Topics in Property Testing

Shirley Halevy
Topics in Property Testing

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Shirley Halevy

Submitted to the Senate of
the Technion — Israel Institute of Technology
TISHRAY 5766 Haifa October 2005
I would like to thank Eyal for his endless guidance, devotion and encouragement, and above all for his support and sense of humor.

I will like to thank my wonderful family that accompanies me in every road I take, and unconditionally supports me in every decision I make. Especially, I like to thank my husband Yuval, my partner for the journey of life, for hearing me all the time and sometimes even listening.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

Abstract 1
Notation 3

1 Introduction 5
   1.1 Previous Work 6
   1.2 Our Contributions 9
   1.3 Consecutive work 12
   1.4 Organization: 12

2 Definitions 13

3 Distribution-free Low-Degree Testers for Polynomials 15
   3.1 Preliminaries 15
   3.2 Low-degree test for the uniform distribution 16
   3.3 Distribution-free low-degree testing algorithm 16

4 Distribution-Free Testing of Properties with Self-Correctors 19

5 Distribution-Free Connectivity Testing for Sparse Graphs 23
   5.1 Connectivity testing of bounded-degree graphs 23
   5.1.1 Testing connectivity for graphs with bounded-degree \(d \geq 3\) 25
   5.1.2 Testing connectivity for graphs with degree \(d = 2\) 27
   5.1.3 Proof of Lemma 5.1.9: 30
   5.2 Alternative testing models for sparse graphs 33
   5.2.1 The edge-bounded model 33
   5.2.2 The combined model 34

6 Distribution-Free Monotonicity Testing on the \(d\)-Dimensional Cube 41
   6.1 Preliminaries 41
   6.2 Testing monotonicity for the line \((d = 1)\) 41
   6.3 Distribution-free monotonicity testing for general \(d\) 44

7 A Lower Bound for Distribution-Free Monotonicity Testing in the High-Dimensional Case 47
   7.1 The Two Families 47
      7.1.1 The family \(\mathcal{F}_1\) 49
      7.1.2 The family \(\mathcal{F}_2\) 50
   7.2 A lower bound for one-sided error testing 51
7.3 A lower Bound for Two-Sided Error Testing .................................................. 56

8 Testing Monotonicity over Graph Products ....................................................... 61
  8.1 Preliminaries and general approach ............................................................. 61
  8.2 Testing boolean functions over graph products ............................................ 64
  8.3 General functions defined over products of the line .................................... 68
     8.3.1 Monotonicity testing for functions defined over $[n]^2$ ..................... 69
     8.3.2 Monotonicity testing over products of the line .................................. 71
     8.3.3 Testing monotonicity over $[n]^d$ ...................................................... 72

9 Open problems .................................................................................................. 79

Bibliography ......................................................................................................... 80
List of Figures

1.1 An example for a product of two graphs $G_1 \times G_2$ .......................... 8
3.1 Distribution-free low degree tester ...................................................... 17
4.1 Generalized distribution-free testing scheme ......................................... 20
5.1 Distribution-free tester for $d \geq 3$ ..................................................... 25
5.2 Distribution-free tester for $d = 2$ ..................................................... 28
5.3 Distribution-free tester for $\mathcal{P}_m^d$ ............................................... 37
6.1 Distribution-free monotonicity tester for $d = 1$ .................................... 42
6.2 Distribution-free monotonicity tester for general $d$ .............................. 44
7.1 An example for a function in $\mathcal{F}_1$ for $m = 2$ ................................. 49
7.2 An example for a function in $\mathcal{F}_2$ for $m = 2$ ................................. 50
8.1 $b$-copy of $G_1$ in the product $G_1 \times G_2$ of Figure 1.1 ....................... 62
8.2 The left function, $f_1$, is $G_1$-monotone, while the right function, $f_2$, is $G_2$-monotone. None of $f_1, f_2$ is $G_1 \times G_2$ monotone. ............... 62
8.3 General testing scheme for graph products ........................................... 63
8.4 An example of a $G_1$-monotone function defined over $[n] \times [n]$ and a possible transformation of it to a $G_2$-monotone function .................. 64
8.5 Two functions $f_1$ and $f_2$ over $\{0,1\}^d$ and the set $X$ ....................... 65
8.6 The function $f$ ......................................................................................... 66
8.7 Areas $S, B, R$ and $L$. ........................................................................... 66
8.8 The points $(i_c, j_c)$ and $(i_r, j_r)$ are denoted $a$ and $b$, respectively. ........ 67
8.9 Example for a set $X$, as in the proof of Lemma 8.3.5, that does not cover all the diagonals violations (the points in $X$ are surrounded by circles). ... 71
8.10 Linear monotonicity edge test for the line ............................................ 74
8.11 Testing scheme for linear edge tests ....................................................... 74
8.12 Testing scheme over graph powers .......................................................... 75
Abstract

