Let f be a labeling of $G = (V, E)$. Then $dist(f, MON(G))$ is equal to the minimum vertex cover of the graph of violated edges of* $\mathrm{TC}(G)$.

A *matching* in a graph is a collection of edges that share no common vertex. The next two lemmas relate a function's distance to monotone to the number of edges it violates in the transitive closure of the graph. The first of them follows from Corollary 2.2 and the fact that the size of a maximum matching is at least $1/2$ of the size of a minimum vertex cover.

Lemma 2.3 ([DGL$^+$99]) *Let f be a labeling which is ε-far from monotone on a graph G with n nodes. Then* $\mathrm{TC}(G)$ *has a matching of violated edges of size $\varepsilon n/2$.*

Lemma 2.4 *Let f be a Boolean labeling which is ε-far from monotone over a graph G with n nodes. Then* $\mathrm{TC}(G)$ *has a matching of violated edges of size εn.*

Proof. Let P' be a poset of vertices in V with partial order defined by $v \leq u$ if (v, u) is a violated pair in G. Let $A \subseteq V$ be a maximal antichain in P'. Certainly, f is monotone on the subgraph of TC (G) induced by A, as A contains no violated pairs. Then by Lemma 2.1, $dist(f, MON(G)) \leq |V| - |A|$. By Dilworth's theorem [Dil50], $|A|$ is equal to the minimum number of disjoint chains that cover P'. However, a chain in P' consists of at most two vertices as (v, u) and (u, w) cannot be both violated by a Boolean function. Hence, to cover $|V|$ elements, at least $|V| - |A|$ out of $|A|$ chains have to be of length exactly two (otherwise, less than $|V|$ elements are covered). This collection of at least $|V| - |A| \geq dist(f, MON(G))$ disjoint chains of size two is a matching of violated pairs. $\square$

Chapter 3

Properties equivalent to 2CNF properties

In this chapter, we show that 2CNF properties are equivalent with respect to the number of queries required for testing to a few other classes of properties. The first one is Boolean monotonicity of graph labelings. In the second class, each property is a set of vertex covers for a fixed graph. Each graph gives rise to a property of binary strings corresponding to vertex covers of that graph, with each bit representing whether the corresponding vertex is in the cover. Here distance captures the number of vertices that need to be added to make the set into a vertex cover. In the third class, each property is a set of cliques for a fixed graph. Here distance refers to the number of vertices that need to be removed to make the set into a clique. We also show that testing *monotone* 2CNFs (i.e., 2CNFs with only positive literals) is as hard as the general 2CNF testing problem. In addition, for all labeling alphabets, testing monotonicity on bipartite graphs $(X, Y; E)$ with $|X| = |Y|$ and all edges directed from X to Y is as hard as monotonicity testing on general graphs. The last result will be used in the presentation of our monotonicity test of Chapter 4.

Our proofs of equivalence are reductions that transform instances of one problem into another, so that the tests for the second problem could be used for the first. Intuitively, we need three features from our reductions. First, they have to transform positive instances into positive instances. Second, negative instances (i.e. instances which are ε-far from having the property) should be transformed into instances which are reasonably far from having the property. Our reductions change distances by at most a constant factor. Third, every query for the original problem should be computable from a constant number of queries for the new problem. Notice that the first requirement ensures that a 1-sided error test for the new problem yields a 1-sided error test for the original problem. The third requirement guarantees that a non-adaptive test for the new problem can be converted to a non-adaptive test for the original problem.

3.1 Boolean monotonicity is equivalent to 2CNF properties

We start by showing that Boolean monotonicity reduces to 2CNF properties.

Theorem 3.1 *For every graph G with n vertices, there is a corresponding* 2CNF φ_G *on n variables such that if $SAT(\varphi_G)$ is $(\varepsilon, \mu_+, \mu_-, q)$-testable then $MON(G)$ is also $(\varepsilon, \mu_+, \mu_-, q)$-testable for Boolean labelings.*

Proof. Let $G = (V, E)$ be a directed graph. With each $v \in V$ associate a Boolean variable x_v. Define the 2CNF formula φ_G on the set of variables $X = \{x_v |\ v \in V\}$ as follows: for each edge $(u, v) \in E$, form the clause $(\overline{x_u} \vee x_v)$. A Boolean labeling f on $V(G)$ defines an assignment $\tilde{f}$ on X by $\tilde{f}(x_v) = f(v)$. Clearly, $dist(f, MON(G)) = dist(\tilde{f}, SAT(\varphi_G))$. Thus, a test for $SAT(\varphi_G)$ can be used as a test for $MON(G)$. $\square$

Our next step is the reduction in the other direction.

Theorem 3.2 *For every* 2CNF φ *on n variables, there is a corresponding graph G_φ with $2n$ vertices such that if $MON(G_\varphi)$ is $(\varepsilon/2, \mu_+, \mu_-, q)$-testable for Boolean labelings then $SAT(\varphi)$ is $(\varepsilon, \mu_+, \mu_-, q)$-testable.*

Proof. Let φ be a satisfiable 2CNF formula on a set X of n variables. (If φ is unsatisfiable, it has a trivial test that rejects all assignments). With each Boolean variable $x \in X$, associate two vertices v_x and $v_{\overline{x}}$ that represent literals corresponding to x. We use the convention $v_x = \overline{v_{\overline{x}}}$ and $v_{\overline{x}} = \overline{v_x}$. Define the *implication graph*, G_φ, on the set of the corresponding $2n$ vertices, as follows: for each clause $x \vee y$, where x and y are literals, add edges $(\overline{v_y}, v_x)$ and $(\overline{v_x}, v_y)$. For any edge (u, v) call edge $(\overline{v}, \overline{u})$ its *dual* edge. Note that dual edges appear in the implication graph in pairs, with the exception of edges of the form $(u, \overline{u})$, which are dual to themselves.

Let $f : X \to \{0,1\}$ be an assignment to φ. Define the associated Boolean labeling f_G of G_φ by $f_G(v_x) = f(x)$ for all literals x. If f satisfies φ, the corresponding labeling f_G is monotone on G_φ. It remains to prove that $dist(f, SAT(\varphi)) \le dist(f_G, MON(G_\varphi))$. To show this we transform f into a satisfying assignment for φ by changing at most $dist(f_G, MON(G_\varphi))$ variable assignments. To this end, a Boolean labeling of an implication graph is called *negation-compliant* if v_x and $\overline{v_x}$ have different labels for all literals x. Note that every negation-compliant labeling of G_φ has a corresponding assignment to φ. Furthermore, if $\tilde{f}$ is monotone and negation-compliant for G_φ then the corresponding assignment f for φ, given by $f(x) = \tilde{f}(v_x)$ for every literal x, is a satisfying assignment for φ.

Note that for every literal x, v_x and $\overline{v_x}$ are never in the same strongly connected component because φ is satisfiable. Also, if v_x is equivalent to v_y in G_φ then $\overline{v_x}$ is equivalent to $\overline{v_y}$.

The following algorithm transforms f_G into a nearby monotone, negation-compliant labeling.

1. Convert f_G to a nearest monotone assignment $\tilde{f}_G$ on G_φ. ($\tilde{f}_G$ is not necessarily negation-compliant.)

2. While G_φ has nodes v_x with $\tilde{f}_G(v_x) = \tilde{f}_G(\overline{v_x}) = 0$:
   Find a maximal v_x (with respect to G_φ) among those with $\tilde{f}_G(v_x) = \tilde{f}_G(\overline{v_x}) = 0$. Change $\tilde{f}_G(v_z)$ to 1 for all v_z that are equivalent to v_x (including v_x itself).

3. While G_φ has nodes v_x with $\tilde{f}_G(v_x) = \tilde{f}_G(\overline{v_x}) = 1$:
   Find a minimal v_x among those with $\tilde{f}_G(v_x) = \tilde{f}_G(\overline{v_x}) = 1$. Change $\tilde{f}_G(v_z)$ to 0 for all v_z that are equivalent to v_x (including v_x itself).

First, we show that the resulting labeling $\tilde{f}_G$ is monotone on G_φ. Indeed, $\tilde{f}_G$ is monotone after step 1. Since it is monotone, nodes in the same strongly connected component (i.e., equivalent nodes with respect to G_φ) have the same labels. Hence, after each change in step 2, equivalent nodes still have the same labels. Suppose $\tilde{f}_G$ is monotone on G before some iteration of step 2 and is not monotone after it. Then some edge (v_x, v_y) is violated by changing $\tilde{f}(v_x)$ to 1. Then $\tilde{f}_G(v_y) = 0$ both before and after this iteration, and v_y is not equivalent to v_x. Since $v_y \ge_G v_x$, it must be that $\tilde{f}_G(\overline{v_y}) = 1$ (otherwise, v_y would have changed before v_x). But then the dual edge $(\overline{v_y}, \overline{v_x})$ is violated before the iteration, giving a contradiction.

Similarly, if $\tilde{f}_G$ is monotone on G before some iteration of step 3 then it is monotone after it.

Secondly, the resulting labeling $\tilde{f}_G$ is negation-compliant because step 2 relabels all nodes v_x with $\tilde{f}(v_x) = \tilde{f}(\overline{v_x}) = 0$, and step 3 relabels all nodes with $\tilde{f}(v_x) = \tilde{f}(\overline{v_x}) = 1$.

Finally, let $\tilde{f}$ be the assignment to X with $\tilde{f}(x) = \tilde{f}_G(v_x)$ for every literal $x \in X$. By the remarks above, $\tilde{f}$ is a satisfying assignment for φ. To calculate $dist(f, \tilde{f})$, note that f and $\tilde{f}$ may disagree for a variable x only if $\tilde{f}_G(v_x) \neq f_G(v_x)$, where $\tilde{f}_G$ is the outcome of the algorithm above. Let $\tilde{f'}_G$ be the labeling resulted after step 1.

Now, step 1 modifies f_G on $dist(f_G, MON(G_\varphi))$ places which correspond to a set of variables $D = \{x | \tilde{f'}_G(v_x) \neq f_G(v_x),\ or\ \tilde{f'}_G(\overline{v_x}) \neq f_G(\overline{v_x})\}$. Successive steps change $\tilde{f'}_G$ on v_x only if $\tilde{f'}_G(v_x) = \tilde{f'}_G(\overline{v_x})$. Since the original f_G is negation-compliant, it can only happen if step 1 modifies f_G on v_x or $\overline{v_x}$. Hence, there is no variable x in $\{x | \tilde{f'}_G(v_x) \neq \tilde{f}_G(v_x),\ or\ \tilde{f'}_G(\overline{v_x}) \neq \tilde{f}_G(\overline{v_x})\}$ that is not already in D. Therefore, $dist(f, \tilde{f}) \leq dist(f_G, MON(G_\varphi))$. $\square$

We completed the proofs of theorems 3.1 and 3.2, which show that monotonicity of Boolean graph labelings is equivalent to 2CNF properties with respect to the number of queries required for testing.

3.2 Other testing problems equivalent to 2CNF testing

In this section, we present a reduction from monotonicity on general graphs to monotonicity on a special kind of bipartite graphs. The reduction works for all labeling alphabets. Then we give reductions between 2CNF properties, monotone 2CNF properties, and properties of being a vertex cover and being a clique in a fixed graph.

From monotonicity on general graphs to monotonicity on bipartite graphs

We now prove that testing monotonicity on arbitrary graphs is equivalent to testing monotonicity on balanced bipartite graphs with all edges directed to the same layer. The bipartite graphs we consider are formally defined below.

Definition 3.1 *For each directed graph $G = (\{v_1, \ldots, v_n\}, E)$, let B_G be the bipartite graph $(\{v_1, \ldots, v_n\},\ \{v'_1, \ldots, v'_n\}; E_B)$ where $E_B = \{(v_i, v'_j) |\ v_j \text{ is reachable from } v_i \text{ in } E\}$. For each labeling f of G, define the corresponding labeling f_B of B_G by $f_B(v_i) = f_B(v'_i) = f(v_i)$.*

Note that B_G is a transitively closed DAG with $2n$ vertices and the same number of edges as $\mathrm{TC}(G)$.

Claim 3.3 *Let f be a labeling on a graph G. Then $dist(f, MON(G)) = dist(f_B, MON(B_G))$.*

Proof. Assume that $dist(f_B, MON(B_G)) = k$ and let $\tilde{S}$ be a set of vertices in B_G of size k such that f_B is monotone on the subgraph of B_G induced by remaining vertices. Let S be the corresponding set of nodes in G. Note that S has size of at most k (it could be smaller as two vertices in $\tilde{S}$ might correspond to the same vertex in S). By the definition of f_B, labeling f is monotone on the subgraph of $\mathrm{TC}(G)$ induced by $V \backslash S$. By Lemma 2.1, $dist(f, MON(G)) \leq dist(f_B, MON(B_G))$.

To prove that $dist(f_B, MON(B_G)) \leq dist(f, MON(G))$, assume $dist(f, MON(G)) = k$. By Corollary 2.2, the graph of violated edges of $\mathrm{TC}(G)$ has a vertex cover S of size k. It is enough to show how to construct a vertex cover $\tilde{S}$ of the graph of violated edges of B_G of size k. We do so inductively by moving one vertex at a time from S to $\tilde{S}$, making sure every violated edge (v, u) in $\mathrm{TC}(G)$ is covered by S or has its counterpart (v, u') in B_G covered by $\tilde{S}$. Initially, $\tilde{S}$ is empty. When S becomes empty, $\tilde{S}$ is the required vertex cover.

It remains to show how to move a vertex from S to $\tilde{S}$. Let x be a minimal vertex in S according to the partial order imposed by the graph of violated edges of $\mathrm{TC}(G)$. If all violated edges of B_G incoming into x' are covered by $\tilde{S}$, move x from S to $\tilde{S}$ to cover outgoing violated edges from x in B_G. Otherwise, let (u, x') be an uncovered violated edge in B_G. Remove x from S and put x' into $\tilde{S}$. Now, all incoming edges into x' are covered by $\tilde{S}$. We claim that if there is a violated edge (x, v') not covered by $\tilde{S}$ then (x, v) is covered by S, i.e. $v \in S$. By transitivity of $\mathrm{TC}(G)$ and definition of f_B, edge (u, v') is also violated. Since it is not covered by $\tilde{S}$, its counterpart (u, v) in $\mathrm{TC}(G)$ must be covered by S. Since x was minimal in S, $u \notin S$. Hence, $v \in S$, as claimed. □

The next lemma shows that testing monotonicity on G reduces to testing monotonicity on B_G.

Theorem 3.4 *If $MON(B_G)$ is $(\frac{\varepsilon}{2}, \mu_+, \mu_-, q)$-testable for a graph G then $MON(G)$ is $(\varepsilon, \mu_+, \mu_-, q)$-testable. The reduction preserves 1-sided error: a 1-sided test for $MON(B_G)$ gives a 1-sided test for $MON(G)$.*

Proof. Let f be a labeling of G and let B_G be the associated graph with labeling f_B as defined above. By Claim 3.3, $dist(f, MON(G)) = dist(f_B, MON(B_G))$. If f is ε-far from monotone on G then f_B is $\varepsilon/2$-far from monotone on B_G because B_G has twice as many nodes. A test for G on input f can simulate a test for B_G on input f_B, asking at most the same number of queries. $\square$

Testing monotone 2CNFs

Recall that a *monotone* CNF is a CNF with only positive literals. We prove that testing 2CNFs is equivalent to testing monotone 2CNFs. One direction is obvious, for if all 2CNFs are testable, clearly all monotone 2CNFs are testable. To show that testability of monotone 2CNFs implies testability of all 2CNFs, recall that 2CNF testing is equivalent to testing Boolean monotonicity over general graphs (Theorems 3.1 and 3.2), which is equivalent to testing Boolean monotonicity on special kind of bipartite graphs (Definition 3.1 and Theorem 3.4). Therefore, to show the second direction, it is enough to prove that testability of monotone 2CNFs implies testability of Boolean monotonicity on this special kind of bipartite graphs.

Theorem 3.5 *Let $G = (X, Y; E)$ be a bipartite digraph with all edges directed from X to Y and $|X| = |Y| = n$. For each G there is a corresponding monotone* 2CNF *φ_G on n variables such that if $SAT(\varphi_G)$ is $(\varepsilon, \mu_+, \mu_-, q)$-testable then monotonicity of Boolean functions over G is also $(\varepsilon, \mu_+, \mu_-, q)$-testable.*

Proof. Associate a variable z_v with every node v in $X \cup Y$. Each node y in Y is represented by z_y, while each node x in X is represented by $\overline{z_x}$. Define a Boolean formula φ_G on the set of variables $Z = \{z_v |\ v \in X \cup Y\}$ as follows: form a clause $(z_x \vee z_y)$ for each edge $(x, y \in E)$. A Boolean labeling f of G defines an assignment $\tilde{f}$ for Z by $\tilde{f}(z_x) = 1 - f(x)$ if $x \in X$ and $\tilde{f}(z_x) = f(y)$ if $y \in Y$. Then an edge (x, y) is violated if and only if the corresponding clause $(z_x \vee z_y)$ is unsatisfied. Therefore, $dist(f, MON(G)) = dist(\tilde{f}, SAT(\varphi_G))$, and each test for φ_G can be used as a test for $MON(G)$. $\square$

Vertex cover and clique testing

Let $U = (V, E)$ be an undirected graph. For a $S \subseteq V$, let $f_S : V \to \{0, 1\}$ be a characteristic function of S, i.e. $f(v) = 1$ if and only if $v \in S$. A *vertex cover* of U is a subset of the vertices where every edge of U touches one of those vertices. A *clique* in U is a subset of the vertices that induces a complete graph in U. The property $VC(U)$ is the set of all characteristic functions f_S such that S is a vertex cover of U. Similarly, the property $CLIQUE(U)$ is the set of all characteristic functions f_S such that S is clique of U.

Theorem 3.6 *The following statements are equivalent:*

- $SAT(\varphi)$ *is* $(\varepsilon, \mu_+, \mu_-, q)$*-testable for every monotone* 2CNF φ *on* n *variables.*
- $VC(U)$ *is* $(\varepsilon, \mu_+, \mu_- q)$*-testable for every graph* U *on* n *nodes.*
- $CLIQUE(U)$ *is* $(\varepsilon, \mu_+, \mu_- q)$*-testable for every graph* U *on* n *nodes.*

The theorem follows from the following three lemmas.

Lemma 3.7 *For every undirected graph* U *on* n *nodes there is a corresponding monotone* 2CNF φ_U *on* n *variables s. t. if* $SAT(\varphi_U)$ *is* $(\varepsilon, \mu_+, \mu_-, q)$*-testable then so is* $VC(U)$*.*

Proof. Let $U = (V, E)$ be an undirected graph. Associate a Boolean variable x_v with each $v \in V$. Define the 2CNF formula φ_U on the set of variables $X = \{x_v | \ v \in V\}$ as follows: form the clause $(x_u \vee x_v)$ for each edge $(u, v) \in E$. A subset S of vertices in V defines an assignment $\tilde{f}$ to variables in X by $\tilde{f}(x_v) = f_S(v)$. Clearly $dist(f_S, VC(U)) = dist(\tilde{f}, SAT(\varphi_U))$, and every ε-test for $SAT(\varphi)$ can be used as a test for $VC(U_\varphi)$. $\square$

Lemma 3.8 *For every undirected graph* U *on* n *nodes there is a corresponding graph* U' *on* n *nodes s. t. if* $VC(U')$ *is* $(\varepsilon, \mu_+, \mu_-, q)$*-testable then so is* $CLIQUE(U)$*.*

Proof. Let $U = (V, E)$ be an undirected graph. Define $U' = (V, E')$ where E' is the set of vertex pairs that are not edges in E. For a subset S of V, let $S' = V \backslash S$. Clearly, $dist(f_S, CLIQUE(U)) = dist(f_{S'}, VC(U'))$, and every ε-test for $VC(U')$ can be used as an ε-test for $CLIQUE(U)$. $\square$

Lemma 3.9 *For every monotone* 2CNF φ *on* n *variables, there is a corresponding undirected graph* U_φ *on* n *nodes such that if* $CLIQUE(U_\varphi)$ *is* $(\varepsilon, \mu_+, \mu_-, q)$*-testable then so is* $SAT(\varphi)$*.*

Proof. Let φ be a monotone 2CNF. Associate a node v_x with each variable x of φ. Define the undirected graph U_φ on the set of vertices $V = \{v_x | \ x \in \varphi\}$ as follows: start with a complete graph on V and then for each clause $(x \vee y)$ in φ delete an edge (u_x, u_y) from U. An assignment f to the variables of φ defines a subset S of the vertices of V by $S = \{v_x | \ f(x) = 0\}$. Clearly, $dist(f, SAT(\varphi_U)) = dist(f_S, CLIQUE(U_\varphi))$, and every ε-test for $CLIQUE(U)$ can be used as a test for $SAT(\varphi_U)$. $\square$

Chapter 4

Bounds on Query Complexity for Monotonicity Testing

In this Chapter, we present upper and lower bounds on complexity of monotonicity tests on general graphs. All bounds apply to testing 2CNF properties and all other equivalent testing problems from Chapter 3. Instead of working with general graphs, we restrict our attention to bipartite graphs $G = (X, Y; E)$ with all edges directed from X to Y, described in Definition 3.1. By Theorem 3.4, testing monotonicity on such graphs is equivalent to testing monotonicity on general graphs.

The Chapter is organized as follows. In section 4.1, we present an algorithm for monotonicity on bipartite graphs that works for all labeling alphabets. In section 4.2, we define Ruzsá-Szemerédi graphs and show that non-adaptive monotonicity tests on these graphs require many queries. Section 4.3 contains constructions of Ruzsá-Szemerédi graphs, which imply lower bounds on monotonicity testing.

4.1 General upper bound

We present a simple 1-sided error ε-test for monotonicity (not necessarily Boolean) on bipartite graphs $G = (X, Y; E)$ with $|X| = |Y| = n$ and all edges are directed from X to Y. By Theorems 3.4–3.6, it implies 1-sided error ε-tests with the same query complexity for monotonicity over general graphs and four properties in Chapter 3.

Test T_1 for $G = (X, Y; E)$

1. Query $q = \left\lceil 2\sqrt{n/\varepsilon} \right\rceil$ vertices uniformly and independently from each of X and Y.

2. Reject if a violated pair of vertices is found; otherwise, accept.

Theorem 4.1 *If $G = (X, Y; E)$ as above, then algorithm T_1 is a 1-sided error $\left(\varepsilon, O(\sqrt{n/\varepsilon})\right)$-test for $MON(G)$.*

Proof. The test accepts all monotone functions. Suppose a function is ε-far from monotone. Then by Lemma 2.3, there are $\varepsilon n/2$ vertex-disjoint violated pairs. Call them *witness-pairs* and their vertices, *witnesses*. A randomly chosen X-vertex is a witness with probability ε.

Let F be the event that no violated pair is detected, F_X be the event that $\leq \varepsilon q/2$ X-witnesses are queried, and F_Y be the event that $\leq \varepsilon q/2$ Y-witnesses are queried.

$$\begin{aligned}
\Pr[F] &\leq \Pr[F_X] + \Pr[F_Y] + \Pr[F | \overline{F_X} \wedge \overline{F_Y}] \\
&\leq e^{-8} + e^{-8} + \left(1 - \frac{\varepsilon q/2}{\varepsilon n/2}\right)^{\varepsilon q/2} \leq 2e^{-8} + e^{-\frac{\varepsilon q^2}{2n}} < \frac{1}{3}.
\end{aligned}$$

Thus, the test fails with probability less than 1/3. $\square$

Corollary 4.2 *The following problems can be solved with $O(\sqrt{n/\varepsilon})$ queries by algorithms with 1-sided error:*

1. *monotonicity testing of functions with arbitrary ranges on general graphs;*

2. *testing if an assignment satisfies a* 2CNF *formula;*

3. *testing if a subset of vertices is a vertex cover;*

4. *testing if a subset of vertices is a clique.*

4.2 General lower bound

This section develops tools for a lower bound for testing monotonicity on general graphs. We restrict our attention to the Boolean case which implies matching lower bounds for all properties in Theorem 3.6. The lower bound is proved for bipartite graphs $G = (X, Y; E)$

where all edges are directed from X to Y (and hence longest directed path is of length 1). Definition 3.1 and Theorem 3.4 imply that this is in fact the general case. We first define what we call Ruzsá-Szemerédi graphs. We then show that monotonicity over such graphs (with suitable parameters) is hard to test non-adaptively.

Definition 4.1 *Let $U = (V, E)$ be an undirected graph and let $M \subseteq E$ be a matching in U, i.e. no two edges in M have a vertex in common. Let $V(M)$ be the set of the endpoints of edges in M. A matching M in U is called* induced *if the induced graph $U[V(M)]$ contains only the edges of M. Namely, $(u, v) \in E(U)$ if and only if $(u, v) \in M$ for all $u, v \in V(M)$. A (bipartite) graph $U = (X, Y; E)$ is called (s, t)-* Ruzsá-Szemerédi *if its edge set can be partitioned into at least s induced matchings $M_1, \ldots, M_s$, each of size at least t.*

The following theorem relates Ruzsá-Szemerédi graphs to lower bounds on monotonicity testing.

Theorem 4.3 *Let $U = (X, Y; E)$ be an $(m, \varepsilon n)$-Ruzsá-Szemerédi graph with $|X| = |Y| = n$. Direct all edges of U from X to Y to obtain a graph G. Then any non-adaptive $\frac{\varepsilon}{6}$-test for $MON(G)$ requires $\Omega(\sqrt{m})$ queries.*

Proof. We use Yao's principle, which says that to show a lower bound on the complexity of a randomized test, it is enough to present an input distribution on which any deterministic test with that complexity is likely to fail. Namely, we define distributions D_P, D_N on positive (monotone) and negative (ε-far from monotone) inputs, respectively. Our input distribution first chooses D_P or D_N with equal probability and then draws an input according to the chosen distribution. We show that every deterministic non-adaptive test with $q = o(\sqrt{m})$ queries has error probability larger than $1/3$ (with respect to the induced probability on inputs).

We now define the distributions D_P and D_N, as well as the auxiliary distribution $\tilde{D}_N$. For D_P and $\tilde{D}_N$, choose a random $i \in \{1, \ldots, m\}$ uniformly. For all nodes $x \in X$ and $y \in Y$ outside of matching M_i, set $f(x) = 1$ and $f(y) = 0$. For D_P, uniformly choose $f(x) = f(y) = 0$ or $f(x) = f(y) = 1$ independently for all edges $(x, y) \in M_i$. For $\tilde{D}_N$, uniformly choose $f(x) = 1 - f(y) = 0$ or $f(x) = 1 - f(y) = 1$ independently for all $(x, y) \in M_i$.

D_P is supported only on monotone labelings, but $\tilde{D}_N$ is not supported only on negative inputs. However, for n large enough, with probability more than $8/9$ at least $1/3$ of the edges of M_i are violated when the input is chosen according to $\tilde{D}_N$, making the input $\varepsilon/6$-far from monotone. Denote the latter event by A and define $D_N = \tilde{D}_N|_A$, namely, D_N is $\tilde{D}_N$ conditioned on the event A. Note that for $\tilde{D}_N$ an edge is violated only if it belongs to M_i, since the matchings are induced.

Given a deterministic non-adaptive test that makes a set V' of q queries, the probability that one or more of M_i's edges have both endpoints in V' is at most $q^2/(4m)$ for both $D_P, \tilde{D}_N$. This is because the matchings are disjoint, and the vertex set V' induces at most $q^2/4$ edges of G. For $q = o(\sqrt{m})$, with probability more than $1 - o(1)$ no edge of M_i has both endpoints in V'. Conditioned on any choice of i for which M_i has no such edge, the distribution of $f|_{V'}$ is identical for both $\tilde{D}_N$ and D_P: every vertex outside of M_i is fixed to 1 if it is in X and to 0 if it is in Y, and the value of every other vertex is uniform and independent over $\{0,1\}$. Let $C(\varphi)$ denote the set of inputs consistent with query answers $\varphi : V' \to \{0,1\}$. Then $\Pr_{D_P}[C(\varphi)|\text{no edge in } M_i] = \Pr_{\tilde{D}_N}[C(\varphi)|\text{no edge in } M_i]$. For every tuple of answers φ, the error probability under the above conditioning (with negative inputs chosen under $\tilde{D}_N$ rather than D_N) is $1/2$. As the probability of the condition is $\geq 1 - o(1)$, the overall error probability without the conditioning is $\geq 1/2 - o(1)$. Since negative inputs are chosen under D_N, not $\tilde{D}_N$, the success probability is $(1/2 + o(1)) \cdot (\Pr[A])^{-1} \leq (1/2 + o(1)) \cdot 9/8 \leq 9/16 + o(1)$. Thus, the error probability is $\geq 7/16 - o(1)$. $\square$

4.3 Construction of Ruzsá-Szemerédi graphs

This section presents a construction of Ruzsá-Szemerédi graphs that yields the $n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}$ non-adaptive lower bound for monotonicity testing. We give four different constructions, starting from a very intuitive one with relatively weak parameters and gradually improving it to the final technically involved construction with the desired parameters. All our constructions yield bipartite graphs. We commence with an extremely simple "toy" construction of Ruzsá-Szemerédi graphs whose edge sets can be partitioned into a *logarithmic* number of matchings of linear size. The main idea in the construction is to identify each layer of the bipartite graph with the set of binary strings of length $\log n$, and then match a top node with a bottom node when they differ in exactly one coordinate. This construction is extended to the second "toy" construction resulting in graphs whose edge sets can be partitioned into a *polylogarithmic* number of matchings of linear size. The idea behind the improvement is to match vertices based on values in a *subset* of coordinates.

After the toy constructions, we present a simplified version of the real construction. It yields Ruzsá-Szemerédi graphs with $n^{\Omega\left(\frac{1}{\log\log n}\right)}$ matchings of size almost $n/3$. In this construction, the strings corresponding to the nodes of the graph are over a larger alphabet, and they are matched according to the sum of values in a subset of coordinates. In the final construction, the sums of values are weighted. The last construction proves the following theorem.

Theorem 4.4 *For every n and $1/4 \geq \varepsilon > 0$, there exists a $\left(n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}, \varepsilon n\right)$-Ruzsá-Szemerédi graph $U = (X, Y; E)$ with $|X| = |Y| = n$.*

In conjunction with Theorem 4.3, this implies the desired lower bound for monotonicity testing.

Corollary 4.5 *For some $2n$-vertex graphs G, every non-adaptive (2-sided error) $(\varepsilon/6)$-test for $MON(G)$ requires $n^{\Omega\left(\frac{\log 1/\varepsilon}{\log\log n}\right)}$ queries.*

Closing the gap between the upper and the lower bound for monotonicity testing on general graphs remains an interesting open question. One possible method for improving the lower bound is finding better constructions of Ruzsá-Szemerédi graph graphs. In addition, as we discussed in the introductory Chapter, these graphs have other applications. In the final subsection of this Chapter, we discuss parameters of Ruzsá-Szemerédi graphs that are currently attainable.

4.3.1 Toy constructions

This subsection introduces two toy constructions of Ruzsá-Szemerédi graphs formalized in the lemmas below. The main purpose of these constructions is to ease the introduction of the real ones.

Lemma 4.6 *For every n, there exists a $(2\log n, n/2)$-Ruzsá-Szemerédi graph $U = (X, Y; E)$ with $|X| = |Y| = n$.*

Proof. Let $m = \log n$. Set both X and Y to $\{0,1\}^m$. Connect a node in X and a node in Y if they differ in exactly one coordinate. There are $2m$ matchings, each corresponding to a pair (i, x_i) where i is a coordinate in $[m]$ and $x_i \in \{0,1\}$. To obtain a matching corresponding to (i, x_i), take all nodes in X with x_i in coordinate i and match them with the nodes in Y with the opposite value in that coordinate. It is evident that the resulting graph has $2m = 2\log n$ edge-disjoint matchings of size $n/2$. To see that the matchings are induced, observe that if nodes $x \in X$ and $y \in Y$ both appear in the same matching, but are not matched in it, they differ on the coordinate corresponding to the matching and some other coordinate. Therefore, no edge in U connects x to y. $\square$

Our first improvement idea is to match nodes based on values at several coordinates. This modification to the construction above yields the following lemma.

Lemma 4.7 *For every n, there exists a $\left(\frac{1}{\varepsilon}\left(\frac{\log n}{\log 1/\varepsilon}\right)^{\log 1/\varepsilon}, \varepsilon n\right)$-Ruzsá-Szemerédi graph $U = (X, Y; E)$ with $|X| = |Y| = n$.*

Proof. As before, let $m = \log n$ and $X = Y = \{0,1\}^m$. Let p be a parameter which dictates how many coordinates are used for defining a matching. We set $p = \log 1/\varepsilon$. Connect a node in X and a node in Y if they differ in exactly p coordinates. Each matching corresponds to a pair (T, x_T) where T is a subset of coordinates $[m]$ of size $|T| = p$ and $x_T \in \{0,1\}^p$. To obtain a matching corresponding to (T, x_T), take all nodes in X with x_T in coordinates T and match them with the nodes in Y with the opposite values in those coordinates. It is evident that each such set of edges is a matching and that it has size $n/2^p = \varepsilon n$. By definition, matchings are edge-disjoint. The number of matchings is $2^p\binom{n}{p} \geq 2^p\left(\frac{m}{p}\right)^p = \frac{1}{\varepsilon}\left(\frac{\log n}{\log 1/\varepsilon}\right)^{\log 1/\varepsilon}$. To see that the matchings are induced, observe that if nodes $x \in X$ and $y \in Y$ both appear in the same matching, but are not matched in it, they differ on the coordinates T corresponding to the matching and some other coordinate. Therefore, no edge in U connects x to y. $\square$

This completes the two toy constructions. They produced Ruzsá-Szemerédi graphs with a polylogarithmic number of linear matchings.

4.3.2 A simple construction

This subsection improves on the toy constructions by identifying the nodes of the graph with strings over a larger alphabet and by using the sum of a subset of coordinates as a criterion for matching two nodes. It yields Ruzsá-Szemerédi graphs with much better parameters.

Lemma 4.8 *For every n, there exists a $\left(n^{\Omega\left(\frac{1}{\log\log n}\right)}, n/3 - o(n)\right)$-Ruzsá-Szemerédi graph $U = (X, Y; E)$ with $|X| = |Y| = n$.*

*Proof.*Let a, m be two integers where m is divisible by 3 and $m = o(a)$. The vertex set of U is $X = Y = [a]^m$, thus $n = a^m$. We define a family of (partial) matchings on the vertices of U and take the edge-set of the graph to be the union of the edge-sets of these matchings. The matchings are indexed by a family of $\frac{m}{3}$-subsets of $[m]$. Let $T \subseteq [m]$, $|T| = \frac{m}{3}$. Let $p = \frac{m}{3}$.

Definition of a matching M_T. Color the points in the two copies of $[a]^m$ by blue, red and white. The color of a point x is determined by $\sum_{i \in T} x_i$. First, partition the vertex set into *levels*, where the level L_s is the set $\left\{x : \sum_{i \in T} x_i = s\right\}$. Then combine levels into *strips*, where for an integer $k = 1...a$, the strip $S_k = L_{kp} \cup ... \cup L_{(k+1)p-1}$. Color the strips S_k with $k \equiv 0(\mod 3)$ blue, the strips with $k \equiv 1(\mod 3)$ red, and the remaining strips white. The matching M_T is defined by matching blue points in X to red points in Y as follows: If a blue point b in X has all its T-coordinates greater than 2, match it to a point $r = b - 2 \cdot 1_T$ in Y. The vector 1_T is the characteristic vector of T; it is 1 on T and 0 outside T. Note that r is necessarily red. M_T is clearly a matching. Our next step is to show that it is large.

Lemma 4.9 $|M_T| \geq n/3 - o(n)$.

Proof. Consider the "projected" matching M on the vertices of the bipartite graph $U^T = \left([a]^T, [a]^T\right)$, which is defined by T. Namely, partition the points of $[a]^T$ as described above, coloring them by blue, red and white, and match a blue point in one copy of $[a]^T$ to a red one in another, by subtracting $2 \cdot 1_T$. Since M_T is determined by the coordinates in T, it is enough to show that $|M| \geq P/3 - o(P)$, where $P = a^p$. Let $B, R, W \subseteq [a]^T$ be the sets of the blue, red and white points, respectively. Then $P = |B| + |R| + |W|$.

First, we claim that $|W| \leq |R| + \left|\{x : \exists i,\ x_i = 1\}\right|$. Indeed, consider a new matching between W and R defined by matching $w \in W$ to $w - 1_T$. Assume that $a \equiv 0(\mod 3)$. Then the only unmatched points in W are contained in the set $\{x : \exists i,\ x_i = 1\}$, proving this claim. Similarly $|W| \leq |B| + \left|\{x : \exists i,\ x_i = a\}\right|$.

Next, observe that the only blue and red points (in the corresponding copies of $[a]^T$) unmatched by M are these which have a coordinate whose value is in $\{1, 2, a-1, a\}$. It follows that

$$\begin{aligned}
|M| &> (|R|+|B|)/2 - \big|\{x : \exists i, x_i \in \{1,2,a-1,a\}\}\big| \\
&> P/3 - \Big(\big|\{x : \exists i, x_i \in \{1,2,a-1,a\}\}\big| + \big|\{x : \exists i, x_i = 1, a\}\big|\Big) \\
&\geq P/3 - \frac{6p}{a} \cdot P.
\end{aligned}$$

Since $p = o(a)$, the claim of the lemma follows. $\square$

Now, let T, T_1 be two $\frac{m}{3}$-sets in $[m]$, such that $|T \cap T_1| \leq m/7$. We claim that no edge of M_T is induced by M_{T_1}. Indeed, let b be matched to r by M_T, in particular $b - r = 2 \cdot 1_T$. If the edge (b, r) is induced by M_{T_1}, then b is colored blue and r is colored red in the coloring defined by T_1. By the definition of the coloring, since $\sum_{i=1}^m b_i > \sum_{i=1}^m r_i$, b is located in a blue level separated by a white level from the red level of r. This implies that

$$\Big|\sum_{i \in T_1} b_i - \sum_{i \in T_1} r_i\Big| \geq \frac{m}{3}.$$

On the other hand,

$$\Big|\sum_{i \in T_1} b_i - \sum_{i \in T_1} r_i\Big| = \Big|\sum_{i \in T_1} (b_i - r_i)\Big| = \Big|\sum_{i \in T_1} (2 \cdot 1_T)_i\Big| = 2 \cdot |T \cap T_1| \leq \frac{2m}{7} < \frac{m}{3},$$

reaching a contradiction.

We would like to have a large family $\mathcal{F}$ of $\frac{m}{3}$-subsets of $[m]$, such that the intersection between any two of them is of size at most $\frac{m}{7}$, or, equivalently, such that the Hamming distance between any two of them is at least $\frac{2m}{3} - \frac{2m}{7} = \frac{8m}{21}$. So we need a lower bound on the size of a constant weight binary error-correcting code $\mathcal{F}$ with the following parameters: block length m, weight $w = \frac{m}{3}$, distance $d = \frac{8m}{21}$. The Gilbert-Varshamov (or the "greedy") bound for constant weight codes [Lev71] gives, for $d \leq \frac{2w(m-w)}{m}$:

$$\frac{1}{m} \log |\mathcal{F}| \geq H\left(\frac{w}{m}\right) - \frac{w}{m} H\left(\frac{d}{2w}\right) - \left(1 - \frac{w}{m}\right) H\left(\frac{d}{2(m-w)}\right) - o(1).$$

Substituting the values of d and w, we get

$$\frac{1}{m} \log |\mathcal{F}| \geq H(1/3) - 1/3 \cdot H(4/7) - 2/3 \cdot H(2/7) - o(1) = 0.014 - o(1).$$

Choose $a = m^2$ and define the edge-set $E(U)$ of U by

$$E(U) = \bigcup_{T \in \mathcal{F}} M_T.$$

By the preceding discussion, U is a graph on $n = m^{2m}$ vertices, whose edge-set is a disjoint union of $2^{\Omega(m)} = n^{\Omega\left(\frac{1}{\log\log n}\right)}$ induced matchings of size $n/3 - o(n)$. $\square$

4.3.3 A more general construction

The construction of the previous section has a weakness - even if we allow the required size of the matching to be 'small', the number of matchings is bounded from above by 2^m - since the matchings are indexed by subsets of $[m]$. This section tries to deal with this problem, by presenting a slightly more general construction. There will be sufficient similarities for us to skip some details.

Let $1/4 \geq \varepsilon > 0$ be given. We construct a graph with n vertices and edge-set which can be partitioned into $n^{\Omega\left(\frac{\log\frac{1}{\varepsilon}}{\log\log n}\right)}$ induced matchings of size εn, thus proving Theorem 4.4.

Let $\theta = \frac{2\varepsilon}{1-2\varepsilon}$. (We will need θ later.) We will assume that $k = \frac{1}{\theta}$ is an integer.

Let a, m, p be three integers. We think about m as large, and require also $m = O(p)$, $kmp = o(a)$.

The vertex set of G is still $[a]^m$. In particular, $n = a^m$. The edges of G will be defined, as before, as a union of edge-sets of a family of matchings.

The matchings will be indexed by a family of vectors in $(k[p])^m = \{k, 2k ... pk\}^m$. (*This is the generalization.*) Let $v \in (k[p])^m$. Note that θv is an integer vector.

Definition of a matching M_v. We still color the points in $[a]^m$ by three colors blue, red and white. The color of a point x will be determined by $\langle x, v\rangle = \sum_{i=1}^m v_i x_i$. We partition the vertex set into *levels*, where the level L_s is the set $\{x : \langle x, v\rangle = s\}$. We combine levels into *strips* and color the consecutive strips red, white, blue, white, red, white, blue... as before. The only difference is the width of the strips. The non-white strips will be of width $\langle \theta v, v\rangle$, while the white strips will be of width $\langle v, v\rangle$.

The matching M_v matches blue points to red points. If a blue point b_i satisfies $b_i \geq (1+\theta)v_i$ for all $1 \leq i \leq m$, we match it to a point $r = b - (1+\theta)v$. Note that r is necessarily red. M_v is a matching. We need to show that it is large.

Lemma 4.10

$$|M_v| \geq \varepsilon n - o(n).$$

Proof. Once again, let $B, R, W \subseteq [a]^m$ be the sets of the blue, red and white points respectively. Clearly, $n = |B| + |R| + |W|$.

We claim that $|W| \leq k(|B|+|R|) + \big|\{x : \exists i,\ x_i \leq v_i\}\big|$. To see this, partition each white strip into k consecutive strips of width $\langle \theta v, v \rangle$. Let these substrips be numbered $1...k$. For $j = 1...k$, let W_j be the union over all the white strips of the substrips numbered j. Then $W = W_1 \cup W_2 ... \cup W_k$, a disjoint union. Now, we match each of the W_j to $B \cup R$, matching $w \in W_j$ to $w - j \cdot \theta v$. Assuming $a \equiv 2(4)$, the only unmatched points in W_j are contained in the set $\{x : \exists i,\ x_i \leq v_i\}$. Consider the union of these k matchings. It defines a function from $W \setminus \{x : \exists i,\ x_i \leq v_i\}$ to $B \cup R$, such that any point in $B \cup R$ is covered at most k times. The claim is proved.

Next, observe that the only blue and red points unmatched by M_v are these which have a coordinate i whose value lies in $\{1...(1+\theta)v_i\} \cup \{a - (1+\theta)v_i + 1...a\}$. It follows that

$$\begin{aligned}
|M_v| &\geq (|B|+|R|)/2 - \big|\{x : \exists i,\ x_i \in \{1...(1+\theta)v_i\} \cup \{a - (1+\theta)v_i + 1...a\}\}\big| \\
&\geq \frac{\theta}{2+2\theta} \cdot n \\
&\quad - \Big(\big|\{x : \exists i,\ x_i \in \{1...(1+\theta)v_i\} \cup \{a - (1+\theta)v_i + 1...a\}\}\big| + \big|\{x : \exists i,\ x_i \leq v_i\}\big|\Big) \\
&\geq \varepsilon \cdot n - \frac{(3+2\theta)kmp}{a} \cdot n.
\end{aligned}$$

The last inequality uses the identity $\frac{\theta}{2+2\theta} = \varepsilon$. Since $knp = o(m)$, the claim of the lemma follows. $\square$

Now, let v, w be two vectors in $(k[p])^m$, such that $\langle v, w \rangle < \frac{1}{1+\theta} \langle w, w \rangle$. We claim that no edge of M_v is induced by M_w. Indeed, let b be matched to r by M_v, in particular $b - r = (1+\theta)v$. If the edge (b, r) is induced by M_w, then b and r are *not* colored white in the coloring defined by w. This, by the definition of the coloring, implies that

$$\big|\langle b, w \rangle - \langle r, w \rangle\big| \geq \langle w, w \rangle .$$

On the other hand,

$$\big|\langle b, w \rangle - \langle r, w \rangle\big| = \big|\langle b - r, w \rangle\big| = \big|\langle (1+\theta)v, w \rangle\big| = (1+\theta) \cdot \langle v, w \rangle < \langle w, w \rangle ,$$

reaching a contradiction.

It remains to construct a large family $\mathcal{F}$ of vectors in $(k[p])^m$, such that for any two vectors $v, w \in \mathcal{F}$ holds $\left| \langle v, w \rangle \right| < \frac{1}{1+\theta} \langle w, w \rangle$.

A natural way to generalize the preceding section construction, would be to choose $\mathcal{F}$ as a spherical code of an appropriate distance. For this approach to work, we need a good lower bound on the size of a spherical code which is also a subset of a lattice $(k[p])^m$. To obtain such bound seems to be not entirely trivial. For this reason, we choose a slightly different way to proceed.

First, we need the Berry-Esseen theorem ([Dur96], p. 126):

Theorem 4.11 *Let $X_1, X_2 ...$ be i.i.d. random variables with $\mathbf{E}X_i = 0$, $\mathbf{E}X_i^2 = \sigma^2$, and $\mathbf{E}|X|_i^3 = \rho$. If $F_m(x)$ is the distribution of $(X_1 + ... + X_m)/\sigma\sqrt{m}$ and $\backslash(x)$ is the standard normal distribution then*

$$\left| F_m(x) - \backslash(x) \right| \leq \frac{3\rho}{\sigma^3 \sqrt{m}}.$$

For $v \in (k[p])^m$, let $Y(v) = \|v\|^2$ be the square of the Euclidean norm of v. If v is distributed uniformly over $(k[p])^m$, then Y is a sum of m i.i.d. random variables $Z_1 ... Z_m$, distributed uniformly over $k^2, (2k)^2, ...(pk)^2$. The expectation of Z_i is $(pk)^2/3 + O\left(pk^2\right)$, implying $\mathbf{E}Y = m\mathbf{E}Z_1 = mp^2k^2/3 + O\left(mpk^2\right)$. Let $X_i = Z_i - \mathbf{E}Z_i$, and apply the Berry-Esseen theorem to the distribution of $Y - \mathbf{E}Y = \sum_{i=1}^m X_i$. It is not hard to see that $\mathbf{E}X_i^2 = \sigma^2 = \Theta\left((pk)^4\right)$, while $\mathbf{E}|X|_i^3 = \rho = O\left((pk)^6\right)$. Therefore $\rho/\sigma^3 = O(1)$. It follows that for any $x > 0$,

$$\left| \mathbf{P}\left\{ \mathbf{E}Y - x\sigma(Y) \leq Y \leq \mathbf{E}Y + x\sigma(Y) \right\} - \frac{1}{\sqrt{2\pi}} \int_{-x}^{x} e^{-t^2/2} dt \right| \leq O\left(\frac{1}{\sqrt{m}}\right).$$

Here $\sigma(Y) = \sqrt{m}\sigma = \Theta\left(\sqrt{m}p^2k^2\right)$. Note also that, for a sufficiently large m, the second order term $O\left(mpk^2\right)$ in $\mathbf{E}Y = mp^2k^2/3 + O\left(mpk^2\right)$ is negligible compared to $\sigma(Y)$. Applying (4.1) with an appropriate x close to 1, we see that, for a sufficiently large m, Y lies in the interval $mp^2k^2/3 \pm \sqrt{m}p^2k^2$ with bounded away from 0 probability.

In other words, a constant fraction of points of $(k[p])^m$ lie in the m-dimensional spherical annulus $A = \left\{ x : \left(1 - \frac{3}{\sqrt{m}}\right) R^2 \leq \|x\|^2 \leq \left(1 + \frac{3}{\sqrt{m}}\right) R^2 \right\}$, where we have set $R = pk\sqrt{m/3}$.

Lemma 4.12 *Let $R > 0$, $0 \leq \alpha \leq \beta$, and let v, w be two points in a spherical annulus*

$\{x : \alpha R \leq \|x\| \leq \beta R\}$ *with* $\|v - w\| > \sqrt{\beta^2 - \frac{1-\theta}{1+\theta}\alpha^2} \cdot R$. *Then*

$$\langle v, w \rangle < \frac{1}{1+\theta} \langle w, w \rangle .$$

Proof. Let $\delta = \sqrt{\beta^2 - \frac{1-\theta}{1+\theta}\alpha^2}$. Expand $\langle v - w, v - w \rangle$ to obtain

$$\delta^2 R^2 < \langle v - w, v - w \rangle = \langle v, v \rangle + \langle w, w \rangle - 2 \langle v, w \rangle .$$

Therefore $\langle v, w \rangle < \frac{\langle v,v \rangle + \langle w,w \rangle - \delta^2 R^2}{2}$. We want the RHS of this inequality not to exceed $\frac{1}{1+\theta} \langle w, w \rangle$. This is equivalent to

$$\langle v, v \rangle - \delta^2 R^2 \leq \frac{1-\theta}{1+\theta} \langle w, w \rangle .$$

In the last inequality the worst possible case is when v is of the maximal possible norm in the annulus and w is of the minimal possible norm. Substituting the corresponding norms and the value of δ, we obtain an equality. □

Lemma 4.13 *An m-dimensional Euclidean ball of radius r contains at most* $\left(O\left(\frac{r}{k\sqrt{m}} + 1\right)\right)^m$ *points of the lattice* $k\mathbf{Z^m}$.

Proof. Let B be an m-dimensional Euclidean ball of radius r containing I points of the lattice $k\mathbf{Z^m}$. Take an m-dimensional cube with sides of length k around each lattice point in B. These cubes have disjoint interiors and they are contained in a ball of radius $r + k\sqrt{m}$. Recall that the volume of an m-dimensional Euclidean ball of radius t is $\frac{\pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2}+1)} \cdot t^m = O\left(\left(\sqrt{\frac{2\pi e}{m}} \cdot t\right)^m\right)$. Since the volume of each cube is k^m,

$$I \leq O\left(\frac{\left(\sqrt{\frac{2\pi e}{m}} \cdot (r + k\sqrt{m})\right)^m}{k^m}\right) \leq \left(O\left(\frac{r}{k\sqrt{m}} + 1\right)\right)^m .$$

□

We now choose $\mathcal{F}$ to be a subset of $(k[p])^m \cap A$, such that for any two distinct points $v, w \in A$ holds $\|v - w\| > \delta R = \sqrt{\frac{2\theta}{1+\theta} + \frac{6}{\sqrt{m}}} \cdot R$. By lemma 4.12, with $\alpha = \sqrt{1 - \frac{3}{\sqrt{m}}}$ and $\beta = \sqrt{1 + \frac{3}{\sqrt{m}}}$, any two distinct points $v, w \in A$ satisfy $\langle v, w \rangle < \frac{1}{1+\theta} \langle w, w \rangle$. By lemma 4.13, there exists such a family with at least

$$\left(\Omega\left(\frac{p}{\left(\frac{\delta R}{k\sqrt{m}}+1\right)}\right)\right)^m \geq \left(\Omega\left(\frac{1}{\delta}\right)\right)^m$$

points. To see the inequality, recall that $R = pk\sqrt{m/3}$, and therefore $\frac{\delta R}{k\sqrt{m}} = \Omega(\delta p) \gg 1$ since, by our (soon to be specified) choice of parameters, $p = \Omega(n)$. It remains to write $\frac{1}{\delta}$ in terms of ε. By the definition of θ, $\delta = \sqrt{\frac{2\theta}{1+\theta} + \frac{6}{\sqrt{m}}} = \sqrt{4\varepsilon + \frac{6}{\sqrt{m}}} = O\left(\sqrt{\varepsilon}\right)$, for $\varepsilon \geq \Omega\left(1/\sqrt{m}\right)$. We will deal with subconstant ε in the next section. For now we have proved the following claim: For any constant ε and for a sufficiently large m, there exists a family $\mathcal{F}$ of size $\left(\Omega\left(\frac{1}{\varepsilon}\right)\right)^{\frac{m}{2}}$.

We choose $p = m$ and $m = kn^3 = \frac{1-2\varepsilon}{2\varepsilon}n^3$, and define the edge-set $E(G)$ of G as follows:

$$E(G) = \bigcup_{v \in \mathcal{F}} M_v.$$

By the preceding discussion, G is a graph on $n = \left(\frac{1-2\varepsilon}{2\varepsilon}\right)^m \cdot m^{3m}$ vertices, whose edge-set is a disjoint union of $\left(\Omega\left(\frac{1}{\varepsilon}\right)\right)^{\frac{m}{2}} = n^{\Omega\left(\frac{\log\frac{1}{\varepsilon}}{\log\log n}\right)}$ induced matchings of size $\varepsilon n - o(n)$.

4.3.4 Attainable parameters of (s,t)-Ruzsá-Szemerédi graphs

Consider the following question: For which values of s and t is there an (s,t)-Ruzsá-Szemerédi graph? We are interested in the asymptotic version of this question as $n \to \infty$. Call a sequence of pairs $(s(n), t(n))$-*realizable* if there is an infinite sequence of n, and graphs U_n with n vertices, that are $(s(n), t(n))$-Ruzsá-Szemerédi. Define $\mathcal{P}$ to be the set $\{(s(n), t(n))\}$ of realizable sequences. Note that $\mathcal{P}$ is monotone in the natural order on pairs, namely if it contains (s,t), and $s' \leq s$, $t' \leq t$, then it contains (s', t'). Therefore it is defined by its set of maximal points.

Two trivial maximal points in $\mathcal{P}$ are $\left(\binom{n}{2}, 1\right)$, coming from a complete graph on n vertices, and $(1, n/2)$, coming from a perfect matching on n vertices. A much more interesting point in $\mathcal{P}$ is given by a construction of Ruzsá and Szemerédi [RS78], following Behrend [Beh46]. Their result, with some abuse of notation, can be stated as follows:

Theorem 4.14

$$\left(n/3, n/2^{O\left(\sqrt{\log n}\right)}\right) \in \mathcal{P}.$$

Our goal here is to check what realizable pairs could be obtained by the construction of the previous section. We have already seen in section 4.3.2 that for $\varepsilon = \Omega(1)$ there is an absolute constant c, such that $(n^{c/\log\log n}, \varepsilon n) \in \mathcal{P}$. This trivially implies that there is a constant c such that for any positive ε, $\left(1/\varepsilon \cdot n^{c/\log\log n}, \varepsilon n\right) \in \mathcal{P}$. A more technically involved construction of section 4.3.3 gives the following theorem.

Theorem 4.15 *There is a constant c such that for any constant positive $\varepsilon \leq 1/4$,*

$$\left(n^{(c\cdot\log 1/\varepsilon)/\log\log n}, \varepsilon n\right) \in \mathcal{P}.$$

The case of a constant ε is the interesting case from the "testing" point of view.

Consider now the construction of the previous section, and let ε go to 0 as n grows. As a matter of fact, since we first choose ε and then define the other parameters m, n, p, the right order of things should be as follows: We choose a sequence $\varepsilon_i \to 0$ and then define m_i, n_i, p_i as functions of ε_i. Having this in mind, since (for a fixed i) everything depends on one parameter, it will be convenient to choose this basic parameter to be m (and drop i).

Let us first look at $\varepsilon = \Omega\left(1/\sqrt{m}\right)$. This is an easy case, since it is not hard to see that the only change that needs to be introduced to the analysis of the preceding section is ensuring that the error term in lemma 4.10 is in fact $o(\varepsilon n)$. For this it is sufficient to require $\frac{knp}{m} = o(\varepsilon)$, or equivalently $\frac{np}{\varepsilon^2} = o(m)$. Choosing $p = m$ and $m = \frac{n^3}{\varepsilon^2}$ we obtain a graph on $n = \left(\frac{1}{\varepsilon}\right)^{2n} \cdot n^{3n}$ vertices, whose edge set is a disjoint union of $\left(\Omega\left(\frac{1}{\varepsilon}\right)\right)^{\frac{m}{2}}$ induced matchings of size $\varepsilon n - o(n)$. Expressing everything through n, we see that for $\varepsilon = \Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$, the point $\left(n^{\Omega\left(\frac{\log\frac{1}{\varepsilon}}{\log\log n}\right)}, \varepsilon n\right) \in \mathcal{P}$. In particular, for some absolute constant $c < 1$, such that $\left(n^c, n/O\left(\sqrt{\log n/\log\log n}\right)\right) \in \mathcal{P}$.

Chapter 5

Monotonicity Testing on Special Graphs

This Chapter contains results on monotonicity testing for specific graphs which can be tested with much better query complexity than given by the general lower bound. The first section deals with lower bounds for the well-studied Boolean hypercube. The second section identifies new families of graphs which are testable with constant or logarithmic number of queries.

5.1 Lower bounds for the Boolean hypercube

The Boolean hypercube is a directed graph with vertex set $\{0,1\}^m$ and the edge set $\{(x,y) \mid x_i \leq y_i \ \ \forall i \in [m]\}$. Throughout this Chapter, $\|x\|$ denotes the Hamming weight of vector x. This section contains lower bounds on the query complexity of non-adaptive monotonicity tests for Boolean functions over the Boolean hypercube. In the first subsection, we present a lower bound of $\Omega(\sqrt{m})$ for the 1-sided error case. In the second subsection, we give a lower bound of $\Omega(\log m)$ for the 2-sided error case. The corresponding adaptive lower bounds can be obtained by taking a logarithm of the non-adaptive bounds. The justification for this is very simple: every adaptive test can be simulated by a non-adaptive test that makes all queries that the adaptive test could have made on all its branches. There is still a large gap between our lower bounds and the best-known upper bound of $O(m/\varepsilon)$ by Dodis et al. [DGL$^+$99]. This upper bound is given by a non-adaptive algorithms with 1-sided error.

5.1.1 1-sided error non-adaptive lower bound

We give a $\Omega(\sqrt{m})$ lower bound on the query complexity of non-adaptive 1-sided error monotonicity tests for Boolean functions over the Boolean hypercube. This implies a logarithmic lower bound for adaptive 1-sided error testing of this property.

Theorem 5.1 $\exists \varepsilon > 0$ *such that every non-adaptive 1-sided error ε-test for monotonicity of Boolean functions on the m-dimensional Boolean hypercube requires $\Omega(\sqrt{m})$ queries.*

Proof. Note that a 1-sided error test must accept if no violation is uncovered; otherwise, the test fails on monotone functions consistent with the query results. For $i = 1, ..., m$ define a function $f_i : \{0,1\}^m \to \{0,1\}$ by

$$f_i(x_1, \ldots, x_m) = \begin{cases} 1 & \text{if } \|x\| > m/2 + \sqrt{m} \\ 0 & \text{if } \|x\| < m/2 - \sqrt{m} \\ 1 - x_i & \text{otherwise} \end{cases}$$

It is easy to see that for all $1 \le i \le m$, f_i is ε-far from monotone, for some constant $\varepsilon > 0$ independent of m. Lemma 5.2 immediately implies our theorem. $\square$

Lemma 5.2 *For every non-adaptive q-query monotonicity test, there exists an index $i \in [m]$, such that the test succeeds (finds a violation) on f_i with probability at most $O(q/\sqrt{m})$.*

Proof. It suffices to prove the claim for tests that only query vertices with Hamming weight in the range $m/2 \pm \sqrt{m}$, as vertices outside of this range do not participate in any violations.

We show that every set of q queries reveals a violation for at most $O(q\sqrt{m})$ of the functions f_i. It follows that for every test that makes q queries, $\sum_{i=1}^{m} \Pr[\text{a violation for } f_i \text{ is found}] = O(q\sqrt{m})$, and so there exists an f_i for which the test finds a violation with probability at most $O(q/\sqrt{m})$, as claimed.

Let Q be the set of queried vertices of $\{0,1\}^m$ of size q. The queries detect a pair of vertices violated by f_i only if Q contains comparable vertices u and v that differ in coordinate i. Construct an undirected graph with vertex set Q, by drawing an edge between x and y if they are comparable. Consider a spanning forest of this graph. If such vertices u and v exist, they must lie in the same tree. Furthermore, there must exist adjacent vertices on the path between u and v that differ in coordinate i. Therefore, the number of functions f_i for which the queries reveal a violation is at most the maximum number of edges in the forest (which is at most $q-1$) multiplied by the maximum possible distance between adjacent vertices ($2\sqrt{m}$). The total is at most $O(q\sqrt{m})$. $\square$

5.1.2 2-sided error non-adaptive lower bound

We give a logarithmic lower bound on the query complexity of non-adaptive 2-sided error monotonicity tests for Boolean functions over the Boolean hypercube. This implies a non-constant (though doubly logarithmic) lower bound for adaptive 2-sided error testing of this property.

Theorem 5.3 *$\exists\varepsilon > 0$ such that every non-adaptive ε-test for monotonicity of Boolean functions on the m-dimensional Boolean hypercube requires $\Omega(\log m)$ queries.*

Proof. The lower bound uses Yao's method. We define two kinds of input functions – trimmed oligarchy and trimmed anti-oligarchy functions (see Definition 5.1). Lemma 5.4 proves that trimmed oligarchy functions are monotone and trimmed anti-oligarchy functions are ε-far from monotone for a constant ε. Then definition 5.2 gives distributions D_P and D_N over trimmed oligarchy and anti-oligarchy functions, correspondingly. Lemma 5.5 shows that for every set of $q \leq \frac{1}{20}\log m$ vertices of the hypercube, the distributions induced on $\{0,1\}^q$ by restricting the functions to the q vertices are $< \frac{1}{3}$ close. By Yao's minimax principle, this proves the theorem. $\square$

For $x \in \{0,1\}^m$, we view x both as a binary vector of length m and a subset $\{i : x_i = 1\}$ of $[m]$.

Definition 5.1 *Let $\alpha = \frac{1}{100}$. Given $B \subseteq [m]$, let $maj(x \cap B)$ be 1 when $|x \cap B| > \frac{1}{2}|B|$ and 0 otherwise.*

The trimmed oligarchy function according to B *is*

$$f_B(x) = \begin{cases} 1 & \text{if } \|x\| > m/2 + \alpha\sqrt{m} \\ 0 & \text{if } \|x\| < m/2 - \alpha\sqrt{m} \\ maj(x \cap B) & \text{otherwise} \end{cases}$$

The trimmed anti-oligarchy function according to B *is*

$$f_B^A(x) = \begin{cases} 1 & \textit{if } \|x\| > m/2 + \alpha\sqrt{m} \\ 0 & \textit{if } \|x\| < m/2 - \alpha\sqrt{m} \\ 1 - maj(x \cap B) & \textit{otherwise} \end{cases}$$

Lemma 5.4 *There exists $\varepsilon > 0$, such that for any nonempty set B, f_B is monotone and f_B^A is ε-far from monotone.*

Proof. It is easy to see that trimmed oligarchy functions are monotone. For trimmed anti-oligarchy functions, we will find $\varepsilon 2^m$ vertex-disjoint violated pairs.

Let $b = |B|$. For every integer w such that $m/2 - \alpha\sqrt{m} \le w < m/2$, and every integer v such that $0 \le v < b/2$, let $U_{w,v}$ denote the set $\{x \in \{0,1\}^m : \|x\| = w \text{ and } |x \cap B| = v\}$, and $V_{w,v}$ denote the set $\{x \in \{0,1\}^m : \|x\| = m - w \text{ and } |x \cap B| = b - v\}$. By definition, $f(x) = 1$ for every $x \in U_{w,v}$ and $f(x) = 0$ for every $x \in V_{w,v}$. Sets $U_{w,v}$ and $V_{w,v}$ have the same size, since x is in $U_{w,v}$ iff the complement of x is in $V_{w,v}$. We want to find a bijection $\sigma : U_{w,v} \to V_{w,v}$ such that $x \preceq \sigma(x)$ for every $x \in U_{w,v}$.

Consider the bipartite graph over $U_{w,v} \cup V_{w,v}$ with the poset relations as edges. By symmetry, this graph has a constant degree, so a matching exists (by Hall's Theorem) if this degree is nonzero. This happens if w, v satisfy $b/2 - v \le m/2 - w$ in addition to the conditions above. The union over all such w, v of the sets $U_{w,v} \cup V_{w,v}$ covers a fixed fraction of the hypercube, so we are done. □

Definition 5.2 *To define D_P and D_N pick a random set $B \subset [m]$ by independently choosing each coordinate to lie in B with probability $\frac{1}{10\sqrt{m}}$. For D_P, take the corresponding f_B and for D_N, take the corresponding f_B^A.*

Lemma 5.5 *D_N and D_P restricted to any set of $q = \frac{1}{20}\log m$ queries are ε-close, for any $\varepsilon > 0$.*

Proof. Let $q = \frac{1}{20}\log m$ and let $x_1, \ldots, x_q$ be the queries. Since we are considering only non-adaptive tests, queries $x_1, \ldots, x_q$ form a fixed subset of $\{0,1\}^m$. Let d_P and d_N be the induced distributions on $\{0,1\}^q$ obtained by restricting D_P and D_N to the queries. Our goal is to show $\|d_P - d_N\|_1 = o(1)$.

We can assume, without loss of generality, that the queries satisfy $m/2-\alpha\sqrt{m} \leq \|x_i\| \leq m/2+\alpha\sqrt{m}$. This is because functions f_B and f_B^A are constant and identical outside this range. Inside the range, for every B, f_B and f_B^A *complement* each other. Therefore, the induced distributions d_P and d_N on $\{0,1\}^q$ are mirror images of each other: $d_P(a) = d_N(\bar{a})$ for any $a \in \{0,1\}^q$, where $\bar{a}$ is the complement of a. For a distribution d on $\{0,1\}^q$, let $\bar{d}$ be its mirror image. Call d *symmetric* if $d = \bar{d}$.

Our claim amounts to showing that d_P is almost symmetric. Namely, we construct a symmetric distribution s, such that $\|d_P - s\|_1 = o(1)$. This implies our claim since

$$\|d_P - d_N\|_1 = \|d_P - \bar{d_P}\|_1 \leq \|d_P - s\|_1 + \|\bar{d_P} - \bar{s}\|_1 = 2\|d_P - s\|_1.$$

We exhibit two intermediate distributions, d_2 and d_3, such that every distribution in the sequence $d_1 = d_P$, d_2, d_3, s is close to its predecessor. The triangle inequality then implies that the distance between d_P and s is at most the sum of the distances between the consecutive elements of the sequence.

We define distribution d_2 by replacing each query vertex with a nearby vertex from the middle layer of the hypercube. For $1 \leq i \leq q$, fix $y_i \in \{0,1\}^m$ with $\|y_i\| = m/2$ and $\|x_i - y_i\| \leq \alpha\sqrt{m}$. Let d_2 be the distribution on $\{0,1\}^q$ induced by restricting the functions in D_P to $y_1, ..., y_q$. Then d_2 is close to d_1 because with high probability over the choice of a function f from D_P, changing the queries by at most $O(\sqrt{m})$ bits, does not change the value of f on the queries.

To see that d_2 is close to d_1, let us look again at the way d_1 and d_2 are obtained from D_P. Let $p = \frac{1}{10}m^{-\frac{1}{2}}$, and let Ω be the probability space of all subsets B of $[m]$ endowed with product measure $\mu(B) = p^{|B|}(1-p)^{m-|B|}$. Let $X_1, X_2 : \Omega \to \{0,1\}^q$ be two random variables, defined as follows: $X_1(B) = (f(x_1)...f(x_q))$, and $X_2(B) = (f(y_1)...f(y_q))$, where f is the oligarchy function corresponding to B. Then the distribution of X_1 on $\{0,1\}^q$ is d_1 and the distribution of X_2 is d_2.

Now we need a following standard and easy fact: for two random variables defined on the same probability space, the l_1-distance between their distributions is at most $2Pr\{X \neq Y\}$.

Therefore, $\|d_1 - d_2\|$

$$
\begin{aligned}
&\leq \; 2Pr_{\mu}\{X_1 \neq X_2\} = 2Pr_{D_P}\left\{\exists i \; : \; f(x_i) \neq f(y_i)\right\} \\
&\leq \; 2Pr_{\mu}\left\{\exists i \; : \; \left||y_i \cap B| - \frac{1}{2}|B|\right| \leq m^{\frac{1}{3}} \bigvee \left||y_i \cap B| - |x_i \cap B|\right| \geq m^{\frac{1}{3}}\right\} \\
&\leq \; 2\left(\sum_{i=1}^{q} Pr_{\mu}\left\{\left||y_i \cap B| - \frac{1}{2}|B|\right| \leq m^{\frac{1}{3}}\right\} + \sum_{i=1}^{q} Pr_{\mu}\left\{\left||y_i \cap B| - |x_i \cap B|\right| \geq m^{\frac{1}{3}}\right\}\right) \\
&\leq \; O\left(\frac{q}{m^{\frac{1}{6}}}\right) = O\left(\frac{\log m}{m^{\frac{1}{6}}}\right).
\end{aligned}
$$

The last inequality follows from two simple applications of Chebyshev's inequality to the random variables $|y_i \cap B| - \frac{1}{2}|B|$ and $|y_i \cap B| - |x_i \cap B|$.

The sets $y_1 ... y_q$ induce a standard partition of $[m]$ into 2^q disjoint subsets, indexed by $\{0,1\}^q$. For $I \in \{0,1\}^q$, the I'th element of the partition is $A_I = \bigcap_{i:I_i=1} y_i \cap \bigcap_{i:I_i=0} y_i^c$. Here y_i^c is the set complement of y_i. We define 2^q random variables on Ω by setting $R_I(B) = |B \cap A_I|$. If A_I is empty, R_I is identically 0.

Note that $\{R_I\}$ are independent binomially distributed variables, and that they determine X_2. In fact, the i'th coordinate of $X_2(B)$ is 1 if and only if $|y_i \cap B| > |y_i^c \cap B|$, which is equivalent to $\sum_{I:i \in I} R_I > \sum_{I:i \notin I} R_I$.

Since $\mathbf{E}\left(\sum_{I:i\in I} R_I\right) = mp/2 = \mathbf{E}\left(\sum_{I:i\notin I} R_I\right)$, we can replace each R_I by a random variable $Z_I = R_I - \mathbf{E}R_I$ with zero expectation.

Next, we would like to replace each Z_I by a symmetric random variable. We know, say by the Berry-Esseen theorem, that if R has a binomial distribution with parameters k and p, such that $kp \gg 1$, then the distribution of R is, in some sense, close to the normal distribution which is, of course, symmetric. We will give a precise meaning to this intuition shortly. However, first we have to get rid of Z_I corresponding to small A_I. Let $X_3 : \Omega \rightarrow \{0,1\}^q$ be defined as follows: the i'th coordinate of $X_3(B)$ is 1 if and only if $\sum_{I:i\in I,|A_I|\leq m^{3/5}} Z_I > \sum_{I:i\notin I,|A_I|\leq m^{3/5}} Z_I$. Let d_3 be the distribution of X_3. We claim that

$$\|d_2 - d_3\| \leq o(1).$$

Indeed, Let $\tilde{A} = \bigcup_{I:|A_I| \leq m^{3/5}} A_I$. Clearly, $|\tilde{A}| \leq 2^q \cdot m^{3/5} < m^{7/10}$. We estimate from above the probability that X_2 differs from X_3. Similarly to what we had before,

$$Pr_\mu\{X_2 \neq X_3\} \leq Pr_\mu\left\{\exists i \ : \ \left||y_i \cap B| - \frac{1}{2}|B|\right| \leq m^{\frac{9}{40}} \bigvee |\tilde{A} \cap B| \geq m^{\frac{9}{40}}\right\} \leq$$

$$\sum_{i=1}^{q} Pr_\mu\left\{\left||y_i \cap B| - \frac{1}{2}|B|\right| \leq m^{\frac{9}{40}}\right\} + Pr_\mu\left\{|\tilde{A} \cap B| \geq m^{\frac{9}{40}}\right\} \leq O\left(\frac{\log m}{m^{\frac{1}{40}}}\right).$$

Let us prove the last inequality. $|\tilde{A} \cap B|$ is a binomial random variable with expectation $|\tilde{A}| \cdot p \leq m^{\frac{1}{5}}$ and variance of a similar magnitude. Therefore, by the Chebyshev inequality, $Pr_\mu\left\{|\tilde{A} \cap B| \geq m^{\frac{9}{40}}\right\} \leq O\left(\frac{1}{m^{\frac{5}{20}}}\right)$. On the other hand, $|y_i \cap B| - \frac{1}{2}|B| = \frac{1}{2}\left(|y_i \cap B| - |y_i^c \cap B|\right)$, is a difference of two independent identically distributed binomial random variables with parameters $m/2$ and p. By the extended version for different random variables [Dur96] of the Berry-Esseen theorem, the distribution of $|y_i \cap B| - \frac{1}{2}|B|$ is close, up to $O\left(\frac{1}{m^{\frac{1}{4}}}\right)$ in the supremum norm, to the normal distribution with mean 0 and variance $\Theta\left(m^{\frac{1}{5}}\right)$. Therefore $Pr_\mu\left\{\left||y_i \cap B| - \frac{1}{2}|B|\right| \leq m^{\frac{9}{40}}\right\} \leq O\left(\frac{1}{m^{\frac{1}{40}}}\right)$, and we are done.

Now we are left only with Z_I for which $|A_I| \geq m^{\frac{3}{5}}$. We replace them by symmetric random variables S_I without losing much, due to the following claim.

Claim 5.6 *Let R be a binomial random variable with parameters k and p, $p \leq \frac{1}{2}$, such that $\mathbf{E}R = kp$ is integer. Let $Z = R - \mathbf{E}R = R - kp$. Let S be a symmetric random variable with integer values, such that $Pr(S = t) = \frac{1}{2} \cdot (Pr(Z = t) + Pr(Z = -t))$. Let d_Z, d_S be the distributions of Z and S correspondingly. Then*

$$\|d_Z - d_S\|_1 \leq O\left(\frac{\log^{3/2}(kp)}{\sqrt{kp}}\right).$$

Proof. First, we may clearly assume that kp is sufficiently large, in particular $kp \gg \log^3(kp)$. Let $\varepsilon = \frac{\log^{3/2}(kp)}{\sqrt{kp}}$. By the standard large deviation inequalities [AS00], $Pr\{|Z| \geq t\} \leq exp\left\{-\frac{2t^2}{kp} + \frac{t^3}{2k^2p^2}\right\}$. Therefore, for some constant c,

$$\|d_Z - d_S\|_1 \leq \frac{1}{2} \cdot \left(\sum_{t=1}^{c\sqrt{kp\log(kp)}} |Pr(Z = t) - Pr(Z = -t)|\right) + O(\varepsilon).$$

It follows that in order to prove the claim, it suffices to show that for any $1 \le t \le O\left(\sqrt{kp\log(kp)}\right)$ holds

$$1 - O(\varepsilon) \le \theta := \frac{Pr(Z = t)}{Pr(Z = -t)} \le 1 + O(\varepsilon).$$

Let $q = 1 - p$. [1] By direct computation,

$$\theta = \frac{p^{2t}}{q^{2t}} \cdot \frac{(kq + t)!(kp - t)!}{(kq - t)!(kp + t)!} = \frac{kp - t}{kq - t} \cdot \frac{p^{2t+1}}{q^{2t+1}} \cdot \prod_{i=1}^{t} \frac{k^2q^2 - i^2}{k^2p^2 - i^2} = \frac{kpq - tq}{kpq - tp} \cdot \frac{p^{2t+2}}{q^{2t+2}} \cdot \prod_{i=1}^{t} \frac{k^2q^2 - i^2}{k^2p^2 - i^2}.$$

Note that, for every i, $\frac{k^2q^2-i^2}{k^2p^2-i^2} \ge \frac{p^2}{q^2}$, and therefore $\theta \ge \frac{kpq-tq}{kpq-tp} \ge 1 - \frac{t}{kp} \ge 1 - \varepsilon$.

On the other hand, using the fact that for $0 \le \delta \le \frac{1}{2}$ holds $\frac{1}{1-\delta} \le 1 + 2\delta$,

$$\theta \le \left(\frac{k^2p^2q^2 - t^2p^2}{k^2p^2q^2 - t^2q^2}\right)^t \le \left(1 + \frac{2t^2}{k^2p^2}\right)^t \le exp\left\{\frac{2t^3}{k^2p^2}\right\} \le 1 + O(\varepsilon).$$

□

Using the claim, for each remaining random variable Z_I, we construct a symmetric random variable S_I.[2] By the claim, for each I with $|A_I| \ge m^{\frac{3}{5}}$ the distributions of Z_I and S_I are $O\left(\frac{\log^{\frac{3}{2}} m}{m^{\frac{1}{10}}}\right)$-close.

Now, consider a new random variable $X_4 : \Omega \to \{0,1\}^q$, defined as follows: the i'th coordinate of $X_4(B)$ is 1 if and only if $\sum_{I:i\in I,|A_I|\le m^{3/5}} S_I > \sum_{I:i\notin I,|A_I|\le m^{3/5}} S_I$. Let d_4 be the distribution of X_4. We claim that

$$\|d_3 - d_4\|_1 = o(1).$$

To show this, we will need a following simple and well-known fact: let X_1, X_2, Y_1, Y_2 be two pairs of independent random variables, and let d_X denote the distribution of X. Then

$$\|d_{X_1+X_2} - d_{Y_1+Y_2}\|_1 \le \|d_{X_1} - d_{Y_1}\|_1 + \|d_{X_2} - d_{Y_2}\|_1.$$

[1]For the sake of this proof only. No confusion with q as the number of queries should occur.

[2]In fact, there is a technical difficulty we don't deal with, since the expectations of Z_I don't necessarily have to be integer. However, since these expectations are large, at least $\Omega\left(m^{\frac{1}{10}}\right)$, we may, in fact, assume their integrality. We leave the details to the full version of the paper.

Now, making a couple of simple shortcuts, $Pr_\mu\{X_3 \neq X_4\} \leq$

$$\sum_{i=1}^{q} Pr_\mu \left\{ \sum_{I: i\in I, |A_I| \leq m^{3/5}} S_I \;-\; \sum_{I: i\notin I, |A_I| \leq m^{3/5}} S_I \neq \sum_{I: i\in I, |A_I| \leq m^{3/5}} Z_I \;-\; \sum_{I: i\notin I, |A_I| \leq m^{3/5}} Z_I \right\}.$$

For any $1 \leq i \leq q$, the corresponding summand is bounded by the l_1 distance between the distributions of $\sum_{I: i\in I, |A_I| \leq m^{3/5}} S_I - \sum_{I: i\notin I, |A_I| \leq m^{3/5}} S_I$ and of $\sum_{I: i\in I, |A_I| \leq m^{3/5}} Z_I - \sum_{I: i\notin I, |A_I| \leq m^{3/5}} Z_I$. Since the $\{S_I\}$ and the $\{Z_I\}$ are families of independent random variables, the subadditivity property above implies that this distance is at most $O\left(2^q \cdot \frac{\log^{\frac{3}{2}} m}{m^{\frac{1}{10}}}\right) = O\left(\frac{\log^{\frac{3}{2}} m}{m^{\frac{1}{20}}}\right)$. Therefore $\|d_3 - d_4\| = O\left(\frac{\log^{\frac{5}{2}} m}{m^{\frac{1}{20}}}\right) = o(1)$, as claimed.

We are almost done. Since S_I are symmetric random variables, it is easy to see that

$$\|d_4 - \bar{d}_4\|_1 \leq \sum_{i=1}^{q} Pr \left\{ \sum_{I: i\in I, |A_I| \leq m^{3/5}} S_I = \sum_{I: i\notin I, |A_I| \leq m^{3/5}} S_I \right\} \leq$$

$$\sum_{i=1}^{q} Pr \left\{ \sum_{I: i\in I, |A_I| \leq m^{3/5}} Z_I = \sum_{I: i\notin I, |A_I| \leq m^{3/5}} Z_I \right\} + O\left(\frac{\log^{\frac{5}{2}} m}{m^{\frac{1}{20}}}\right).$$

The last inequality follows, as before, from the proximity of S_I to Z_I, and from the subadditivity of distances.

Now, for each i, the RHS probability is the probability that two independent binomial variables with parameters $k_1, k_2 = m/2 - o(m)$ and p are equal. It is easy to see, for instance by the Berry-Esseen theorem, that this probability is at most $O\left(\frac{1}{\sqrt{mp}}\right) = O\left(\frac{1}{m^{\frac{1}{4}}}\right)$. Therefore $\|d_4 - \bar{d}_4\|_1 \leq O\left(\frac{\log^{\frac{5}{2}} m}{m^{\frac{1}{20}}}\right)$.

Finally, let $s = \frac{1}{2}\cdot(d_4 + \bar{d}_4)$. This is a symmetric distribution over $\{0,1\}^q$, with $\|s - d_4\|_1 = \frac{1}{2}\|d_4 - \bar{d}_4\|_1 \leq o(1)$, and we are done. $\square$

5.2 Families of graphs with efficient monotonicity tests

This section describes several families of efficiently testable graphs, including graphs with few edges in the transitive closure, graphs with small width, top-parallel graphs, trees and graphs with small separators. All tests presented have 1-sided error. Hence, we only need to analyze the probability of error for functions that are far from monotone. Throughout the section, we denote the transitive closure of a graph G by $\mathrm{TC}(G)$.

5.2.1 Graphs with sparse transitive closure

We start with an easy test that samples q edges from the transitive closure of the graph where q is a parameter.

<u>TEST $T_2(q)$</u>

1. Pick q edges from the transitive closure of the graph uniformly and independently.
2. For each edge, query its endpoints. Reject if it is violated; otherwise, accept.

Note that test $T_2(q)$ queries at most $2q$ vertices.

Lemma 5.7 *If G is a graph with at most cn edges in $\mathrm{TC}(G)$, then algorithm T_2 with parameter q set to $4c/\varepsilon$ is a 1-sided error $(\varepsilon, 8c/\varepsilon)$-test for monotonicity on G.*

Proof. If a function is ε-far from monotone, by Lemma 2.3, it violates at least $\varepsilon n/2$ edges in the transitive closure. With probability at least $1 - q^{-2} > 2/3$, the test will find one of them. □

5.2.2 Boolean functions over small width graphs

A graph G has *width* w if every set of mutually incomparable vertices has size at most w. The following shows that T_2 can be used to test small width graphs.

Lemma 5.8 *If G is a graph of width w, then algorithm T_2 with q set to $2w/\varepsilon^2$ is a 1-sided error $(\varepsilon, 4w/\varepsilon^2)$-test for monotonicity of Boolean functions on G.*

Proof.[of Lemma 5.8] Let G be a graph of width w and let f be a Boolean labeling of $V(G)$ that is ε-far from monotone. We will show that the number of violated edges in the transitive closure is at least $\varepsilon^2 n^2/(2w) - o(1)$. Since the total number of edges in the graph is at most $n^2/2$, the test will find a violated edge with probability at least $1 - q^{-2} > 2/3$.

Claim 5.9 *If $dist(f, G) \geq d$ for a Boolean labeling f, then there is a set $T, |T| \leq w$, of 0-labeled vertices, such that T is incident to at least d violated pairs.*

Proof.[of claim]If $dist(f, mon_G) \geq d$, by Lemma 2.4, $\mathrm{TC}(G)$ has a matching of violated edges of size d. Call endpoints of the edges in the matching *witnesses*. Let Z be the set of 0-labeled witnesses and let $T \subseteq Z$ be a minimal set of vertices such that $\forall z \in Z, \exists t \in T$

with $z \leq_G t$. Clearly, T contains no comparable pairs, and hence is of size at most w. Each 1-labeled witness is below one of the nodes in Z and hence in T. $\square$

To end the proof, we apply the claim to G and remove the nodes in T from G. We get a new graph for which the restricted f is of distance $\varepsilon n - w$ from monotone (by Lemma 2.1). Repeating until no vertices are left, we observe that the number of violated edges in $\mathrm{TC}\,(G)$ is at least

$\varepsilon n + (\varepsilon n - w) + (\varepsilon n - 2w) + \cdots + (\varepsilon n \mod w) \sim (\varepsilon^2 n^2)/(2w)$. $\square$

5.2.3 Boolean functions over top-parallel graphs

Here we define *top-parallel* graphs and show that they are efficiently testable.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be disjoint graphs. Graph G obtained by connecting G_1 and G_2 *in parallel* is defined by $G = (V_1 \cup V_2, E_1 \cup E_2)$. Graph G obtained by connecting G_1 and G_2 using the *top operation* is defined by $G = (V_1 \cup V_2, E_1 \cup E_2 \cup E_x)$, where $E_x = \{(v_2, v_1) | v_1 \in V_1 \textit{ and } v_2 \in V_2\}$. *Top-parallel* graphs are defined recursively: the 1-vertex graph is top-parallel, and a graph formed by top or parallel operations from two top-parallel graphs is also top-parallel. Note that top-parallel graphs are transitively closed. Examples of top-parallel graphs include the transitive closure of a rooted tree with all edges directed either towards the root or away from the root, and the transitive closure of a complete layered graph.

The rest of this subsection is devoted to proving the following lemma.

Lemma 5.10 *If G is a top-parallel graph, it has a 1-sided $(\varepsilon, 4/\varepsilon^2)$-test for Boolean monotonicity.*

A distribution $\mathcal{D}$ on the edges of a transitive closure of a graph G is called *detecting* if for every Boolean labeling which is ε-far from monotone on G, the probability that $\mathcal{D}$ selects a violated edge is at least ε^2. A graph is *detectable* if it has a detecting distribution.

Claim 5.11 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be disjoint detectable graphs. Then the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ is also detectable.*

Proof. Let $n_1 = |V_1|$ and $n_2 = |V_2|$. Suppose $\mathcal{D}_1$ and $\mathcal{D}_2$ are detecting distributions for G_1 and G_2, correspondingly. We define a distribution $\mathcal{D}$ for G by $\mathcal{D} = p_1\mathcal{D}_1 + p_2\mathcal{D}_2$ where $p_1 = n_1/(n_1 + n_2)$ and $p_2 = n_2/(n_1 + n_2)$. It remains to show that $\mathcal{D}$ is detecting.

Consider a function $f : V_1 \cup V_2 \to \{0,1\}$ with (absolute) distance d from monotone on G. Suppose f restricted to G_1 has distance d_1 to monotone, and f restricted to G_2 has distance d_2. Since there are no edges between G_1 and G_2, $d = d_1 + d_2$. We can lower bound the probability that $\mathcal{D}$ selects a violated edge as follows.

$$\Pr_{\mathcal{D}}[\text{violated edge}] \geq p_1 \cdot \left(\frac{d_1}{n_1}\right)^2 + p_2 \cdot \left(\frac{d_2}{n_2}\right)^2 = \frac{d_1^2}{n_1(n_1+n_2)} + \frac{d_2^2}{n_2(n_1+n_2)}.$$

This quantity is at least $(d_1+d_2)^2/(n_1+n_2)^2$. This fact follows from straightforward algebra:

$$\begin{aligned}
(n_1 d_2 - n_2 d_1)^2 &\geq 0 \\
n_1^2 d_2^2 + n_2^2 d_1^2 &\geq 2 n_1 n_2 d_1 d_2 \\
n_1^2 d_2^2 + n_1 n_2 d_1^2 + n_2^2 d_1^2 + n_1 n_2 d_2^2 &\geq n_1 n_2 d_1^2 + 2 n_1 n_2 d_1 d_2 + n_1 n_2 d_2^2 \\
(n_1+n_2) n_2 d_1^2 + (n_1+n_2) n_1 d_2^2 &\geq n_1 n_2 (d_1+d_2)^2 \\
\frac{d_1^2}{n_1(n_1+n_2)} + \frac{d_2^2}{n_2(n_1+n_2)} &\geq \frac{(d_1+d_2)^2}{(n_1+n_2)^2}
\end{aligned}$$

Thus, $\mathcal{D}$ is a detecting distribution for G. $\square$

Claim 5.12 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be disjoint detectable graphs. Then the graph $G = (V_1 \cup V_2, E_1 \cup E_2 \cup E_x)$, where $E_x = \{(v_2, v_1) | v_1 \in V_1 \text{ and } v_2 \in V_2\}$, is also detectable.*

Proof. Let $n_1 = |V_1|$ and $n_2 = |V_2|$. Suppose $\mathcal{D}_1$ and $\mathcal{D}_2$ are detecting distributions for G_1 and G_2, correspondingly. Denote the uniform distribution over the edges of E_x by U_x. We define a distribution $\mathcal{D}$ for G by $\mathcal{D} = p_1 \mathcal{D}_1 + p_2 \mathcal{D}_2 + p_x U_x$, where

$$p_1 = \frac{n_1^2}{(n_1+n_2)^2}; \quad p_2 = \frac{n_2^2}{(n_1+n_2)^2}; \quad p_x = 1 - p_1 - p_2 = \frac{2 n_1 n_2}{(n_1+n_2)^2}.$$

It remains to show that $\mathcal{D}$ is detecting. Consider a function $f : V_1 \cup V_2 \to \{0,1\}$ with (absolute) distance d to monotone on G. By lemma 2.4, $\mathrm{TC}(G)$ contains d independent edges violated by f. Let d_1 be the number of these edges in $\mathrm{TC}(G_1)$, d_2 be the number in $\mathrm{TC}(G_2)$, and d_x be the number in E_x. Then G_1 contains at least $d_1 + d_x$ 0-labeled vertices, and G_2 contains at least $d_2 + d_x$ 1-labeled vertices. Since every vertex in G_2 is connected to every vertex in G_1, the number of violated edges in E_x is at least $(d_1 + d_x)(d_2 + d_x)$.

Also note that f restricted to G_1 is d_1/n_1-far from monotone, and f restricted to G_2 is d_2/n_2-far from monotone. We can lower bound the probability that $\mathcal{D}$ selects a violated edge as follows.

$$\begin{aligned}\Pr_{\mathcal{D}}[\text{violated edge}] &\geq p_1 \cdot \left(\frac{d_1}{n_1}\right)^2 + p_2 \cdot \left(\frac{d_2}{n_2}\right)^2 + p_x \cdot \frac{d_x^2 + d_x d_1 + d_x d_2 + d_1 d_2}{n_1 n_2} \\ &= \frac{d_1^2 + d_2^2 + 2(d_x^2 + d_x d_1 + d_x d_2 + d_1 d_2)}{(n_1+n_2)^2} \geq \frac{(d_1+d_2+d_x)^2}{(n_1+n_2)^2}.\end{aligned}$$

Since $d = d_1 + d_2 + d_x$, distribution $\mathcal{D}$ is detecting. $\square$

Proof.[of Lemma 5.10] By claims 5.11 and 5.12, G has a detecting distribution $\mathcal{D}$. Consider the test that repeats the following $2\varepsilon^{-2}$ times: query both endpoints of an edge selected according to $\mathcal{D}$ and reject if the edge is violated. If a Boolean function is ε-far from monotone, the test will detect a violated edge with probability at least $1 - e^{-2} > 2/3$. $\square$

5.2.4 Boolean functions over tree-like graphs

Another example of efficiently testable graphs is forests with edges directed arbitrarily. These graphs are defined formally below.

Definition 5.3

1. *A directed graph $G(V, E)$ is* tree-like *if it is obtained by arbitrarily directing each edge of a forest $T = (V, E)$.*
2. *If G is obtained as above from a tree $T = (V, E)$ by choosing a special vertex $r \in V$ and directing the edges along paths from other vertices to r, then G is called a* rooted tree.

Note that a rooted tree is a special case of a tree-like graph. We use the following notation: For a directed acyclic graph $G = (V, E)$ and $v \in V$, let $Low(v) = \{u \in V | u \leq_G v\}$ and $High(v) = \{u | \ v \leq_G u\}$. Observe that in a rooted tree, $Low(x) \cap Low(y) = \emptyset$ for every pair of incomparable vertices x, y. In a tree-like graph, $Low(v) \cap Low(u) = \emptyset$ or $High(v) \cap High(u) = \emptyset$ (or both) for every pair of incomparable vertices x, y. We start with a simple algorithm for testing rooted trees and then treat more general tree-like graphs.

Rooted trees

Here we present an $(\varepsilon, O(poly(1/\varepsilon)))$-test for rooted trees.

Definition 5.4 *Let f be a Boolean labeling of a rooted tree $G = (V, E)$ and $0 < \varepsilon < 1$. A vertex $v \in V$ is ε-bad with respect to f if $f(v) = 0$ and more than ε fraction of vertices in $Low(v)$ are labeled 1 by f. A vertex is ε-good if it is not ε-bad.*

The following lemma is the heart of the algorithm:

Lemma 5.13 *Let f be a Boolean labeling on a rooted tree $G = (V, E)$. If less than an $\varepsilon/2$ fraction of vertices in G are $\varepsilon/2$-bad then f's relative distance to monotone is less than ε.*

Proof. Assume that less than an $\varepsilon/2$ fraction of vertices in G are $\varepsilon/2$-bad with respect to a Boolean labeling f. We can obtain a monotone Boolean labeling f' on G by changing f on less than an ε fraction of the vertices. Let Z be the set of $\varepsilon/2$-good 0-labeled vertices, and S be the set of maximal nodes of Z. Set $f'(x)$ to 0 if $x \in Low(v)$ for some $v \in S$ and to 1 otherwise. Note that by the definition of S, $\cup_{v \in S} Low(v)$ includes every $\varepsilon/2$-good 0-labeled vertex. Hence less than an $\varepsilon/2$ fraction of the vertices (the 0-labeled vertices outside $\cup_{v \in S} Low(v)$) change labels to 1 in f'. Also, as G is a rooted tree, $Low(u) \cap Low(v) = \emptyset$ for all $u, v \in S$ and hence f' differs from f on at most $\varepsilon/2$ fraction of $\cup_{v \in S} Low(v)$ (as each $v \in S$ is $\varepsilon/2$-good). Thus, $dist(f, f') \leq \varepsilon \cdot |V|$.

It remains to show that f' is monotone. Consider nodes x and y where $f'(x) = 0$ and y is below x. Since $f'(x) = 0$, x is in $Low(v)$ for some $v \in S$. Then y is also in $Low(v)$ for the same v, and therefore $f'(y) = 0$. Thus, there are no violated pairs. □

<u>Test for rooted trees, $T_3(\varepsilon)$</u>

1. Query $k = 4/\varepsilon$ vertices uniformly and independently at random.
2. For each queried vertex with label 0, query k vertices below it uniformly and independently at random and reject if a violated pair is found; otherwise, accept.

Lemma 5.14 *Let G be a rooted tree. Then algorithm T_3 is a 1-sided error $\left(\varepsilon, O\left(\varepsilon^{-2}\right)\right)$-test for monotonicity of Boolean functions on G.*

Proof. Clearly the test cannot reject a monotone function. If a Boolean function is ε-far from monotone then, by Lemma 5.13, at least an $\varepsilon/2$ fraction of vertices are $\varepsilon/2$-bad. Hence step 1 of the algorithm will select an $\varepsilon/2$-bad vertex v with probability at least $1 - e^{-2}$. Step 2 will find a vertex with label 1 below v with probability at least $1 - e^{-2}$. Therefore, the failure probability is at most $2e^{-2} < 1/3$. □

Tree-like graphs

This subsection generalizes the rooted tree test to tree-like graphs. One of the difficulties in generalizing the algorithm is that for two incomparable nodes u, v in a tree-like graph, the sets of elements below them might not be disjoint (contrary to the rooted tree case). We will generalize the definition of ε-good after introducing necessary notation. For a directed acyclic graph $G = (V, E)$ and $v \in V$, let $Low(v) = \{u \in V | u \leq_G v\}$ and $High(v) = \{u | \ v \leq_G u\}$. Observe that in a tree-like graph, $Low(v) \cap Low(u) = \emptyset$ or $High(v) \cap High(u) = \emptyset$ (or both) for every pair of incomparable vertices x, y.

Definition 5.5 *Let f be a Boolean labeling of a tree-like graph $G = (V, E)$. The principal cone of v, denoted by $C^*(v)$, is $Low(v)$ if $f(v) = 0$ and $High(v)$ if $f(v) = 1$. We denote by $C(v) = Low(v) \cup High(v)$, the set of all vertices that are comparable to v.*

Definition 5.6 *Let f be a Boolean labeling of a tree-like graph $G = (V, E)$ and $0 < \varepsilon < 1$. A vertex $v \in V$ is ε-bad with respect to f if more than ε fraction of vertices in $C^*(v)$ are labeled $1 - f(v)$ by f; namely, if v belongs to more than $\varepsilon|C^*(v)|$ violated pairs. A vertex is ε-good if it is not ε-bad.*

Lemma 5.15 *Let f be a Boolean labeling on a tree-like graph G, obtained from a forest $T = (V(T), E(T))$. If less than an $\varepsilon/2$ fraction of vertices in G are $\varepsilon/2$-bad then f's relative distance to monotone is less than ε.*

Proof. We may assume w.l.o.g that T is a tree rather than a forest, since it is enough to prove the lemma separately for each connected component. Let ε and f be fixed. Throughout the proof, we call v *good* if it is $\varepsilon/2$-good with respect to f and *bad* if it is $\varepsilon/2$-bad with respect to f. The set of good vertices is denoted by V_G. The main part of the proof is to show that there is a Boolean labeling f' which is monotone on V_G and so that $dist(f, f') \leq \frac{\varepsilon}{2} \cdot |V(T)|$. Given that, the lemma follows by lemma 2.1.

We show inductively how to change f into f' such that f' is monotone on V_G. Our inductive process works in phases. After the kth phase a labeling f_k, and a set $B_k \subseteq V(T)$ are defined, so that $B_{k-1} \subset B_k$ and f_k differs from f_{k-1} only on $B_k - B_{k-1}$. For $k \geq 1$ we denote $\Delta_k = B_k - B_{k-1}$. Also, let $L(v) = \{u | \ f(u) = f(v)\}$, and let $f_0 = f$.

We keep the following invariants for every phase $k \geq 1$:

1. f_k is monotone on $B_k \cap V_G$.

2. The induced undirected graph $T[B_k]$ is connected.

3. Δ_k contains all points that are comparable to any point in B_{k-1} and are not already in B_{k-1}.

4. For every node $v \in B_k$ either $Low(v) \subseteq B_k$ or $High(v) \subseteq B_k$.

5. There is a set of points $V_k^* \subseteq \Delta_k \cap V_G$ such that $C^*(v), C^*(w)$ are pairwise disjoint for every $v, w \in \cup_1^k V_i^*$ and such that f_k differs from f_{k-1} only in points in $\cup_{v \in V_k}(C^*(v) - L(v))$.

Note that if we show that we can construct $B_1, ..., B_k$ and f_k as above with $V_G \subseteq B_k$ we are done as by requirement 1 f_k is monotone on V_G while by requirement 5 $dist(f, f_k) \leq \cup_{i=1}^k \cup_{v \in V_k^*} |C^*(v) - L(v)| \leq \frac{\varepsilon}{2} \Sigma_{i=1}^k \Sigma_{v \in V_k^*} |C^*(v)| \leq \frac{\varepsilon}{2}|T|$ (the 2nd inequality is by the fact that $V_k \subseteq V_G$, the 3rd is by condition 5).

To start, let v be any maximal element among the set of all vertices in V_G that are labeled by '$0'$. Let $B_1 = C(v)$ and f_1 be defined by $f_1(x) = 0$ for all $x \in Low(v)$ and f_1 remains identical to f for every other vertex. Let $V_1^* = \{v\}$. Then, by definition, requirements 1,2,4,5 are met while requirement 3 is met vacuously. Note, if there is no such v then either there is an analogous minimal '$1'$ vertex in V_G or $V_G = \emptyset$ for which the lemma trivially holds.

Assume that we have already constructed $B_1, ..., B_k$, f_k, and $V_1^*, \ldots, V_k^*$ meeting requirements 1,2,3,4,5 for each $i \leq k$. Assume also that $V_G - B_k \neq \emptyset$. Let X be the set of all vertices not in B_k that are adjacent to B_k. Since T is a tree, and using condition 2, for each $x \in X$ there is a unique corresponding $y = y_x \in B_k$ so that $(y, x) \in E(T)$. To obtain the construction for $(k+1)$th phase we do the following for every $x \in X$. Let $x \in X$ and assume w.l.o.g that $y_x <_G x$ (the analogue case is completely symmetric). Note, as $x \notin B_k$ it follows (by condition 4) that $Low(y_x) \subseteq B_k$. There are three cases to consider:

Assume first that there is a good vertex labeled by '$1'$ in $C(x) - B_k$ and let z be a minimal such vertex. If $z \in Low(x)$ we put z in V_{k+1} and add $C(z)$ to B_{k+1}. We set f_{k+1} to be identical to f_k except in $High(z) = C^*(z)$ in which every vertex is labeled by '$1'$. If there is such z but $z \in High(x)$ then add $C(x)$ to B_{k+1}. We then take all such minimal z and define f_{k+1} to be '$1'$ on each vertex in $High(z) = C^*(z)$ for all such z's, and let it be identical to f_k for every other vertex.

The last case is when there is no such z at all. Namely, every good vertex in $C(x) - B_k$ is labeled by '$0'$. In this case we add $C(x)$ to B_{k+1} without adding a new vertex to V_{k+1} or making any changes in f_k.

Note, in all cases we add to B_{k+1} a $C(u)$ for some $u \notin B_k$ and such that $u \leq_G x$ which immediately implies that condition 4 is met. Condition 2 will also be met, as by induction $T[B_k]$ is connected and each $C(u)$ is connected and also connected to B_k. Moreover, if we do this for every $x \in X$, then Δ_{k+1} certainly contains all points that are comparable to B_k and are not already in B_k (note that when we add x as above we also add every $x' >_G x$). Hence requirement 3 holds, too. Also, it is quite clear, by the definition of f_{k+1} that f_{k+1} is monotone on each individual piece $C(u)$ that we add to B_k. It is quite easy to realize that this is true for all B_{k+1} which is condition 1.

It remains to show that requirement 5 holds with the set V_{k+1}. We first note that for any z added to V_{k+1}, $C^*(z)$ is disjoint form B_k and hence from any $C^*(w),\ w \in V_j,\ j < k+1$. To see that assume for the contrary that $C^*(z) \cap B_k \neq \varphi$ and that z was added due to some x as above, and such that $y_x \leq_G x$. Then by our definition of $C^*(z)$ it cannot contain y_x (as $y_x \leq_G x$ and using condition 4). But then there is a cycle in T: There is the edge (x, y_x) while there is a path from x to y_x going through the non empty intersection and B_k (using the fact that B_k is connected).

Assume now that $z \in V_{k+1}$, $C^*(z) = High(z)$ and $C^*(z) \cap C^*(w) \neq \varphi$ for some $w \in V_{k+1}$. Assume that w was added due to some $\tilde{x}$ with a corresponding $y_{\tilde{x}}$. It cannot be that both w and z are added due to the same x, as then both are above x and then there will be two different paths between z and w (one through the intersection and one through x). Hence we may assume that $x \neq \tilde{x}$. This, however, would again result in a in T: There is the edge (x, y_x), while there is another path going from x through $C^*(z)$ to $C^*(w)$ to $\tilde{x}$ and $y_{\tilde{x}}$ and then to y_x through B_k. This completes the proof of the lemma. $\square$

We now present a test for monotonicity on tree-like graphs. The test is essentially identical to the test for rooted trees except for the change due to the alternative definition of "good":

Test for tree-like graphs, $T_4(\varepsilon)$

1. Query $k = 4/\varepsilon$ vertices uniformly and independently.

2. For each queried vertex v, query k vertices in $C^*(v)$ uniformly and independently and reject if a violated pair is found; otherwise, accept.

Lemma 5.16 *Let G be a tree-like graph. Then algorithm T_4 is a 1-sided error $(\varepsilon, 16/\varepsilon^2)$-test for monotonicity of Boolean functions on G.*

Proof. If a Boolean function is ε-far from monotone then, by Lemma 5.15, at least an $\varepsilon/2$ fraction of vertices are $\varepsilon/2$-bad. Hence step 1 of the algorithm will select an $\varepsilon/2$-bad vertex v with probability at least $1 - e^{-2}$. Step 2 will find a vertex with label $1 - f(v)$ in $C^*(v)$ with probability at least $1 - e^{-2}$. Therefore, the failure probability is at most $2e^{-2} < 1/3$. □

5.2.5 A test for graphs with small separators

Here we consider graphs that can be broken into relatively small connected components by removing a few vertices.

Definition 5.7 *Let $\mathcal{U}$ be an infinite family of undirected graphs that is closed under taking subgraphs. We say that $\mathcal{U}$ is k-separable if every n-vertex graph $U \in \mathcal{U}$ can be broken into connected components of size at most $2n/3$ by removing a subset of at most k vertices, called a separator.*

For example, forests are 1-separable, bounded tree-width graphs have bounded separators and planar graphs are $O(\sqrt{n})$-separable [LT79]. In the sequel k might be a sublinear non-decreasing function of n.

Let $G = (V, E)$ be a directed graph. Let U_G be the undirected graph obtained from G by undirecting its edges. Call G k-separable if U_G belongs to a k-separable family of graphs.

Consider a 'standard' tree structure over disjoint subgraphs of G generated by inductively taking out separators. Namely, generate a rooted tree T where each node x in T is associated with a set of vertices $V(x)$ of G. Let V_0 be a separator for U_G of size $\leq k$, and suppose that $U_G(V - V_0)$ has l components. The root x of T is associated with V_0 (i.e., $V(x) = V_0$) and has l children, one for each component. The subtrees of the children are generated recursively from their respective components by the same procedure. The recursion stops at components of size less than $k \log n$. The leaves are associated with vertex sets of their components. Note that the depth of the tree is $O(\log n)$.

Let $r = x_0, x_1, ..., x_j = x$ be the path from the root to a node x in T. Denote $\cup_{i=0}^{j} V(x_i)$ by $Path(x)$. Namely, $Path(x)$ contains all vertices of G associated with x and all vertices from separators that appear on the path from the root of T to x. For a vertex $v \in V$ let $T(v)$ denote the node x of T so that $v \in V(x)$.

We present a 1-sided error test for G using the structure T.

<u>Test for graphs with small separators, $T_4(\varepsilon)$</u>

1. Pick $\frac{4}{\varepsilon}$ nodes of G uniformly and independently.

2. For each node v, query all nodes in $Path(T(v))$. Reject if a violated pair is found; otherwise, accept.

Call a vertex v *bad* if $Path(T(v))$ contains a violated pair.

Claim 5.17 *If a function is ε-far from monotone, at least $\varepsilon/2$ fraction of vertices are bad.*

Proof. Consider a violated pair (v, u). We will prove that either v or u is bad. The claim then follows as the graph has at least $\varepsilon n/2$ vertex-disjoint violated pairs (by Lemma 2.3).

If $T(v)$ and $T(u)$ are on the same path from the root to a leaf in T, then $v \in Path(T(u))$ or $u \in Path(T(v))$. W.l.o.g., suppose $v \in Path(T(u))$, then u is bad because $Path(T(u))$ contains a violated pair (v, u). If $T(v)$ and $T(u)$ are not on the same path from the root to a leaf, they got separated when T was constructed, i.e., some vertex w on a directed path from v to u, in G, is in a common ancestor of $T(v)$ and $T(u)$. Since (v, w) or (w, u) has to be violated, either v or u is bad. $\square$

Lemma 5.18 *Let $G = (V, E)$ be a k-separable n-vertex graph. Then algorithm T_4 is a 1-sided error $\left(\varepsilon, O\left(\frac{k}{\varepsilon} \log n\right)\right)$-test for monotonicity of functions (with general ranges) on G.*

Proof. Whenever step 1 selects a bad vertex, step 2 finds a violated pair. If f is ε-far from monotone, by claim 5.17, step 1 will select a bad vertex with probability at least $1 - e^{-2} > 2/3$. The bound on the number of queries follows form the fact that $Path(T(v))$ contains at most $O(k \log n)$ vertices. $\square$

This generalizes the more efficient tests for Boolean functions over tree-like graphs and bounded-width graphs for which tighter results (by $\log n$ factor) are obtained in lemmas 5.16

and 5.8. It also provides an alternative $(\varepsilon, O(\sqrt{n}\log n))$-test for planar graphs, which performs more queries than the general algorithm from section 4.1, but requires fewer label comparisons. We note that this result cannot be dramatically improved as the general monotonicity test for the line (which is 1-separable) requires $\Omega(\log n)$ queries [Fis].

Chapter 6

Testing Vector Spaces

Obtaining lower bounds for general property tests proved much more difficult than for non-adaptive error case. There are many examples of properties where the gap between the known non-adaptive and adaptive lower bounds is exponential, even though the best-known test for the problem is non-adaptive. Examples of such properties include monotonicity on general graphs, considered in Chapter 4, and monotonicity on the Boolean hypercube, considered in Chapter 5.

To the best of our knowledge, currently there is only one technique in property testing for obtaining adaptive lower bounds from the non-adaptive ones. It is based on the easy observation that every adaptive test can be simulated by a non-adaptive test that asks queries for all possible answers that the adaptive test might get. This allows us to obtain an adaptive lower bound for any property by taking a logarithm of the corresponding non-adaptive lower bound[1]. The obvious disadvantage of this technique is that it gives very weak adaptive bounds when adaptivity does not help significantly.

In this Chapter, we describe a different method for obtaining adaptive lower bounds from non-adaptive ones, which avoids this shortcoming, but applies only to a special class of properties. An additional advantage of our technique is that it also provides a transformation from 1-sided error to 2-sided error tests for this special class of properties. An example of a problem with an exponential gap between the best-known 1-sided error and 2-sided error lower bounds is monotonicity testing on the Boolean hypercube, presented in Section 5.1.

[1]This is tight for some properties.

The class of properties for which our method works is *linear properties*. Recall that a *property* is a collection of strings of a fixed size n. A property is *linear* if it forms a vector space. This Chapter shows that for linear properties, 1-sided error non-adaptive tests are as powerful as general tests.

Theorem 6.1 *Let $\mathbb{F}$ be a finite field and $\mathcal{V} \subseteq \mathbb{F}^n$ be a vector space. For every 2-sided error adaptive $(\varepsilon, \mu_+, \mu_-, q)$-test T for $\mathcal{V}$, there is a 1-sided error non-adaptive $(\varepsilon, 0, \mu_+ + \mu_-, q)$-test T' for $\mathcal{V}$.*

The reduction to simpler tests shifts the error from the positive instances to the negative instances and preserves all other parameters. We perform this reduction in two stages: we first reduce an adaptive test with 2-sided error to an adaptive test with 1-sided error (Theorem 6.3) maintaining the sum of the positive and negative errors $(\mu_+ + \mu_-)$ and then reduce this to a non-adaptive test with 1-sided error (Theorem 6.5) maintaining all parameters. The second reduction was suggested by Madhu Sudan.

A natural test for checking membership in a linear subspace $\mathcal{V}$ is one that is determined by a distribution over sets of constraints in the dual space $\mathcal{V}^{\perp}$. This test chooses a set of constraints from the dual space $\mathcal{V}^{\perp}$ according to this distribution, queries all variables that appear in this set of constraints and accepts or rejects depending on whether the constraints are satisfied or not. Clearly, this is a 1-sided error non-adaptive test. The proofs of Theorem 6.3 and Theorem 6.5 demonstrate that any test of a linear property can be converted into one of the above form maintaining the query complexity and the sum of the errors.

Preliminaries

The following notation, terminology and lemma will be useful in analyzing the reductions.

Any probabilistic test can be viewed as a distribution over deterministic tests and each deterministic test can be represented by a decision tree. Thus, any test T can be represented by an ordered pair $(\mp_T, \mathcal{D}_T)$ where $\mp_T = \{\Gamma_1, \Gamma_2, \ldots\}$ is a set of decision trees and $\mathcal{D}_T$ is a distribution on this set such that on input x, T chooses a decision tree Γ with probability $\mathcal{D}_T(\Gamma)$ and then answers according to $\Gamma(x)$.

We say that a test *detects a violation* if no string in $\mathcal{V}$ is consistent with the answers to the queries. By linearity, it is equivalent to having a constraint α in $\mathcal{V}^{\perp}$ such that $\langle x, \alpha \rangle \neq 0$

for all $x \in \mathbb{F}^n$ which are consistent with the answers to the queries.

Let $\mathcal{V}$ be a vector space. For any leaf l of decision tree Γ, let $\mathcal{V}_l$ be the set of all vectors in $\mathcal{V}$ that are consistent with the answers along the path leading to l. Similarly, for any string $x \in \mathbb{F}^n$, let $\mathcal{V}_l^x$ be the the set of all vectors in $x + \mathcal{V}$ that are consistent with the answers along the path leading to l.

Lemma 6.2 *Let $\mathbb{F}$ be a finite field and $\mathcal{V} \subseteq \mathbb{F}^n$ be a vector space. Let $x \in \mathbb{F}^n$. For any decision tree Γ and a leaf l in Γ, if both $\mathcal{V}_l$ and $\mathcal{V}_l^x$ are non-empty, then $|\mathcal{V}_l| = |\mathcal{V}_l^x|$.*

Proof. Let U be the set of all strings in $\mathcal{V}$ which have the element 0 in all the positions queried along the path leading to l. Since $0^n \in U$, we have that U is non-empty. Observe that if $u \in U$ and $v \in \mathcal{V}_l$, then $u + v \in \mathcal{V}_l$. In fact, if $\mathcal{V}_l \neq \emptyset$, $\mathcal{V}_l = v + U$ for any $v \in \mathcal{V}_l$. Hence, $|\mathcal{V}_l| = |U|$. Similarly, if $\mathcal{V}_l^x \neq \emptyset$, we have that $\mathcal{V}_l^x = y + U$ for any $y \in \mathcal{V}_l^x$. Hence, $|\mathcal{V}_l^x| = |U|$ and the lemma follows. $\square$

6.1 Reduction from 2-sided to 1-sided error

This section shows how to convert a 2-sided error (adaptive) test to a 1-sided error (adaptive) test without increasing the sum of the errors on the positive and negative side and without altering the query complexity.

Theorem 6.3 *Let $\mathbb{F}$ be a finite field and $\mathcal{V} \subseteq \mathbb{F}^n$ be a vector space. For every adaptive $(\varepsilon, \mu_+, \mu_-, q)$-test T for $\mathcal{V}$, there is a 1-sided error adaptive $(\varepsilon, 0, \mu_+ + \mu_-, q)$-test T' for $\mathcal{V}$.*

Proof. Let $T = (\mp_T, \mathcal{D}_T)$ be a 2-sided error (adaptive) (ε, μ, q)-test for $\mathcal{V}$. To convert T to a 1-sided error test, we modify the test so that it rejects if and only if it observes that a constraint in $\mathcal{V}^\perp$ has been violated. We say that a leaf l is labelled *optimally* if its label is 0 when the query answers on the path to l falsify some constraint in $\mathcal{V}^\perp$, and 1 otherwise. We relabel the leaves of each tree Γ in $\mp_T$ *optimally* to obtain the tree Γ_{opt}.

Relabelling produces a 1-sided error test with unchanged query complexity. However, the new test performs well only on "average". To get good performance on every string, we randomize the input x by adding a random vector v from $\mathcal{V}$ to it and perform the test on $x + v$ instead of x. Now we formally define the 1-sided error T' corresponding to T.

Definition 6.1 (1-Sided Error Test) *Given a 2-sided error (adaptive) test T for $\mathcal{V}$, define the test T' as follows: On input x, choose a decision tree Γ according to the distribution $\mathcal{D}_T$ as T does, choose a random $v \in \mathcal{V}$ and answer according to $\Gamma_{\text{opt}}(x+v)$.*

Clearly, T' has 1-sided error as it rejects only if it detects a violation. Also, T' has the same query complexity as T. It remains to check that T' has error $\mu_+ + \mu_-$ on negative instances.

For any $x \in \mathbb{F}^n$ and any test T, let ρ_x^T be the average acceptance probability of test T over all strings in $\mathcal{V}+x$, i.e., $\rho_x^T = ave_{y \in \mathcal{V}+x}\big(\Pr[T(y)=1]\big)$. For notational brevity, we denote $\rho_{0^n}^T$, the average acceptance probability of strings in $\mathcal{V}$, by ρ^T. Observe that for the new test T', for each input x, $\Pr[T'(x)=1] = \rho_x^{T'}$.

The following lemma shows that the transformation to a 1-sided error test given by Definition 6.1 increases the acceptance probability of any string not in $\mathcal{V}$ by at most $\rho^{T'} - \rho^T$. Notice that all vectors in $\mathcal{V}+x$ have the same distance to $\mathcal{V}$. Therefore if x is ε-far from $\mathcal{V}$ then $\rho_x^T \le \mu_-$. Together with Lemma 6.4, it implies that for all vectors that are ε-far from $\mathcal{V}$, the error is low:

$$\Pr[T'(x)=1] = \rho_x^{T'} \le \rho^{T'} - \rho^T + \rho_x^T \le 1 - (1-\mu_+) + \mu_- = \mu_+ + \mu_-.$$

This completes the proof of Theorem 6.3 $\square$

Lemma 6.4 *$\rho^T - \rho_x^T \le \rho^{T'} - \rho_x^{T'}$ for any vector $x \in \mathbb{F}^n$.*

Proof. Let $x \in F^n$. It is enough to prove that relabeling one leaf l of a decision tree Γ in $\mp_T$ *optimally* does not decrease $\rho^T - \rho_x^T$. Then we obtain the lemma by relabelling one leaf at a time to get T' from T. There are two cases to consider.

Case (i) The path to l falsifies some constraint in $\mathcal{V}^\perp$. Then l is relabelled from 1 to 0. This change preserves ρ^T because it only affects strings that falsify some constraint. Moreover, it can only decrease the acceptance probability for such strings. Therefore, ρ_x^T does not increase. Hence, $\rho^T - \rho_x^T$ does not decrease.

Case(ii) The path to l does not falsify any constraint in $\mathcal{V}^\perp$. Then l is relabelled from 0 to 1. Let $\mathcal{V}_l$ and $\mathcal{V}_l^x$ respectively be the set of vectors in $\mathcal{V}$ and $\mathcal{V}+x$ that are consistent with the answers observed along the path to l. Thus, every string in $\mathcal{V}_l \cup \mathcal{V}_l^x$ was rejected before relabeling, but is accepted now. The behavior of the algorithm on the remaining strings in $\mathcal{V}$ and $\mathcal{V}+x$ is unaltered. Hence, the probability ρ^T increases by the quantity $\mathcal{D}_T(\Gamma_1) \cdot \frac{|\mathcal{V}_l|}{|\mathcal{V}|}$. Similarly, ρ_x^T increases by $\mathcal{D}_T(\Gamma_1) \cdot \frac{|\mathcal{V}_l^x|}{|\mathcal{V}|}$.

It suffices to show that $|\mathcal{V}_l| \geq |\mathcal{V}_l^x|$. Since the path leading to l does not falsify any constraint, $\mathcal{V}_l$ is non-empty. If $\mathcal{V}_l^x$ is empty, we are done. Otherwise, both $\mathcal{V}_l$ and $\mathcal{V}_l^x$ are non-empty, and by Lemma 6.2, $|\mathcal{V}_l| = |\mathcal{V}_l^x|$.

□

6.2 Reduction from adaptive to non-adaptive

In this section, we argue that adaptivity does not help to check linear constraints. The intuition behind this is as follows: To check if a linear constraint is satisfied, a test needs to query all the variables that participate in that constraint. Based on any partial view involving some of the variables, the test cannot guess if the constraint is going to be satisfied or not until it reads the final variable. Hence, any adaptive decision based on such a partial view does not help.

Theorem 6.5 *Let $\mathbb{F}$ be a finite field and $\mathcal{V} \subseteq \mathbb{F}^n$ be a vector space. For every 1-sided error adaptive $(\varepsilon, 0, \mu, q)$-test T for $\mathcal{V}$, there is a 1-sided error non-adaptive $(\varepsilon, 0, \mu, q)$-test T' for $\mathcal{V}$.*

Proof. Let T be a 1-sided error (adaptive) $(\varepsilon, 0, \mu, q)$-test for $\mathcal{V}$. Let $\mp_T$ and $\mathcal{D}_T$ be the associated set of decision trees and the corresponding distribution respectively. Since T has 1-sided error, T accepts if it does not detect a violation. Furthermore, we may assume that T rejects if it detects a violation since this can only decrease the acceptance probability of strings not in $\mathcal{V}$. This implies that all the trees in $\mp_T$ are *optimally* labeled. We now define the non-adaptive test T' corresponding to T.

Definition 6.2 (1-Sided Error Non Adaptive test) *Given a 1-sided error (adaptive) test T for $\mathcal{V}$, define the test T' as follows: On input x, choose a random $v \in \mathcal{V}$, query x on all variables that T queries on input v, reject if a violation is detected, otherwise accept.*

T' has 1-sided error because it rejects only if it detects a violation. T' asks the same number of queries as T. Moreover, the queries depend only on the random $v \in \mathcal{V}$ and not on the input x. Hence, the test T' is non-adaptive. The following lemma relates the acceptance probability of T' to the average acceptance probability of T.

Lemma 6.6 *Let T be a 1-sided error (adaptive) test and T' the non-adaptive version of T (as in Definition 6.2). Then, for any string $x \in \mathbb{F}^n$,*

$$\Pr[T'(x) = 1] = ave_{v \in \mathcal{V}} \left(\Pr[T(x+v) = 1] \right).$$

Proof. For any decision tree Γ, let $l_1(\Gamma)$ denote the set of leaves in Γ that are labeled 1. For any leaf l in a decision tree Γ, let $\mathrm{var}(l)$ denote the set of variables queried along the path leading to l in the tree Γ. Following the notation of Lemma 6.2, let $\mathcal{V}_l$ and $\mathcal{V}_l^x$ be the set of all vectors in $\mathcal{V}$ and $x + \mathcal{V}$ respectively that are consistent with the answers along the path leading to l. Also let I_l^x be a binary variable which is set to 1 iff x does not violate any constraint in $\mathcal{V}^\perp$ involving only the variables $\mathrm{var}(l)$. Observe that if test T' chooses the decision tree $\Gamma \in \mp_T$ and the vector $v \in \mathcal{V}$ such that $v \in \mathcal{V}_l$ for some leaf l labeled 1 in the tree Γ, then $I_l^x = 1$ iff $T'(x) = 1$.

The quantity "$ave_{v \in \mathcal{V}} \left(\Pr[T(x+v) = 1] \right)$" can be obtained as follows: First choose a decision tree $\Gamma \in \mp_T$ according to the distribution $\mathcal{D}_T$ and then for each leaf l labeled 1 in Γ, find the fraction of vectors in $x + \mathcal{V}$ that follow the path leading to l. The weighted sum of these fractions is $ave_{v \in \mathcal{V}} \left(\Pr[T(x+v) = 1] \right)$. Thus,

$$ave_{v \in \mathcal{V}} \left(\Pr[T(x+v) = 1] \right) = \sum_{\Gamma \in \mp_T} \mathcal{D}_T(\Gamma) \left(\sum_{l \in l_1(\Gamma)} \frac{|\mathcal{V}_l^x|}{|\mathcal{V}|} \right). \tag{6.1}$$

Now consider the quantity "$\Pr[T'(x) = 1]$". Test T' can be viewed in the following fashion: On input x, T' chooses a random decision tree $\Gamma \in \mp_T$ according to the distribution $\mathcal{D}_T$, it then chooses a leaf l labeled 1 in Γ with probability proportional to the fraction of vectors $v \in \mathcal{V}$ that are accepted along the path leading to l (i.e., $|\mathcal{V}_l|/|\mathcal{V}|$), queries x on all variables in $\mathrm{var}(l)$, accepts if $I_l^x = 1$ and rejects otherwise. This gives us the following expression for $\Pr[T'(x) = 1]$.

$$\Pr[T'(x) = 1] = \sum_{\Gamma \in \mp_T} \mathcal{D}_T(\Gamma) \left(\sum_{l \in l_1(\Gamma)} \frac{|\mathcal{V}_l|}{|\mathcal{V}|} \cdot I_l^x \right) \tag{6.2}$$

From Equations (6.1) and (6.2), we obtain that it suffices to prove that $|\mathcal{V}_l^x| = I_l^x \cdot |\mathcal{V}_l|$ for all leaves l labeled 1 in order to prove the lemma.

Observe that $|\mathcal{V}_l|$ is non-empty since l is labeled 1. Hence, by Lemma 6.2, $|\mathcal{V}_l| = |\mathcal{V}_l^x|$ if $\mathcal{V}_l^x$ is also non-empty. It now suffices to show that $\mathcal{V}_l^x$ is non-empty iff $I_l^x = 1$.

Suppose $\mathcal{V}_l^x$ is non-empty. Then there exists $y \in x + \mathcal{V}$ that does not violate any constraint involving only the variables $\mathrm{var}(l)$. But y and x satisfy the same set of constraints. Hence, x also does not violate any constraint involving only the variables $\mathrm{var}(l)$. Thus, $I_l^x = 1$.

Now, for the other direction, suppose $I_l^x = 1$. Then the values of the variables $\mathrm{var}(l)$ of x do not violate any constraint in $\mathcal{V}^\perp$. Hence, there exists $u \in \mathcal{V}$ that has the same values as x for the variables $\mathrm{var}(l)$. Let $v \in \mathcal{V}_l$. Then, the vector $x - u + v \in x + \mathcal{V}$ has the same values for the variables $\mathrm{var}(l)$ as v. Hence, $\mathcal{V}_l^x$ is non-empty. This concludes the proof of the lemma. $\square$

The above lemma proves that T' inherits its acceptance probability from T. As mentioned earlier, T' inherits its query complexity from T. Hence T' is a 1-sided error non-adaptive $(\varepsilon, 0, \mu, q)$-test for $\mathcal{V}$. $\square$

Chapter 7

Some 3CNF Properties Require a Linear Number of Queries

This chapter shows that there are 3CNF properties, for which every test requires a linear number of queries. After discussing earlier work, we present a self contained proof of the main result in section 7.1. The proofs of the claims needed for the proof follow in sections 7.2-7.4.

Earlier work. There are two published linear lower bounds for property testing. One is the generic bound due to Goldreich et al. [GGR98] and the other is for testing 3-coloring in bounded degree graphs due to Bogdanov, Obata and Trevisan [BOT02]. There is a simple and elegant unpublished linear lower bound observed by Sudan [Personal Communication]. His property consists of polynomials over $\mathbb{F}_n$ of degree at most $n/2$ where each polynomial is given by its evaluation on all elements of the field. It is not hard to see that every non-adaptive 1-sided error test for this property requires linear query complexity. Since the property of low-degree polynomials is linear, our reduction from general to non-adaptive 1-sided error tests from Chapter 6 implies a linear lower bound for adaptive 2-sided tests for this property. Observe that this property is easy to decide once all the input is read, but is not expressible by a family of 3CNF formulas.

Both linear lower bounds of Sudan and Bogdanov et. al [BOT02] capitalize on the existence of inputs that are far from having the property, yet *any* local view of a constant fraction of them can be extended to an element having the property[1]. But if the property

[1] E.g. in Sudan's example any evaluation of a polynomial on d points can be extended to an evaluation

is defined by a kCNF φ this cannot happen. For, clearly, any string that does not have the property must falsify at least one clause of φ. Thus, there is some view of the input of size k, that proves the input does not have the property. Our result shows that in certain cases, finding such a falsified clause requires reading a constant fraction of the input, even if the assignment is far from any satisfying one. Another relevant result is the lower bound of Goldreich and Ron on testing bipartiteness in 3-regular, n-vertex graphs [GR02]. They showed a lower bound of $\Omega(\sqrt{n})$ on the query complexity, yet short witnesses of non-bipartiteness do exist, in the form of odd cycles of length $\mathrm{poly}(\log n)$. Our result strengthens this finding, since in our case the query complexity is linear whereas the witness size is constant.

7.1 Main Theorem

In this section we state and prove the main theorem of Chapter 7, saying that some 3CNF properties are hard to test.

Theorem 7.1 (Main) *There exist* $0 < \delta, \varepsilon < 1$, $0 < \mu < \frac{1}{2}$ *such that for every sufficiently large* n*, there is a* 3CNF *formula* φ *on* n *variables such that every adaptive* $(\varepsilon, \mu_+, \mu_-, q)$*-test for* φ *with the sum of errors* $\mu_+ + \mu_- \leq \mu$ *asks at least* $q = \delta n$ *queries.*

To prove Theorem 7.1, we find hard 3CNF formulas that define linear properties. Recall that Theorem 6.1 shows that adaptivity and 2-sided error do not help to test linear properties. We use this theorem for properties over the binary alphabet, namely with $\mathbb{F} = GF(2)$. Equipped with this theorem, we can restrict our attention to proving Theorem 7.1 for non-adaptive 1-sided error tests, provided that the formulas we work with define linear properties. The goal of this section is to prove such a theorem, stated below:

Theorem 7.2 (Main for 1-Sided Non-adaptive) *There exist* $0 < \delta, \varepsilon < 1$, $0 < \mu < \frac{1}{2}$ *such that for every sufficiently large* n*, there is a* 3CNF *formula* φ *on* n *variables which defines a linear property such that every 1-sided error non-adaptive* ε*-test for* φ *with error* μ *asks at least* δn *queries.*

Proof. We find linear properties that are hard to test and then represent them by CNFs. Consider a vector space $\mathcal{V} \subseteq \{0,1\}^n$. Denote the dual space by $\mathcal{V}^{\perp}$. Let $\mathcal{A} = (A_1, \ldots, A_m)$

of a polynomial of degree $d' > d$. Thus, seeing $n/2 - 1$ values of the polynomial still does not mean the polynomial has degree $n/2$.

be a basis for $\mathcal{V}^\perp$. Let $|x|$ denote the weight of vector $x \in \{0,1\}^n$. The ith coordinate of x is denoted by x_i. For two vectors $x, y \in \{0,1\}^n$, let $\langle x, y \rangle = \sum_{i=1}^n x_i y_i \mod 2$. By definition, $\mathcal{V} = \{x | \langle x, A_i \rangle = 0 \text{ for all } A_i \in \mathcal{A}\}$. We can view $\mathcal{A}$ as a collection of linear constraints on Boolean variables $x_1, \ldots, x_n$. The following definition introduces linear formulas formed by linear constraits.

Definition 7.1 (dLIN Formulas) *A* linear *(*LIN*) Boolean formula is a conjunction of constraints, where every constraint is satisfied if and only if the variables in the constraint add up to* $0 \mod 2$. *If all constraints contain at most d literals, the formula is a dLIN.*

Thus, viewing each A_i as a *constraint*, we can represent $\mathcal{V}$ as a dLIN formula where $d = \max_{A_i \in \mathcal{A}} |A_i|$. We work with an arbitrary constant d and later show how to reduce it to 3. Since each 3LIN formula has an equivalent 3CNF, it is enough to find hard 3LINs.

We now present sufficient conditions for a vector space to be hard to test. To understand the conditions, keep in mind that later we employ Yao's minimax principle to show that all vector spaces satisfying these conditions are hard for 1-sided non-adaptive tests. Yao's principle states that to prove that each low-query probabilistic test fails on some input, it is enough to give a distribution on the inputs on which each low-query deterministic test fails. We are only interested in 1-sided error tests which, by definition, have to accept unless no vector in the tested vector space satisfies the answers to the queries. Therefore, to show that a vector space satisfying our conditions is hard, we need to exhibit a distribution on vectors which are far from the vector space, such that every low-query deterministic non-adaptive test on this distribution fails to determine with non-negligible probability that the input violates the constraints of the vector space.

Definition 7.2 (Hard Linear Properties) *Let $\mathcal{V} \subseteq \{0,1\}^n$ be a vector space and let $\mathcal{A}$ be a basis for $\mathcal{V}^\perp$. Fix $0 < \varepsilon, \mu < 1$.*

- *$\mathcal{A}$ is* ε-separating *if every $x \in \{0,1\}^n$ that falsifies exactly one constraint in $\mathcal{A}$ has $|x| \geq \varepsilon n$.*
- *$\mathcal{A}$ is* (q, μ)-local *if every $\alpha \in \{0,1\}^n$ that is a sum of at least μm vectors in $\mathcal{A}$ has $|\alpha| \geq q$.*

Notice that if $\mathcal{A}$ is ε-separating, each string x falsifying exactly one constraint in $\mathcal{A}$ is ε-far from $\mathcal{V}$. To see why, let $y \in \mathcal{V}$. Then $x + y$ falsifies exactly one constraint in $\mathcal{A}$. Since $\mathcal{A}$ is ε-separating, $dist(x, y) = |x + y| \geq \varepsilon n$. By definition, $dist(x, \mathcal{V}) \geq \varepsilon n$.

For the proof that every vector space satisfying the above conditions is hard to test, our bad distribution that foils low-query tests is over strings that falsify exactly one constraint. The falsified constraint is chosen uniformly at random. The first condition ensures that the distributions is over vectors which are ε-far from the vector space.

The second condition ensures that the distribution is hard to test. To get the intuition, suppose the second condition is violated. Then a μ fraction of the constraints sums up to a low-weight vector, and the sum represents a constraint on fewer than q variables. Querying variables in the new constraint would allow a test running on our bad distribution to deduce that some constraint is violated with probability at least μ. The second condition disallows this or, intuitively, ensures that to "get information" about a fraction μ of the constraints in $\mathcal{A}$, a test needs at least q queries.

The following theorem, proved in section 7.2, shows that any linear space conforming to definition 7.2 is hard for 1-sided error non-adaptive tests.

Theorem 7.3 (Definition 7.2 $\Rightarrow$ Lower Bound) *Fix $0 < \varepsilon < 1$, $0 < \mu < \frac{1}{2}$. Let $\mathcal{V} \subseteq \{0,1\}^n$ be a vector space. If $\mathcal{V}^{\perp}$ has an ε-separating (q, μ)-local basis $\mathcal{A} = (A_1, \ldots, A_m)$, then every non-adaptive 1-sided error ε-test for $\mathcal{V}$ with error $1 - 2\mu$ requires q queries.*

Theorem 7.3 shows that every linear property conforming to definition 7.2 is hard to test. The following theorem assures us that such linear properties exist. The proof of this theorem, which uses the probabilistic method, appears in section 7.3.

Theorem 7.4 (Hard Linear Properties Exist) *There exist integer $d > 0$ and constants μ, ε, δ, such that for all sufficiently large n there is a collection $\mathcal{A}_n \subset \{0,1\}^n$ of vectors of weight at most d which is linearly independent, ε-separating and $(\delta n, \mu)$-local.*

We now have dLIN formulas that are hard to test. The following reduction brings d down to 3 while preserving the properties of definition 7.2 (with smaller constants).

Theorem 7.5 (Reduction to 3CNFs) *Every linearly independent, ε-separating, $(\delta n, \mu)$-local $\mathcal{A} \subset \{0,1\}^n$ of vectors of weight at most d can be converted to a linearly independent, $\varepsilon^\star$-separating, $(\delta^\star n^\star, \mu^\star)$-local $\mathcal{A}^\star \subset \{0,1\}^{n^\star}$ of vectors of weight at most 3. If ε, δ, μ are strictly positive constants, so are $\varepsilon^\star, \delta^\star, \mu^\star$.*

Theorem 7.5 is proved in section 7.4. Recall that a 3LIN formula can be defined by a 3CNF. Theorem 7.2 follows from definition 7.2 and theorems 7.3–7.5. □

This completes the proof of the Main Theorem 7.1, showing that there are 3CNF properties that require a linear number of queries.

7.2 Lower bounds for non-adaptive 1-sided error tests

This subsection proves Theorem 7.3.

Proof of Theorem 7.3. We employ Yao's minimax principle. It states that to prove that every q-query randomized test fails with probability more than δ it is enough to exhibit a distribution $\mathcal{B}$ on the inputs for which every q-query deterministic test fails with probability more than δ.

For $i = 1 \dots m$ let $\mathcal{B}_i$ be the uniform distribution over n-bit strings that falsify constraint A_i and satisfy the rest. The distribution $\mathcal{B}$ is the uniform distribution over $\mathcal{B}_i$'s. The comment after definition 7.2 shows that distribution $\mathcal{B}$ is over strings which are ε-far from $\mathcal{V}$. Lemma 7.6 demonstrates that every low complexity deterministic test is likely to fail on $\mathcal{B}$, which completes the proof of Theorem 7.3. □

Lemma 7.6 *Let T be a deterministic 1-sided error non-adaptive test with $< q$ queries. If $\mathcal{A}$ is (q, μ)-local then* $\Pr_{x \leftarrow \mathcal{B}}[T(x) = 0] < 2\mu$.

Proof. Let Q be the set of queries posed by T. A query to variable x_i is viewed as a vector of weight 1 in $\{0,1\}^n$ which is 1 at coordinate i and 0 everywhere else. Observe that since T has 1-sided error, it has to accept if there is a vector in $\mathcal{V}$ consistent with the answers to the queries. By linearity, this is equivalent to saying that T rejects a vector in $\mathcal{B}$ only if the falsified constraint can be expressed as a linear combination of queries and remaining constraints. Thus, we need to show that $< 2\mu$ fraction of constraints in $\mathcal{A}$ can be expressed as a linear combination of queries and remaining constraints.

Any such constraint belongs to some set $C \subseteq \mathcal{A}$ with $\sum_{c \in C} c \in span(Q)$. We show that fewer than $2\mu m$ constraints in $\mathcal{A}$ are in such sets. Let Γ be the family of such sets, i.e., of subsets of $\mathcal{A}$ that sum up to a vector $\alpha \in span(Q)$.

It remains to show $\left|\bigcup_{C\in\Gamma} C\right| < 2\mu m$. Observe that if $\alpha_1, \alpha_2 \in span(Q)$, so does $\alpha_1 + \alpha_2$. In terms of C's this implies that if $C_1, C_2 \in \Gamma$, so is $C_1 \triangle C_2$[2]. Since $|Q| < q$ and $\mathcal{A}$ is (q,μ)-local, $|C| \leq \mu m$ for all $C \in \Gamma$. We can now apply Lemma 7.7 to conclude that $\left|\bigcup_{C\in\Gamma} C\right| < 2\mu m$. □

Lemma 7.7 *Let $\Gamma = \{C | C \subseteq [m]\}$ be a non-empty family of subsets of $[m]$ such that Γ is closed under symmetric difference and for all sets C in Γ, $|C| \leq w$. Then $\left|\bigcup_{C\in\Gamma} C\right| < 2w$.*

Proof. Suppose $x \in C$ for some $C \in \Gamma$. Observe that for any set C' in Γ (including C) either $x \in C'$ or $x \in C \triangle C'$ but not both. Since Γ is closed under symmetric difference and $C' = C \triangle (C \triangle C')$, each element in $\bigcup_{C\in\Gamma} C$ occurs in exactly half of the sets of Γ. Therefore,

$$\frac{|\Gamma|}{2} \cdot \left|\bigcup_{C\in\Gamma} C\right| = \sum_{C\in\Gamma} |C| \leq (|\Gamma| - 1)w < |\Gamma| w.$$

The first inequality holds because the empty set belongs to Γ, and $|C| \leq w$ for all other C in Γ. Since $|\Gamma| > 0$, we conclude that $|\bigcup_{C\in\Gamma} C| < 2w$. □

7.3 Random codes require a linear number of queries

In this subsection we prove Theorem 7.4. In particular, we show that a random (c,d)-regular code with high probability obeys definition 7.2, for large enough constants c, d. We start by defining such codes, originally introduced and analyzed by Gallager [Gal63].

7.3.1 Random regular codes

Let $G = \langle L, R, E\rangle$ be a bipartite multi-graph, with $|L| = n, |R| = m$, and let $d(v)$ be the degree of a vertex v. G is called (c,d)*-regular* if for all $v \in L$, $d(v) = c$, and for all $v \in R$, $d(v) = d$. A random (c,d)-regular graph with n left vertices and $m = \frac{c}{d}n$ right vertices, is obtained by selecting a random matching between cn "left" nodes, and $dm = cn$ "right" nodes. Collapse c consecutive nodes on the left to obtain n c-regular vertices, and collapse d consecutive nodes on the right to obtain m d-regular vertices. Notice that the resulting graph may be a multi-graph (i.e. have multiple edges between two vertices). The code associated with G is obtained by letting R define $\mathcal{C}^{\perp}$, as in the following definition.

[2]For sets A, B, the symmetric difference of A and B, $A \triangle B = \{x | x \in A \text{ and } x \notin B\} \cup \{x | x \notin A \text{ and } x \in B\}$.

Definition 7.3 *Let $G = \langle L, R, E \rangle$ be a bipartite multi-graph, with $|L| = n, |R| = m$. Associate a distinct Boolean variable x_i with any $i \in L$. For each $j \in R$, let $N(j) \subseteq L$ be the set of neighbors of j. The j'th constraint is $A_j = \sum_{i \in N(j)} x_i \mod 2$. Let $\mathcal{A}(G)$ be the $m \times n$ matrix where the jth row of $\mathcal{A}(G)$ is A_j. The code defined by G is*

$$\mathcal{C}(G) = (\mathcal{A}(G))^{\perp} = \{x \in \{0,1\}^n | \mathcal{A}(G) \cdot x = \vec{0}\}.$$

A random (c, d)-regular code is obtained by taking $\mathcal{C}(G)$ as in the previous definition, for G a random (c, d)-regular graph. Notice that a variable may appear several times in a constraint.

7.3.2 Some expansion properties of random regular graphs

To prove $\mathcal{C}(G)$ obeys definition 7.2, we use standard arguments about expansion of the random graph G. We reduce each requirement on $\mathcal{A}(G)$ to a requirement on G, and then show that the expansion of a random G implies that it satisfies the requirements. We need the following notions of neighborhood and expansion.

Definition 7.4 (Neighbors) *Let $G = \langle \mathcal{V}, E \rangle$ be a graph. For $S \subset \mathcal{V}$, let*

- *$N(S)$ be the set of neighbors of S.*
- *$N^1(S)$ be the set of* unique *neighbors of S, i.e. the set of vertices with exactly one neighbor in S.*
- *$N^{odd}(S)$ be the set of neighbors of S with an* odd number *of neighbors in S.*

Notice that $N^1(S) \subseteq N^{odd}(S)$.

Definition 7.5 (Expansion) *Let $G = \langle L, R, E \rangle$ be a bipartite graph with $|L| = n, |R| = m$.*

- *G is called an (λ, γ)-*right expander *if*

$$\forall S \subset R,\ |S| \leq \gamma n,\ |N(S)| > \lambda \cdot |S|.$$

- *G is called an (λ, γ)-*right unique neighbor expander *if*

$$\forall S \subset R,\ |S| \leq \gamma n,\ |N^1(S)| > \lambda \cdot |S|.$$

- *G is called an* (λ, γ)-right odd expander *if*

$$\forall S \subset R,\ |S| \geq \gamma n,\ |N^{odd}(S)| > \lambda \cdot |S|.$$

Notice that expanders and unique neighbor expanders discuss subsets of size *at most* γn, whereas odd expanders discuss subsets of size *at least* γn. Left expanders (all three of them) are defined analogously by taking $S \subset L$ in definition 7.5.

The following lemmas are proved using standard techniques for analysis of expansion of random graphs, such as those appearing in e.g. [CS88, Spi95]. The proofs appear in subsection 7.3.3.

Lemma 7.8 *There exists a constant* $r > 0$ *such that for any integers* $c \geq 5, d \geq 2$*, a random* (c, d)*-regular graph is with high probability a* $(1, r \cdot d^{-2})$*-left unique neighbor expander.*

Lemma 7.9 *For any odd integer* c*, any constants* $\mu > 0, \delta < \mu^c$*, and any integer* $d > \frac{2\mu c^2}{(\mu^c - \delta)^2}$*, a random* (c, d)*-regular graph is with high probability a* (δ, μ)*-right odd expander.*

7.3.3 Proofs of expansion Lemmas 7.8 and 7.9

Proof of Lemma 7.8. We need a couple of lemmas, the proof of which will follow.

Lemma 7.10 *For any integers* $c \geq 2, d$*, and any constant* $\alpha < c-1$*, a random* (c, d)*-regular bipartite graph with* n *left vertices, is with high probability a* (α, ε)*-left expander, for any* ε *satisfying*

$$\varepsilon \leq \left(2e^{(1+\alpha)} \cdot \left(\frac{\alpha d}{c}\right)^{(c-\alpha)}\right)^{-\frac{1}{c-\alpha-1}} \tag{7.1}$$

Lemma 7.11 *Let* G *be a* (c, d)*-regular bipartite graph. If* G *is an* (α, ε)*-left expander, then* G *is an* $(2\alpha - c, \varepsilon)$*-left unique neighbor expander.*

We do not try to optimize constants. Let $\alpha = \frac{c+1}{2}$, Noticing that for $c \geq 5$, $\frac{c}{2} < \alpha < c-1$. By lemma 7.10, G is a (α, ε)-right expander for any ε satisfying equation (7.10).

For our selection of α, and any $c \geq 5$, the following inequalities can be verified:

$$\frac{(1+\alpha)}{(c-\alpha-1)} \leq 3$$

$$\frac{\alpha}{c} \leq 2/3$$

$$\frac{(c-\alpha)}{(c-\alpha-1)} \leq 2$$

Hence setting $\varepsilon = (100 \cdot d)^{-2}$ satisfies equation (7.10). Finally, by lemma 7.11, we get that G is whp a $(1, rd^{-2})$-left unique neighbor expander. $\square$

Proof of Lemma 7.10. Let BAD be the event that the random graph is *not* an expander. This means there is some $S \subset L, |S| \leq \varepsilon n$ such that $|N(S)| \leq \alpha \cdot |S|$.

Fix sets $S \subset L, T \subset R,\ |S| = s \leq \varepsilon n, |T| = \alpha s$, and let B_s be the event that all edges leaving S land inside T. We upper-bound the probability of this bad event.

$$\Pr[B_s] \quad = \quad \prod_{i=0}^{c \cdot s-1} \frac{\alpha ds - i}{cn - i} \leq \left(\frac{\alpha ds}{cn}\right)^{cs}$$

The inequality follows as long as $\alpha ds < cn$. We now use a union bound over all sets $S \subset L\ \ |S| = s \leq \varepsilon n$ and all sets $T \subset R,\ \ |T| = \alpha s$. Let κ be the constant $\kappa = e^{1+\alpha} \cdot \left(\frac{\alpha d}{c}\right)^{c-\alpha}$.

$$\begin{aligned}
\Pr[BAD] &\leq \sum_{s=1}^{\varepsilon n} \binom{n}{s} \cdot \binom{m}{\alpha s} \cdot \Pr[B_s] \\
&\leq \sum_{s=1}^{\varepsilon n} \left(\frac{en}{s}\right)^{s} \cdot \left(\frac{em}{\alpha s}\right)^{\alpha s} \cdot \left(\frac{\alpha ds}{cn}\right)^{cs} \\
&= \sum_{s=1}^{\varepsilon n} \left[e^{1+\alpha} \cdot \left(\frac{\alpha d}{c}\right)^{c-\alpha} \cdot \left(\frac{s}{n}\right)^{c-\alpha-1} \right]^{s} \\
&= \sum_{s=1}^{\varepsilon n} \left[\kappa \cdot \left(\frac{s}{n}\right)^{c-\alpha-1} \right]^{s} \qquad (7.2)
\end{aligned}$$

By definition of $\alpha,\ c - \alpha - 1 > 0$, hence $\left(\frac{s}{n}\right)^{c-\alpha-1} \leq 1$. Set

$$\varepsilon \leq (2\kappa)^{\frac{-1}{(c-\alpha-1)}} = \left(2e^{(1+\alpha)} \cdot \left(\frac{\alpha d}{c}\right)^{(c-\alpha)}\right)^{-\frac{1}{c-\alpha-1}} \qquad (7.3)$$

For this value of ε, each term of the sum (7.2) is at most 1/2. Set $\lambda = \min\{\frac{1}{3}, \frac{c-\alpha-1}{2}\}$, and split the sum (7.2) into two sub-sums.

$$
\begin{aligned}
\Pr[BAD] &\leq \sum_{s=1}^{\varepsilon n} \left[\kappa \cdot \left(\frac{s}{n}\right)^{c-\alpha-1}\right]^s \\
&\leq \sum_{s=1}^{n^\lambda} \left[\kappa \cdot \left(\frac{s}{n}\right)^{c-\alpha-1}\right]^s + \sum_{s=n^\lambda}^{\varepsilon n} \left[\kappa \cdot \left(\frac{s}{n}\right)^{c-\alpha-1}\right]^s \\
&\leq n^\lambda \cdot \kappa \cdot n^{(\lambda-1)2\lambda} + n \cdot 2^{-n^\lambda} \\
&= \kappa \cdot n^{-\lambda+2\lambda^2} + n \cdot 2^{-n^\lambda} \\
&\leq \kappa \cdot n^{-1/9} + n \cdot 2^{-n^\lambda} = o(1)
\end{aligned}
$$

We conclude that with high probability, G is an (α, ε)-left expander. □

Proof of Lemma 7.11. Let $S \subset L, |S| \leq \varepsilon|L|$. Then by expansion we get

$$\alpha \cdot |S| < |N(S)|.$$

Any neighbor of S that is not a unique neighbor, must be touched by at least 2 edges leaving S. Since the left degree of G is c, we get

$$|N(S)| \leq |N^1(S)| + \frac{c \cdot |S| - |N^1(S)|}{2} = \frac{c \cdot |S| + |N^1(S)|}{2}.$$

Combining the two equations, we get our claim. □

Proof of Lemma 7.9. In the proof, we make use of the following theorem (see [MR95]).

Theorem 7.12 (Azuma's Inequality) *If $X_0, \ldots, X_t$ is a martingale sequence such that $|X_i - X_{i+1}| \leq 1$ for all i, then*

$$Pr[|X_t - X_0| \geq \lambda\sqrt{t}] \leq 2e^{-\lambda^2/2}.$$

Fix $T \subseteq R$ $|T| = t \geq \mu m$. Let $X = |N^{odd}(T)|$. We start by computing $E[X]$. For $i = 1 \ldots n$, let X_i be the random variable indicating whether vertex $i \in L$ is in $N^{odd}(T)$. Clearly $X = \sum_{i=1}^n X_i$, so by the linearity of expectation, we need only compute $E[X_i]$. Recall that $cn = dm$, Let $\mathrm{odd}(c) = \{1, 3, 5, \ldots, c\}$ be the set of positive odd integers $\leq c$, and notice that $c \in \mathrm{odd}(c)$ because c is odd.

$$
\begin{aligned}
E[X_i] &= \frac{\sum_{i \in \mathrm{odd}(c)} \binom{\mu dm}{i} \cdot \binom{(1-\mu)dm}{c-i}}{\binom{cn}{c}} \\
&\geq \frac{\binom{\mu cn}{c}}{\binom{cn}{c}} = \mu^c - O(\frac{1}{n})
\end{aligned}
$$

We conclude by linearity of expectation:

$$
E[X] \geq \mu^c \cdot n - O(1)
$$

We now make use of the following edge-exposure martingale to show concentration of X around its expectation. Fix an ordering on the μdm edges leaving T, and define a sequence of random variables $Y_0, \ldots Y_{\mu dm}$ as follows: Y_i is the random variable that is equal to the expected size of $N^{odd}(T)$ after the first i edges leaving T have been revealed. By definition, $Y_{\mu dm} = X$, $Y_0 = E[X]$, and the sequence is a martingale, where $|Y_i - Y_{i+1}| \leq 1$ for all $i \leq \mu dm$. Since $d > \frac{2\mu c^2}{(\mu^c - \delta)^2}$, we apply Azuma's inequality (Theorem 7.12) and get:

$$
\begin{aligned}
\Pr[X \leq \delta n] &\leq \Pr[|Y_{\mu dm} - Y_0| \geq (\mu^c - \delta)n] \\
&= \Pr[|Y_{\mu dm} - Y_0| \geq (\mu^c - \delta)\frac{d}{c}m] \\
&\leq 2e^{-\frac{d(\mu^c - \delta)^2}{2\mu c^2} \cdot m} \leq 2e^{-(1+\varepsilon)m}
\end{aligned}
$$

Where $\varepsilon = \frac{d(\mu^c-\delta)^2}{2\mu c^2} - 1 > 0$. There are at most 2^m possible sets $T \subseteq R$, so a union bound gives:

$$
\Pr[\exists T \subset R \ |T| \geq \mu m \mid \sum_{j \in T} A_j| \leq \delta n] \leq 2^m \cdot 2e^{-(1+\varepsilon)m} = o(1)
$$

We conclude that $\mathcal{A}(G)$ is whp a (δ, μ)-right odd expander. $\square$

7.3.4 Random codes are hard to test

We are ready to prove Theorem 7.4.

Lemma 7.13 *For any odd integer $c \geq 5$, there exists an integer $d > c$, and constants*

$\varepsilon, \delta, \mu > 0$*, such that for a random* (c, d)*-regular graph* G*, the set* $\mathcal{A}(G)$ *is with high probability (i) linearly independent, (ii)* $(\delta n, \mu)$*-local, and (iii)* ε*-separating.*

Proof of Theorem 7.4. Fix $c = 5$. Let $d, \varepsilon, \delta, \mu$ be as in Lemma 7.13. The theorem follows. $\square$

Proof of Lemma 7.13. Given odd $c \geq 5$ we will define the constants $d, \varepsilon, \delta, \mu$ throughout the course of the proof.

(i) We need to show that adding up any subset of $\mathcal{A}(G)$ cannot yield $\vec{0}$. Since we are working modulo 2, this is equivalent to proving

$$\forall T \subseteq R, \quad N^{odd}(T) \neq \emptyset.$$

For small T we use unique neighbor expansion, and for large T we use odd neighbor expansion.

Fix c, and reverse the roles of left and right in lemma 7.8. We conclude the existence of constant $r > 0$, such that for any $d \geq 5$, G is with high probability a $(1, r \cdot c^{-2})$-right unique neighbor expander. This implies that if $|T| \leq r \cdot c^{-2} \cdot |R|$, then $N^{odd}(T) \neq \emptyset$ because $N^{odd}(T) \supseteq N^1(T)$ and $N^1(T) \neq \emptyset$.

Lemma 7.9 says that for any $\mu > 0$, and large enough d, all sets of size at least μm have nonempty odd neighborhood. (Actually, the lemma shows that the odd neighborhood is of linear size, which is more than what we need here.) Fixing μ, δ, d to the following values completes the proof of the first claim:

$$\mu = r \cdot c^{-2}; \quad \delta = \mu/2; \quad d > \frac{2\mu c^2}{(\mu^c - \delta)^2}.$$

(ii) Notice that if $T \subseteq R$, then $N^{odd}(T)$ is exactly the support of $\sum_{j \in T} A_j$. Thus, it suffices to show that $N^{odd}(T)$ is large for large subsets T.

By the definition of d, μ, δ from part (ii) and by lemma 7.9 G is whp a $(\delta n, \mu)$-right odd expander. This means $\mathcal{A}(G)$ is $(\delta n, \mu)$-local. Part *(ii)* is proved.

(iii) Let G_{-j} be the graph obtained from G by removing vertex $j \in R$ and all edges touching it. Since $\mathcal{A}(G)$ is linearly independent, it is sufficient to show that $\mathcal{C}(G_{-j})$ has no element of Hamming weight $< \varepsilon n$.

Let x be a non-zero element of $\mathcal{C}(G_{-j})$, and let $S_x \subseteq L$ be the set of coordinates at which x is 1. Consider the graph G_{-j}. In this graph, the set of unique neighbors of S_x is empty because $x \in \mathcal{C}(G_{-j})$ (otherwise, some $j' \in N^1(S_x)$, so $\langle A_{j'}, x\rangle = 1$, a contradiction.) Thus,

$$N^1(S_x) \subseteq \{j\} \tag{7.4}$$

where $N^1(S_x)$ is the set of unique neighbors of S_x in G. Clearly, $|S_x| > 1$ because the left degree of G is $c > 1$. But if $|S_x| \leq r \cdot d^{-2} \cdot n$ then by lemma 7.8 $|N^1(S_x)| \geq |S_x| > 1$, in contradiction to equation (7.4). We conclude that for any $x \in \mathcal{C}(G_{-j})$, $|x| \geq r \cdot d^{-2}$, so $\mathcal{A}(G)$ is ε-separating for ε satisfying:

$$\varepsilon \leq r \cdot d^{-2}.$$

Part *(iii)* is completed, and with it the theorem. ∎

7.4 Reducing dLIN to 3LIN

This section proves Theorem 7.5 which directly follows from the final theorem of this section. The randomized construction from section 7.3 produces d-linear formulas which are hard to test for some constant d. We would like to make d as small as possible. This section obtains 3-linear hard to test formulas. First we give a reduction from d-linear to $\lceil \frac{d}{2} \rceil + 1$-linear formulas, and then apply it $\log d$ times to get 3-linear formulas.

Let φ be a d-linear formula on variables in $X = \{x_1, \ldots, x_n\}$. The reduction maps φ to a $(\lceil \frac{d}{2} \rceil + 1)$-linear formula on variables $X \cup Z$ where Z is a collection of new variables $\{z_1, \ldots, z_m\}$. For each constraint c_i, say $x_1 \oplus \ldots \oplus x_d = 0$, in φ, two constraints, c_i^1 and c_i^2 are formed: $x_1 \oplus \ldots \oplus x_{\lceil \frac{d}{2} \rceil} \oplus z_i = 0$ and $x_{\lceil \frac{d}{2} \rceil + 1} \oplus \ldots \oplus x_d \oplus z_i = 0$. Let $\mathcal{V} \subseteq \{0,1\}^n$ be the vector space of vectors satisfying φ, and let $\mathcal{A}$ be an m-dimensional basis for the vector space $\mathcal{V}^\perp$ of constraints. Define $\mathcal{R}(\mathcal{A})$ to be the collection of $2m$ vectors in $\{0,1\}^{n+m}$ formed by splitting every constraint in $\mathcal{A}$ in two, as described above.

The following three lemmas show that the reduction preserves the properties which make the formula hard to test. A parameter name prime denotes the value of the parameter after one application of the reduction: for example, $m' = 2m$, $n' = m + n$, and $d' = \lceil \frac{d}{2} \rceil + 1$.

Lemma 7.14 *$\mathcal{R}(\mathcal{A})$ is independent.*

Proof. It is enough to prove that no set of constraints in $\mathcal{R}(\mathcal{A})$ sums up to 0. Let $C \in \mathcal{R}(\mathcal{A})$. If only one of the two constraints involving a new variable z appears in C, then the sum of vectors in C has 1 in z's position. If, on the other hand, all constraints appear in pairs, then the sum of vectors in C is equal to the sum of the constraints in $\mathcal{A}$ from which C's constraints were formed. By independence of old constraints, this sum is not 0. $\square$

Lemma 7.15 *If $\mathcal{A}$ is ε-separating, then $\mathcal{R}(\mathcal{A})$ is ε'-separating where $\varepsilon' = \frac{\varepsilon}{1+m/n}$.*

Proof. Let x' be a vector in $\{0,1\}^{n+m}$ that falsifies exactly one constraint, say c_i^1, in $\mathcal{R}(\mathcal{A})$. Namely, $\langle x', c_i^1\rangle = 1$ and $\langle x', c'\rangle = 0$ for all $c' \in \mathcal{R}(\mathcal{A}), c' \neq c_i^1$. Let $x = x'_1 \ldots x'_n$. Then $\langle x, c_i\rangle = \langle x', c_i^1 + c_i^2\rangle = \langle x', c_i^1\rangle + \langle x', c_i^2\rangle = 1$, and similarly, $\langle x, c\rangle = 0$ for all $c \in \mathcal{A}, c \neq c_i$. Thus, x falsifies exactly one constraint in $\mathcal{A}$. Since $\mathcal{A}$ is ε-separating, $|x| \geq \varepsilon n$. It follows that $|x'| \geq \varepsilon n$, implying that $\mathcal{R}(\mathcal{A})$ is $\frac{\varepsilon n}{n+m}$-separating. $\square$

Lemma 7.16 *If $\mathcal{A}$ is (q,μ)-local, then $\mathcal{R}(\mathcal{A})$ is (q',μ')-local where $q' = \frac{q}{d'}$ and $\mu' = \mu + \frac{q'}{2m}$.*

Proof. Let $\alpha' \in \{0,1\}^{m+n}$ be the sum of a subset T of $\mu' \cdot 2m$ constraints in $\mathcal{R}(\mathcal{A})$. Let T_2 be the subset of constraints in T that appear in pairs. Namely, for every new variable z, both constrains with z are either in T_2 or not in T_2. Let $T_1 = T \setminus T_2$.

Case 1: $|T_1| \geq q'$. For every constraint in T_1, the new variable z from that constraint does not appear in any other constraint in T. Therefore, α' is 1 on z's coordinate. Hence, $|\alpha'| \geq |T_1| \geq q'$.

Case 2: $|T_1| < q'$. Then $|T_2| = |T| - |T_1| \geq \mu' \cdot 2m - q' = 2\mu m$. Let S be the set of constraints in $\mathcal{A}$ that gave rise to constraints in T_2. Then $|S| = |T_2|/2 \geq \mu m$. Old variables appear in the same number of constraints in S and in T_2. Thus,

$$\left|\sum_{c\in T_2} c\right| \geq \left|\sum_{c\in S} c\right| \geq q.$$

The last inequality follows from the fact that $\mathcal{A}$ is (q,μ)-local. When constraints from T_1 are added to $\sum_{c\in T_2} c$, each T_1 constraint zeroes out at most $\lceil \frac{d}{2} \rceil = d' - 1$ coordinates.

$$|\alpha'| \geq \left|\sum_{c\in T_2} c\right| - \frac{d}{2}\left|\sum_{c\in T_1} c\right| \geq q - (d'-1)q' = q'. \qquad \blacksquare$$

If the reduction is applied $\log d$ times, the number of terms in a constraint drops to 3. To see this, think of applying the reduction i times to a formula with $d \leq 2^i + 2$ terms per constraint. The final theorem of the section summarizes what happens to all relevant parameters. To denote the value of a parameter after i applications of the reduction, we add $^{(i)}$ to its name. Parameter * signifies the final value of the parameter after $\log d$ applications of the reduction.

Theorem 7.17 *Let $\mathcal{V} \subseteq \{0,1\}^n$ be a vector space and let $\mathcal{A}$ be an m-dimensional basis for $\mathcal{V}^\perp$ containing vectors of weight at most d. Let $\mathcal{A}^\star$ be a set of $m^\star$ vectors in $\{0,1\}^{n^\star}$, obtained by applying the reduction $\mathcal{R}$ until the weight of every vector is 3. If $\mathcal{A}$ is ε-separating (q,μ)-local, then $\mathcal{A}^\star$ is $\varepsilon^\star$-separating and $(q^\star,\mu^\star)$-local, where*

$$m^\star = dm \ ; \qquad n^\star = n + (d-1)m \ ;$$
$$\varepsilon^\star = \frac{\varepsilon}{1+(d-1)m/n} \ ; \qquad q^\star \geq \frac{q}{d^{\log d}} \ ;$$
$$\mu^\star \leq \mu + \frac{q}{m} \cdot \frac{\log d}{d} \ .$$

Proof. Since each application of the reduction doubles the dimension, $m^\star = 2^{\log d} m = dm$. To calculate $n^\star$, observe that the reduction does not change $m - n$. Therefore,

$$n^\star = n + m^\star - m = n + (d-1)m.$$

Lemma 7.14 guarantees that $\mathcal{A}^\star$ is independent. By lemma 7.15, $\varepsilon' = \frac{\varepsilon}{1+m/n} = \varepsilon \frac{n}{n'}$. Thus,

$$\varepsilon^\star = \varepsilon \frac{n}{n'} \frac{n'}{n^{(2)}} \cdots \frac{n^{(\log d - 1)}}{n^\star} = \varepsilon \frac{n}{n^\star} = \frac{\varepsilon}{1+(d-1)m/n}.$$

Applying lemma 7.16 $\log d$ times, we obtain

$$\begin{aligned} q^\star &= \frac{q}{d' \cdot d^{(2)} \cdot \dots \cdot d^\star} \geq \frac{q}{d^{\log d}}; \\ \mu^\star &= \mu + \frac{q'}{2m} + \frac{q^{(2)}}{4m} + \dots + \frac{q^\star}{dm} = \mu + \frac{q}{m}\left(\frac{1}{2d'} + \frac{1}{4d^{(2)}} + \dots + \frac{1}{d \cdot d^\star}\right) \leq \mu + \frac{q}{m}\frac{\log d}{d}. \end{aligned}$$

□

Chapter 8

Testing Visual Properties

8.1 Overview

Image analysis is one area potentially well suited to the property testing paradigm. Images contain a large amount of data which must be analyzed quickly in a typical application. Some salient features of an image may be tested by examining only a small part thereof. Indeed, one motivation for this study is the observation that the eye focuses on relatively few places within an image during its analysis. The analogy is not perfect due to the eye's peripheral vision, but it suggests that property testing may give some insight into the visual system.

In this chapter, we present algorithms for a few properties of images. All our algorithms have complexity independent of the image size, and therefore work well even for huge images.

We use an image representation popular in learning theory (see, e.g., [MT89]). Each image is represented by an $n \times n$ matrix M of pixel values. We focus on black and white images given by binary matrices with black denoted by 1 and white denoted by 0. To keep the correspondence with the plane, we index the matrix by $\{0, 1, \ldots, n-1\}^2$, with the lower left corner being $(0,0)$ and the upper left corner being $(0, n-1)$. The object is a subset of $\{0, 1, \ldots, n-1\}^2$ corresponding to black pixels; namely, $\{(i,j) | M_{i,j} = 1\}$.

8.1.1 Property testing in the pixel model

The *distance* between two images of the same size is defined as the number of pixels (matrix entries) on which they differ. (Two matrices of different size are considered to have infinite distance.) The *relative distance* is the ratio of the distance and the number of pixels in the image. A *property* is defined as a collection of pixel matrices. The distance of an image (matrix) M to a property $\mathcal{P}$ is $\min_{M' \in \mathcal{P}} dist(M, M')$. Its *relative distance* to $\mathcal{P}$ is its distance to $\mathcal{P}$ divided by the size of the image matrix. An image is ε-far from $\mathcal{P}$ if its relative distance to $\mathcal{P}$ is at least ε. If the image is not ε-far from $\mathcal{P}$, it is ε-close to it.

A property is (ε, q)-*testable* if there is a randomized algorithm that for every input matrix M queries at most q entries of M and with probability at least $\frac{2}{3}$ distinguishes between matrices with the property and matrices which are ε-far from having it. The algorithm is referred to as an (ε, q)-*test*. This definition allows tests to have *2-sided error*. An algorithm has *1-sided error* if it always accepts an input that has the property.

8.1.2 Our results

We present tests for three visual properties: being a half-plane, convexity and connectedness. The number of queries in all tests is independent of the size of the input. The algorithm for testing if the input is a half-plane is a 1-sided error test with $\frac{2\ln 3}{\varepsilon} + o(\frac{1}{\varepsilon})$ queries. The convexity test has 2-sided error and asks $O(1/\varepsilon^2)$ queries. And finally, the connectedness test has 1-sided error and makes $O\left(\frac{1}{\varepsilon^2}\log^2\frac{1}{\varepsilon}\right)$ queries.

8.1.3 Related results in property testing

Previous papers on property testing in computational geometry [CSZ00, CS01] consider a model different from ours. In their model, the input is the set of object points and each query i to the oracle outputs coordinates of the ith point. Their results, in general, are incomparable to ours. In their model, the problems we consider would have query complexity dependent on the number of points in the object. On the other hand, they are able to study properties which are trivially testable in our model because all instances are either close to having the property or close to not having it. An example is the property that a given graph is a Euclidean minimum spanning tree of a given point set in the plane [CSZ00].

Another related work is [FN01] which studies properties of d-dimensional matrices. It gives a class of properties which are testable with the number of queries polynomial in $1/\varepsilon$. Each d-dimensional grid is viewed as a partially ordered set in the natural way. The main result is that for a fixed d and $0,1$-matrices, every property expressible by a finite collection of forbidden induced posets of the grid has an $(\varepsilon, \text{poly}\,(1/\varepsilon))$-test. It does not seem applicable to our geometric properties.

Goldreich and Ron [GR02] study property testing in bounded degree graphs represented by adjacency lists. Note that an image in the pixel model can be viewed as a graph of degree 4 where vertices correspond to black pixels and they are connected by an edge if the corresponding entries in the image matrix are adjacent. (See the definition of the *image graph* in the beginning of section 8.4.) Goldreich and Ron measure distance between graphs as the ratio of the number of edges that need to be changed to transform one graph into the other over the maximum possible number of edges in the graphs with the given number of vertices and degree. In our case, the distance between two image graphs corresponds to the fraction of points (vertices) on which they differ, i.e. the edge structure of the graphs is fixed, and only vertices can be added or removed to transform one graph into another. Our connectedness test is exactly the same as the connectivity test in [GR02], with one minor variations due to different input representation and the fact that the pixel model allows graphs with a small number of vertices. (In the bounded degree graph model, the number of vertices is a part of the input.) However, since our distance measures are different, their proof of correctness of the algorithm does not apply to the pixel model.

One more paper that studies fast algorithms for connectedness in graphs is [CRT01]. It shows how to approximate the number of connected components in an arbitrary graph in a sublinear time.

8.1.4 Related results in learning

In property testing terminology, a PAC (probably approximately correct) learning algorithm [Val84] is given oracle access (or access via random samples) to an unknown *target* object with the property $\mathcal{P}$ and has to output a *hypothesis* which is within relative distance ε to the target with high probability. If the hypothesis is required to have the property $\mathcal{P}$, the learning algorithm is *proper*. As proved in [GGR98], a proper PAC learning algorithm for $\mathcal{P}$ with sampling complexity $q(\varepsilon)$ implies a (2-sided error) $(\varepsilon, q(\varepsilon/2) + O(1/\varepsilon))$-tester for $\mathcal{P}$.

Learning half-planes exactly is considered in [MT89]. This work gives matching upper and lower bound of $\Theta(\log n)$ for the problem. In the PAC model, a proper learning algorithm with $O(1/\varepsilon \log(1/\varepsilon))$ sampling complexity follows from [BEHW89]. Together with the [GGR98] result above, it implies a (2-sided error) $(\varepsilon, O(1/\varepsilon \log(1/\varepsilon)))$-test for the half-plane property. Our result for testing half-planes is a modest improvement of shaving off the log factor and making the error 1-sided.

The generic approach of [GGR98] for transforming PAC proper learners into property testers does not seem to work well for convexity and connectedness. The complexity of PAC learning algorithms is at least proportional to Vapnik Chervonenkis (VC) dimension[1][EHKV89]. Since VC dimension of convexity is $\Theta(n)$ and VC dimension of connectedness is $\Theta(n^2)$, the corresponding property testers obtained by the generic approach have query complexity guarantee $O(n)$ and $O(n^2)$, respectively. Our testers for these properties have query complexity independent of n.

8.2 Testing if an image is a half-plane

First we present an algorithm for testing if the image is a half-plane. An image is a *half-plane* if there is a vector $w \in \mathbb{R}^2$ such that a pixel x is black if and only if $w^T x \geq 0$. The algorithm first finds a small region within which the dividing line falls. Then it checks if pixels on one side of the region are white and on the other side are black.

Call pixels $(0,0), (0, n-1), (n-1, 0), (n-1, n-1)$ *corners*. Call the first and the last row and the first and the last column of the matrix *sides*. For a pair of pixels p_1, p_2, let $\ell(p_1, p_2)$ denote the line[2] through p_1, p_2. Let $R_1(p_1, p_2)$ and $R_2(p_1, p_2)$ denote the regions into which $\ell(p_1, p_2)$ partitions the image pixels not on the line.

[1]The *VC dimension* is the cardinality of the largest set $X \subseteq \{0, \ldots, n-1\}^2$ shattered by $\mathcal{P}$. A set $X \subseteq \{0, \ldots, n-1\}^2$ is *shattered* by $\mathcal{P}$ if for every partition (X_0, X_1) of X, $\mathcal{P}$ contains a matrix M with $M_x = 1$ for all $x \in X_1$ and $M_x = 0$ for all $x \in X_0$.

[2]Throughout the paper, whenever a geometric notion is used without a definition, it refers to the standard continuous notion. Examples of such notions are line, angle, convex hull. All discretized notions will be defined.

Half-plane test $T_1(\varepsilon)$

Given access to an $n \times n$ pixel matrix,

1. Query the four corners. Let s be the number of sides with differently colored corners.
   (a) If $s = 0$ (all corners are of the same color c), query $\frac{\ln 3}{\varepsilon}$ pixels independently at random. Accept if all of them have color c. Reject otherwise.
   (b) If $s = 2$,
      i. For both sides with differently colored corners, do binary search of pixels on the side to find two differently colored pixels within distance less than $\varepsilon n/2$. For one side, call the white pixel w_1 and the black pixel b_1. Similarly, define w_2 and b_2 for the second side.
      ii. Let $W_i = R_i(w_1, w_2)$ and $B_i = R_i(b_1, b_2)$ for $i = 1, 2$. W.l.o.g., suppose W_2 and B_1 intersect while W_1 and B_2 do not. Query $\frac{2\ln 3}{\varepsilon}$ pixels from $W_1 \cup B_2$ independently at random. Accept if all pixels from W_1 are white, all pixels from B_2 are black. Otherwise, reject.
   (c) If s is not 0 or 2, reject.

Theorem 8.1 *Algorithm T_1 is a 1-sided error $(\varepsilon, \frac{2\ln 3}{\varepsilon} + o(\frac{1}{\varepsilon}))$-test for the half-plane property.*

Proof. The algorithm queries at most $\frac{2\ln 3}{\varepsilon} + O(\log(1/\varepsilon))$ pixels. To prove correctness, we need to show that all half-planes are always accepted, and all images that are ε-far from being half-planes are rejected with probability at least 2/3.

Case (a) [0 differently colored sides]: The image is a half-plane if and only if it is unicolored. If it is unicolored, the test always accepts since it never finds pixels of different colors. If it is ε-far from being a half-plane, it has at least εn^2 pixels of a wrong color. Otherwise, it can be made unicolored, and hence a half-plane, by changing less than an ε-fraction of pixels. The test fails to find an incorrectly colored pixel and accepts with probability at most $(1-\varepsilon)^{\ln 3/\varepsilon} < e^{-\ln 3} = 1/3$.

Case (b) [2 differently colored sides]: The test always accepts all half-planes because it rejects only if it finds two white pixels and two black pixels such that the line through the white pixels intersects the line through the black pixels.

It remains to show that if an image is ε-far from being a half-plane, it is rejected with probability $\geq 2/3$. We prove the contrapositive, namely, that if an image is rejected with probability $< 2/3$, modifying an ε fraction of pixels can change it into a half-plane.

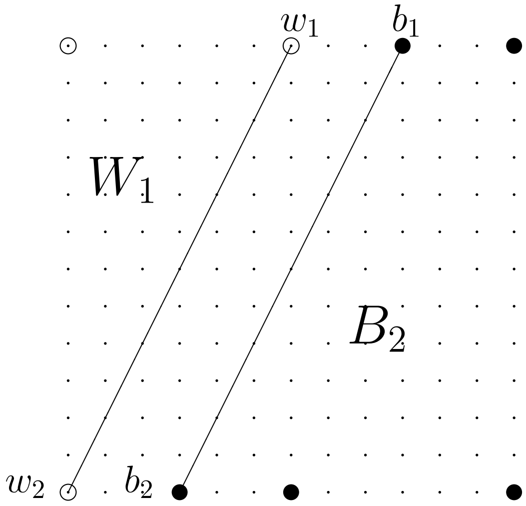


Figure 8-1: Half-plane test

Suppose that an image is accepted with probability $\geq 1/3 = e^{-\ln 3} > (1-\varepsilon/2)^{2\ln 3/\varepsilon}$. That means that $< \varepsilon/2$ fraction of pixels from which we sample in step 1(b)ii differ from the color of their region (white for W_1 and black for B_2). Note also that there are at most $\varepsilon n/2$ pixels outside of $W_1 \cup B_2$. Changing the color of all black pixels in W_1 and all white pixels in B_2 and making all pixels outside of those regions white, creates a half-plane by changing $< \varepsilon$ fraction of the pixels, as required.

Case (c) [everything else]: Note that the number of image sides with differently colored corners is even $(0, 2, \text{ or } 4)$. That holds because the cycle $((0,0),(n-1,0),(n-1,n-1),(0,n-1),(0,0))$ visits a vertex of a different color every time it moves along such a side. So, the only remaining case is 4 differently colored sides. In this case, the image cannot be a half-plane. The test always rejects. $\square$

8.3 Convexity testing

The image is *convex* if the convex hull of black pixels contains only black pixels. The test for convexity first roughly determines the object by querying pixels on the $n/u \times n/u$ grid with a side of size $u = \Theta(\varepsilon n)$. Then it checks if the object corresponds to the rough picture it obtained.

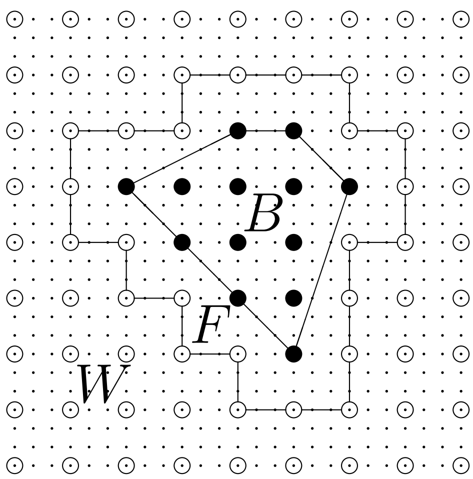


Figure 8-2: Convexity test

For all indices i, j divisible by u, call the set $\{(i',j') |\ i' \in [i, i+u], j' \in [j, j+u]\}$ a *u-square*. We refer to pixels $(i,j),(i+u,j)(i+u,j+u)$, and $(i, j+u)$ as its corners.

CONVEXITY TEST $T_2(\varepsilon)$

Given access to an $n \times n$ pixel matrix,

1. Query all pixels with both coordinates divisible by $u = \lfloor \varepsilon n/120 \rfloor$.

2. Let B be the convex hull of discovered black pixels. Query $\frac{5}{\varepsilon}$ pixels from B independently at random. Reject if a white pixel in B is found in steps 1 or 2.

3. Let W be the union of all u-squares which do not contain any pixels from B. Query $\frac{5}{\varepsilon}$ pixels from W independently at random. Reject if a black pixel is found. Otherwise accept.

Lemma 8.2, used in the analysis of the convexity test, asserts that the number of pixels outside $B \cup W$ is small.

Lemma 8.2 *In an $n \times n$ image, let B be the convex hull of black pixels with coordinates divisible by u. Let W be the union of u-squares which do not contain any pixels from B. Let the "fence" F be the set of pixels not contained in B or W. Then F contains at most $4un$ pixels.*

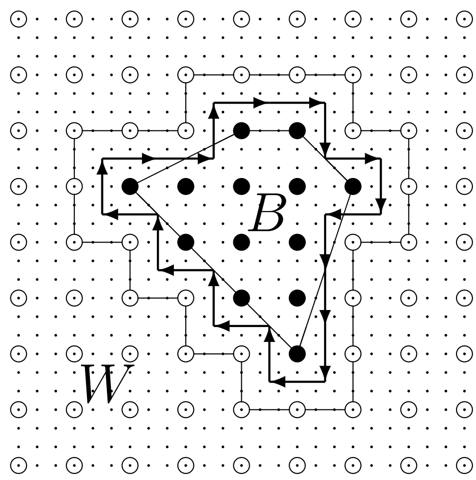


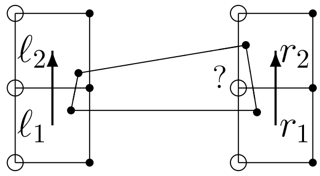
Figure 8-3: Walk over fence u-squares

Proof. Intuitively, F is the largest when it contains all u-squares along the sides of the image.

We call u-squares that are not fully contained in B or W *fence u-squares*. Note that F is covered by *fence u-squares*. Therefore, to prove the lemma it is enough to show that there are at most $4n/u$ *fence u-squares*.

To count the *fence u-squares*, we will define a cyclic ordering on them. To do that, we describe a walk that connects centers of all *fence u-squares*. The walk goes from one center to the next by traveling left, right, up or down. It visits the centers of *fence u-squares* by traveling clockwise and keeping the boundary between F and W on the left-hand side. All *fence u-squares* are visited because each of them intersects with some u-square in W in at least one pixel.

There are n/u rows of u-squares. We claim that from each of these rows the walk can travel up at most once. Suppose for contradiction that it goes up twice, from ℓ_1 to ℓ_2 and from r_1 to r_2, where ℓ_1 and r_1 are *fence u-squares* with centers in row $(k+0.5)u$, and ℓ_2 and r_2 are *fence u-squares* with centers in row $(k+1.5)u$ for some integer k.



W.l.o.g. suppose that the centers of l_1, l_2 are in a column with a lower index than the centers of r_1, r_2. Since the walk keeps the boundary between W and F on the left-hand side, the left corners of ℓ_1, ℓ_2, r_1, r_2 are in W. By definition of *fence u-squares*, ℓ_1, ℓ_2, r_1, r_2 each contain a pixel from B. Then the common left corner of r_1 and r_2 is also in B, since B is convex. But this is a contradiction because W and B are disjoint.

Thus, the walk can travel up only once per row. Similarly, it can travel down only once per row, and travel left (right) only once per column. Since there are n/u rows (columns) of u-squares, the walk can have at most $4n/u$ steps. As it visits all *fence u-squares*, there are at most $4n/u$ of them. Since each u-square contributes u^2 pixels, the number of pixels in F is at most $4nu$. $\square$

The analysis of the convexity test uses the fact that if an image is convex, W contains only a small number of black pixels. Claim 8.3 proves this fact for a special case of an image which is "invisible" on the big grid. Later, we use the claim to handle the general case in lemma 8.4.

Claim 8.3 *In an $n \times n$* convex *image, if all pixels with both coordinates divisible by u are white, then the image contains less than $2un$ black pixels.*

Proof. Let $black(r)$ denote the number of black pixels in a row r. If each row contains fewer than $u-1$ pixels, the total number of black pixels is at most un. Otherwise, consider a row r with $black(r) \geq u$. Let integers k and t be such that $r = ku + t$ and $0 \leq t < u$. Since the image is convex, black pixels of every fixed row must have consecutive column indices. Since every pixel with both coordinates divisible by u is white, $black(ku) < u$ and $black((k+1)u) < u$.

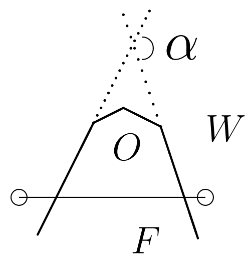
Because of the convexity of the object, if $black(r_1) < black(r)$ for a row $r_1 > r$ then $black(r_2) \leq black(r_1)$ for all rows $r_2 > r_1$. Similarly, if $black(r_1) < black(r)$ for a row $r_1 < r$ then $black(r_2) \leq black(r_1)$ for all rows $r_2 < r_1$. Thus, all rows r_2 excluding

$ku+1, ku+2, \ldots, (k+1)u-1$ have $black(r_2) < u$. Together, they contain $< (n-u)u$ black pixels. Cumulatively, remaining $u-1$ rows contain $< (u-1)n$ pixels. Therefore, the image contains less than $2un$ black pixels. $\square$

Lemma 8.4 *In an $n \times n$* convex *image, let W be the union of all u-squares which do not contain any pixels from B. Then W contains less than $8un$ black pixels.*

Proof. As before, let F be the set of all pixels not contained in B or W. We call pixels on the boundary between F and W with both coordinates divisible by u *fence posts.* Since all *fence posts* are white, any portion of the object protruding into W has to squeeze between the *fence posts.* We show that there are at most three large protruding pieces, each of which, by claim 8.3, contains less than $2un$ pixels. All other sticking out portions fall close to the fence and are covered by the area containing less than $2un$ pixels.

Let O be the boundary of our convex object. O can be viewed as a piecewise linear trajectory on the plane that turns 360°. Whenever O leaves region F to go into W, it has to travel between two *fence posts.* Whenever O comes back into F, it has to return between the same fence posts because the object is convex and *fence posts* do not belong to it. The figuredepicts an excursion of O into W with accumulated turn α.



Notice that since O turns 360° total, at most 3 excursion into W have accumulated turn $> 90°$. Each of them can be viewed as delineating a part of our convex object, cut off by the line between the *fence posts.* This part of the object is convex, and therefore, by claim 8.3, has less than $2un$ pixels. This gives us a total of less than $6un$ pixels for the protruding parts where O turns more than 90°.

Consider any excursion into W where O leaves F between *fence posts* p_1 and p_2 and turns $\leq 90°$ before coming back. Any such trajectory part lies inside the circle of diameter u containing p_1 and p_2. The half of the circle protruding into W is covered by a half of a u-square. By an argument identical to counting *fence squares* in lemma 8.2, there are at most $4n/u$ segments of the F/W boundary between adjacent fence posts. Therefore, the total number of pixels that might be touched by the parts of the object, described by O's excursions into W that turn $\leq 90°$ is at most $4n/u \cdot u^2/2 = 2un$.

Thus the total number of black pixels in W is at less than $8un$. $\square$

Theorem 8.5 *Algorithm T_2 is a $(\varepsilon, O(1/\varepsilon^2))$-test for convexity.*

Proof. The algorithm asks $(n/u)^2 + O(1/\varepsilon) = O(1/\varepsilon^2)$ queries. We prove that the algorithm fails with probability $< 1/3$, considering convex and far from convex images separately.

If the input image is convex, B contains only black pixels. The test never rejects in step 2. By lemma 8.4, the fraction of black pixels in W is $< 8u/n = \varepsilon/15$. By the union bound, the probability that the test rejects in step 3 is $< \frac{\varepsilon}{15}\frac{5}{\varepsilon} = \frac{1}{3}$.

If the input image is ε-far from convex, it has $\geq 2\varepsilon n^2/5$ white pixels in B or $\geq 2\varepsilon n^2/5$ black pixels in W. Otherwise, we could make the image convex by making all pixels in W white and all remaining pixels black. It would require $< 2\varepsilon n^2/5$ changes in B, $< 2\varepsilon n^2/5$ changes in W, and by lemma 8.2, $\leq 4un < \varepsilon n^2/5$ changes in F. Thus, the distance of the image to convex would be less than εn^2.

Suppose w.l.o.g. that there are $\geq 2\varepsilon/5$ black pixels in W. Step 3 will fail to find a black pixel with probability $\leq (1 - \frac{2\varepsilon}{5})^{5/\varepsilon} \leq e^{-2} < \frac{1}{3}$. $\square$

8.4 Connectedness testing

Define the *image graph* $G_M = (V, E)$ of image matrix M by $V = \{(i,j) | M_{i,j} = 1\}$ and $E = \{((i_1, j), (i_2, j)) | \; |i_1 - i_2| = 1\} \cup \{((i, j_1), (i, j_2)) | \; |j_1 - j_2| = 1\}$. In other words, the image graph consists of black pixels connected by the grid lines. The image is *connected* if its image graph is connected. When we say that the image has k connected components, we are also referring to its image graph.

The test for connectedness looks for isolated components of size less than $d = 4/\varepsilon^2$. Before presenting the test, we prove that a significant fraction of pixels are in such components if the image is far from connected. When a small isolated component is discovered, the test rejects if it finds a black pixel outside of the component. Lemma 8.6 implies that for an image to be far from connected, it has to have a large number of connected components. Then an averaging argument in lemma 8.7 demonstrates that many of them have to be small. This gives rise to a simple algorithm T_3, which is later improved to algorithm T_4 with more careful accounting in claim 8.9.

Both algorithms for connectedness and claim 8.9 are adopted from [GR02]. The only change in the algorithms, besides parameters, is that after finding a small component, we have to make sure there is some point outside of it before claiming that the image is far from connected.

Lemma 8.6 *If an $n \times n$ image contains at most p connected components, they can be linked into one connected component by changing at most $n(\sqrt{2p} + O(1))$ pixel values from white to black.*

Proof. Let $s = n\sqrt{2/p}$. To turn the image into one connected component, we first add the comb-like set $S = \{(i, j) |\ j = n - 1 \text{ or } i = n - 1 \text{ or } s \text{ divides } i\}$. Now every connected component is linked to S by adding at most $s/2$ pixels leading to the nearest "tooth of the comb". That is, if a component contains a pixel $(ks+\ell, j)$ for an integer k and $0 \leq \ell \leq s/2$, add pixels $(ks+1, j), (ks+2, j), \ldots, (ks+\ell-1, j)$. Otherwise (a component contains a pixel $(ks + \ell, j)$ for integer k and $s/2 < \ell < s$), add pixels $(ks + \ell + 1, j), (ks + \ell + 2, j), \ldots, (ks + s - 1, j)$.

The first stage adds $|S| = n(n/s + O(1))$ pixels and the second, less than $s/2$ per connected component, adding the total of $n(n/s+O(1))+ps/2 = n\sqrt{2p}+O(1)$ pixels. □

Lemma 8.7 *If an image is ε-far from connected, at least an $\frac{\varepsilon^2}{4} - o(1)$ fraction of its pixels are in connected components of size less than $d = 4/\varepsilon^2 + o(1)$.*

Proof. Consider an $n \times n$ ε-far from connected image with p connected components. By lemma 8.6, changing at most $n(\sqrt{2p} + O(1))$ pixels makes it connected. Therefore,

$$n(\sqrt{2p} + O(1)) \geq \varepsilon n^2,$$
$$p \geq \varepsilon^2 n^2/2 - O(n).$$

Let b be the number of black pixels. The average component size is

$$\frac{b}{p} \leq \frac{n^2}{\varepsilon^2 n^2/2 - O(n)} = \frac{2}{\varepsilon^2} + o(1).$$

Thus, the fraction of components of size up to $d = \frac{4}{\varepsilon^2} + o(1)$ is at least $1/2$. That is, there are at least $p/2 = \varepsilon^2 n^2/4 - O(n)$ such components. Since each connected component contains a pixel, at least $\varepsilon^2/4 - o(1)$ fraction of pixels are in connected components of size d. □

Connectedness test $T_3(\varepsilon)$

Let $\delta = \frac{\varepsilon^2}{4} - o(1)$ and $d = 4/\varepsilon^2$. Given access to an $n \times n$ pixel matrix,

1. Query $2/\delta$ pixels independently at random.
2. For every pixel (i, j) queried in step 1, perform a breadth first search (BFS) of the image graph starting from (i, j) until d black pixels are discovered or no more new black pixels can be reached; i.e., for each discovered black pixel query all its neighbors if they haven't been queried yet. If no more new black pixels can be reached, a small connected component has been found.
3. If a small connected component is discovered for some (i, j) in step 2, query $2/\varepsilon$ pixels outside of the square $[i - d, i + d] \times [j - d, j + d]$ independently at random. If a black pixel is discovered, reject. Otherwise (if no small connected component is found or if no black pixel is discovered outside of the small component), accept.

Theorem 8.8 *Algorithm T_3 is a 1-sided $(\varepsilon, O(\varepsilon^{-4}))$-test for connectedness.*

Proof. The algorithm accepts all connected images because it rejects only if an isolated component and some pixel outside of it are found.

It remains to show that an ε-far from connected image is rejected with probability at least 2/3. By lemma 8.7, such an image has at least a δ fraction of its pixels in connected components of size less than d. The probability that step 1 fails to find a pixel from a small connected component is $(1 - \delta)^{2/\delta} \leq e^{-2}$. In step 2, $3d - 1$ queries are sufficient to discover that a component of size $d - 1$ is isolated because it has at most $2d$ neighboring white pixels. There are at least $\varepsilon n^2 - 4d^2$ black pixels outside of the $2d \times 2d$ square containing the small isolated component. Step 3 will fail to find a black pixel with probability $(1 - \varepsilon)^{2\varepsilon} \leq e^{-2}$. By the union bound, the failure probability is at most $2/e^2 < 1/3$.

The number of queries is at most $2/\delta \times (3d - 1) + 2/\varepsilon = O(\varepsilon^{-4})$. □

The algorithm can be improved by employing Goldreich–Ron trick [GR02] of considering small components of different sizes separately. The following claim is adopted from [GR02].

Claim 8.9 *If an image has at least C connected components of size less than d, there is $\ell \leq \log d$ such that at least $\frac{C \cdot 2^{\ell-1}}{\log d}$ points are in connected components of size between $2^{\ell-1}$ and $2^\ell - 1$.*

Proof. For some $\ell \le \log d$, the image has at least $C/\log d$ connected components of size between $2^{\ell-1}$ and $2^{\ell}-1$. Each of them contains at least $2^{\ell-1}$ points. $\square$

(IMPROVED) CONNECTEDNESS TEST $T_4(\varepsilon)$
Let $\delta = \frac{\varepsilon^2}{4} - o(1)$ and $d = 4/\varepsilon^2$. Given access to an $n \times n$ pixel matrix, 1. For $\ell = 1$ to $\log d$ (a) Query $\frac{4 \log d}{\delta 2^{\ell}}$ pixels independently at random. (b) For every pixel (i,j) queried in step 1a, perform a BFS of the image graph starting from (i,j) until 2^{ℓ} black pixels are discovered or no more new black pixels can be reached (a small connected component has been found). 2. If a small connected component is discovered for some (i,j) in step 1, query $2/\varepsilon$ pixels outside of the square $[i-d, i+d] \times [j-d, j+d]$ independently at random. If a black pixel is discovered, reject. Otherwise (if no small connected component is found or if no black pixel is discovered outside of the small component), accept.

Theorem 8.10 *Algorithm T_4 is a 1-sided $\left(\varepsilon, O\left(\frac{1}{\varepsilon^2}\log^2\frac{1}{\varepsilon}\right)\right)$-test for connectedness.*

Proof. The algorithm accepts all connected images because it rejects only if an isolated component and some pixel outside of it are found.

If an $n \times n$ image is ε-far from connected, by the proof of lemma 8.7, it has at least a δn^2 connected components of size less than d. Claim 8.9 implies that for some $\ell < \log d$, at least an $\frac{\delta \cdot 2^{\ell-1}}{\log d}$ fraction of its points are in connected components of size between $2^{\ell-1}$ and $2^{\ell}-1$. For this ℓ, the probability that step 1 fails to find a pixel from a component of size between $2^{\ell-1}$ and $2^{\ell}-1$ is at most e^{-2}. The rest of the correctness analysis is the same as in theorem 8.8.

The number of queries is at most $\log d \cdot O\left(\frac{\log d}{\delta}\right) + 2/\varepsilon = O\left(\frac{1}{\varepsilon^2}\log^2\frac{1}{\varepsilon}\right)$. $\square$

8.5 Conclusion and open problems

Employing the paradigm from the half-space test

The strategy employed in the half-plane test of section 8.2 is very simple. First we approximately learn the position of the dividing line. Then, using the fact that all half-planes consistent with our knowledge of the dividing line can differ only on a fixed $\varepsilon/2$ fraction of the pixels, we randomly check if the matrix corresponds to these half-planes on the remaining pixels.

This suggests a general paradigm for transforming PAC learning algorithms into property testers with *1-sided error*. Namely, consider a property $\mathcal{P}$ where all objects with $\mathcal{P}$ which are $\varepsilon/2$-close to a given object are the same on all but $\varepsilon/2$ fraction of the points. In addition, assume there is a proper PAC learning algorithm with sampling complexity $q(n, \varepsilon)$. Then the following test for $\mathcal{P}$ has 1-sided error and query complexity $q(n, \varepsilon/2) + O(1/\varepsilon)$: learn the property within relative error of $\varepsilon/2$ and then randomly test the object on points where all objects $\varepsilon/2$-close the hypothesis coincide. The proof of this fact is very similar to the case 2 of the analysis of the half-plane test.

Extension to d dimensions and lower bounds

A straightforward extension of our tests to d dimensions seems to give tests with dependence on d, and thus dependent on the size of the image. It would be interesting to investigate if this dependence is necessary.

It is known that testing some properties requires the number of queries linear in the size of the input [BOT02, BHR03]. However, known hard properties do not seem to have a natural geometric interpretation. It would be nice to find natural 2-dimensional visual properties which are hard to test. One result follows directly from [BEK$^+$03] which shows that testing whether a string of length n is a shift of another string requires $\Omega(n^{1/2})$ queries. This implies that testing whether the lower half of an $n \times n$ image is a shift of the upper half requires $\Omega(n^{1/2})$ queries. It would be interesting to find even harder visual properties.

Extension to non-binary matrices

We restricted our attention to images representable by binary matrices. However, in real life images have many colors (or intensity values). Property testers for images represented by integer-valued matrices would be a natural generalization. For example, one can generalize convexity in the following way. Call an image represented by an $n \times n$ matrix with values in R *convex* if the corresponding function $\{0, 1, \ldots, n-1\}^2 \to R$ is convex.

Chapter 9

Conclusion and Open Problems

This thesis studies algorithms and lower bounds for various problems in the property testing model that allows us to investigate approximate sublinear computation. In addition, it examines the issue of identifying *classes* of problems that are amenable to similar algorithmic techniques.

Characterizing testable properties. The first topic we touched is characterizing general properties according to their query complexity. A series of previous works identified classes of properties testable with constant query complexity. This thesis makes an attempt to characterize properties over the binary alphabet with sublinear but not necessarily constant query complexity.

A few open questions emerge from this work. We prove there exists a class of non-testable 3CNF properties by a random construction. As a result, the class of non-testable 3CNF properties we present is non-uniform. The question of finding a uniform class of 3CNF properties is still open. One possibility for answering it is "derandomizing" our construction. This requires replacing random graphs in our construction with explicit expanders, and checking if they satisfy our conditions. The only known construction of expanders that might be good enough for our application is given by [CRVW02]. It is quite complicated, and so far it is not clear if it satisfies our conditions.

More generally, it would be interesting to find other classes of properties testable with sublinear complexity and to extend our results to properties over arbitrary alphabets.

Locally checkable codes. Another question concerns an intermediate result from Chapter 7. We prove that codes obtained from random (c, d)-regular graphs (for sufficiently large constants c and d) are not testable with sublinear query complexity. This gives families of codes with a linear distance and constant rate which are not testable or, in the coding theory terminology, not *checkable* with sublinear query complexity. Are there any codes with a linear distance and constant rate which are checkable with sublinear complexity? So far, it is not clear what the answer is. But hopefully, the techniques used for proving 3CNF lower bound will yield some lower bound on testing general codes with linear distance and constant rate. Finding a family of codes with linear distance and constant rate that are checkable with a small number of queries would be a big breakthrough.

Adaptivity. A property tester that asks all its queries before getting any answers is called *non-adaptive*. Obtaining lower bounds for general property testers proved much more difficult than for non-adaptive case. There are many examples of properties where the gap between the known non-adaptive and adaptive lower bounds is exponential, even though the best-known test for the problem is non-adaptive. Examples of such properties include monotonicity on the Boolean hypercube and monotonicity on general graphs.

Only two techniques are known for obtaining adaptive lower bounds. The first is to take a logarithm of the non-adaptive lower bound. This works for all properties because every adaptive test can be simulated by a non-adaptive test that asks queries for all possible answers that the adaptive test might get. The obvious disadvantage of this technique is that it gives very weak adaptive bounds when adaptivity does not help significantly. The second method described in Chapter 6 avoids this shortcoming, but applies only to linear properties. One broad research project is to continue to investigate when adaptivity helps and try to develop methods for proving adaptive lower bounds for other classes of properties that perform better than the generic method.

Extending the property testing model. Stepping a little bit outside of the traditional property testing model, one can ask for algorithms which would accept inputs with a small number of mistakes. We might require that the input is accepted with high probability if *at most* an ε_1 fraction of the input characters has to be modified to make it satisfy the property and rejected with high probability if *at least* an ε_2 fraction of the input needs modifications.

This “double threshold” model could be more useful for some applications. In a recent paper [BEK$^+$03], we applied this model to testing edit distance between two strings. It can be applied to many problems previously considered in the traditional property testing model; e.g., testing various graph and function properties would be very interesting. Unfortunately, in most cases, property testing algorithms do not work in the double threshold model. Typically they reject as soon as they find an error, and there are some inputs with a single error which the tests are certain or likely to notice.

A more general question is which pairs of disjoint sets can be distinguished by sublinear algorithms. Say that two sets are *distinguishable* if there is a sublinear-query test that given an input from one of them, determines with high probability which set the input came from. Notice that two distinguishable sets do not necessarily have a large Hamming distance. For example, the set of strings that start with 0 and the set of strings that start with 1 are distinguishable with one query. Are 3-colorable graphs distinguishable from graphs which are not 4-colorable? Which other interesting sets are distinguishable?

Pixel model for geometric problems Another broad research project to continue to investigate testing of visual properties of images, initiated in Chapter 8. It would be nice to find sublinear algorithms for more visual properties, and learn for which properties it is impossible to do. Another interesting direction is extending the model to more than 2 dimensions.

Bibliography

[AFKS00] Noga Alon, Eldar Fischer, Michael Krivelevich and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.

[AKNS01] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal of Computing*, 30(6):1842–1862, 2001.

[AS02] Noga Alon and Asaf Shapira. Testing Satisfiability. In *Proceedings of 13th SODA*, 645–654, 2002.

[AS00] Noga Alon and Joel H. Spencer. *The probabilistic method.* Second edition, Wiley press, New York, 2000.

[BEK$^+$03] Tuğkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A Sublinear Algorithm for Weakly Approximating Edit Distance. To appear in *Proceedings of the 35th ACM STOC*, 2003.

[BRW99] Tuğkan Batu, Ronitt Rubinfeld and Patrick White. Fast approximation PCPs for multidimensional bin-packing problems. In *Proceedings of the* 3^{rd} *International Workshop on Randomization and Approximation Techniques in Computer Science*, 246–256, 1999.

[Beh46] F. A. Behrend. On sets of integers which contain no three terms in arithmetic progression. In *Proceedings of the National Academy of Sciences of the United States of America* 32, 331–332, 1946.

[BHR03] Eli Ben-Sasson, Prahladh Harsha, and Sofya Raskhodnikova, Some 3CNF properties are hard to test. To appear in *Proceedings of the 35th ACM STOC*, 2003.

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. To appear in *Proceedings of the 35th ACM STOC*, 2003.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for computing machinery* 36(4):929–965, 1989.

[BOT02] Andrej Bogdanov, Kenji Obata, and Luca Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *Proceedings of the* 43*rd IEEE Symp. on Foundations of Comp. Science*, pages 93–102, Vancouver, Canada, 16–19 Nov. 2002.

[CRVW02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness Conductors and Constant-Degree Lossless Expanders. STOC-CCC 2002.

[CRT01] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. In *Proceedings ICALP*, 2001.

[CS88] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, Oct. 1988.

[CS01] Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *Proceedings of the 9th Annual European Symposium on Algorithms*, 266–277, 2001.

[CSZ00] Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In *Proceedings of the 8th Annual European Symposium on Algorithms*, 155–166, 2000.

[Dil50] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Math* (2) 51, 161–166, 1950.

[DGL$^+$99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of the* 3^{rd} *International Workshop on Randomization and Approximation Techniques in Computer Science*, 97–108, 1999.

[Dur96] Rick Durrett. *Probability: theory and examples.* Second edition, Duxbury Press, Belmont, 1996.

[EHKV89] Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A General Lower Bound on the Number of Examples Needed for Learning. *Information and Computation* 82(3), 247–261, 1989.

[EKK$^+$98] Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot checkers. *Journal of Computer and System Science* 60, 717–751, 2000. (A preliminary version appeared in Proceedings of the 30^{th} STOC, 1998.)

[ESY84] S. Even, A. Selman and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control,* 61(2):159–173, May 1984.

[Fis] Eldar Fischer. *On the strength of comparisons in property testing.* Manuscript (available as ECCC TR00-083).

[Fis01] Eldar Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, Oct. 2001. The Computational Complexity Column.

[Fis01a] Eldar Fischer. Testing graphs for colorability properties. In *Proceedings of the* 12*th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 873–882, New York, 7–9 Jan. 2001.

[FLN$^+$02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the* 34*th ACM Symp. on Theory of Computing*, pages 474–483, New York, 19–21 May 2002.

[FN02] Eldar Fischer and Ilan Newman. Functions that have read-twice, constant width, branching programs are not necessarily testable. In *Proceedings of the* 17*th Conference on Computational Complexity*, pages 73–77, Montréal, Québec, Canada, 21–24 May 2002.

[FN01] Eldar Fischer and Ilan Newman. Testing of matrix properties. In *Proceedings of the* 33^{rd} *ACM STOC*, 286–295, 2001.

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes.* MIT Press, Cambridge, MA, 1963.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998. (A preliminary version appeared in Proceedings of the 37^{th} FOCS, 1996.)

[GGL+00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica* 20, 301–337, 2000. (A preliminary version written by the first four authors appeared in *Proceedings of the* 39^{th} *IEEE FOCS*, 426–435, 1998.)

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. (A preliminary version appeared in *Proceedings of the* 29^{th} *ACM STOC*, 406–415, 1997.)

[GS02] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost linear length. In *Proceedings of the* 43*rd IEEE Symp. on Foundations of Comp. Science*, pages 13–22, Vancouver, Canada, 16–19 Nov. 2002.

[HW01] Johan Håstad and Avi Wigderson. Simple Analysis of Graph Tests for Linearity and PCP. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, 2001.

[Lev71] V. I. Levenšteĭn. Upper bounds for codes with a fixed weight of vectors (in Russian). Проблемы передачи информации 7, 3–12, 1971.

[LT79] R. J. Lipton, and R. E. Tarjan. A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics* 36, 177-189, 1979.

[MT89] Wolfgang Maass and György Turan. On the complexity of learning from counterexamples. In *Proceedings of the* 30^{th} *IEEE FOCS*, 262–267, 1989.

[MP69] Marvin Lee Minsky and Seymour Papert. *Perceptrons; an introduction to computational geometry.* MIT Press, 1969.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, 1995.

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

[Ras] Sofya Raskhodnikova. *Approximate testing of visual properties.* Manuscript.

[Ras99] Sofya Raskhodnikova. *Monotonicity testing.* Master's Thesis, Massachusetts Institute of Technology, June 1999.

[Ron01] Dana Ron. Property testing (a tutorial). In S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. Rolim, editors, *Handbook of Randomized Computing*, volume 9 of *Combinatorial Optimization*, pages 597–649. Kluwer Academic Publishers, 2001.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing*, 25(2):252–271, Apr. 1996. (First appeared as a technical report, Cornell University, 1993).

[RS78] I. Z. Ruzsá and E. Szemerédi. Triple systems with no six points carrying three triangles. Colloquia Mathematica Societatis János Bolyai 18, 939-945, 1978.

[Spi95] Dan Spielman. *Computationally Efficient Error-Correcting Codes and Holographic Proofs.* PhD thesis, Massachusetts Institute of Technology, June 1995.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM* 27, 1134–1142, 1984.