The classical notion of decision problems requires an algorithm to distinguish objects having some property P from those objects which do not have the property. Property testing is a relaxation of decision problems, where algorithms are only required to distinguish objects having the property P from those which are “far” from every such object. We consider the problem of distribution-free property-testing of functions. In this setting of property testing, the distance between functions is measured with respect to a fixed but unknown distribution $D$ on the domain, and the testing algorithms have an oracle access to random samples from the domain according to this distribution $D$. This notion of distribution-free testing was previously defined, but no distribution-free property-testing algorithm was known for any (non-trivial) property. By extending known results (from “standard”, uniform distribution, property-testing), we present the first such distribution-free algorithms for some of the central problems in this field:

- A distribution-free testing algorithm for low-degree multivariate polynomials with query complexity $O(d^2 + d \cdot \epsilon^{-1})$, where $d$ is the total degree of the polynomial. The same approach that is taken for the distribution-free testing of low-degree polynomials is shown to apply also to several other problems.

- We consider three models for testing properties of sparse graphs in the distribution-free setting. These models are generalizations of the ones presented previously for sparse graphs in the context of uniform distribution testing. We show that for each of these models, connectivity can be tested in a distribution-free manner, using a number of queries that is independent of the size of the graph.

- A distribution-free monotonicity testing algorithm for functions $f : [n]^d \rightarrow A$ for low dimensions (e.g., when $d$ is a constant) with query complexity similar to the one achieved in the uniform setting.

On the other hand, we show that, though in the uniform setting testing of boolean functions defined over the boolean hypercube can be done using query complexity that is polynomial in $\frac{1}{\epsilon}$ and in the dimension $d$, in the distribution-free setting such testing requires a number of queries that is exponential in the dimension $d$. By this we show an inherent exponential gap in the query complexity between the uniform and the distribution-free settings for a natural testing problem, for which the testing is possible in both settings using non-trivial query complexity.

As for the uniform setting, we consider a new testing approach for testing monotonicity of functions defined over graph products (this approach has been used before for limited classes of functions). We show that it can be applied whenever the functions in question are boolean and whenever one of the graphs in the product is the line. We then use our results to improve the best known query complexity for testing monotonicity of functions defined over the hypercube in the low-dimensional case.
Notation

\( \mathcal{P}^d \) — The class of multivariate polynomials of total degree \( d \)

\( \mathcal{P}_{deg} \) — The class of functions that represent connected graphs with bounded-degree \( d \)

\( \mathcal{P}_m \) — The class of functions that represent connected graphs with at most \( m \) edges

\( \mathcal{P}^d_m \) — The class of functions that represent bounded-degree \( d \) connected graph with at most \( m \) edges

\( n_C \) — The number of vertices in a connected component \( C \)

\( n_{\chi(C)} \) — The number of free pairs \((v, i)\) such that \( v \in C \)

\([n]\) — The set \( \{1, \ldots, n\} \)

\([n]^d\) — The set of \( d \)-tuples over \( n \)

\( \mathcal{A}_f \) — The active set of \( f \)

\( p||p' \) — The concatenation of \( p \) and \( p' \)

\( H(x, y) \) — The hamming distance between \( x \) and \( y \)

\( B^d_\lambda \) — The set of points in \( \{0,1\}^d \) of weight \( \lambda d \)

\( S_f \) — The knowledge sequence learnt by \( A \) during its execution on \((f, D_f)\)

\( S_f(i) \) — The length \( i \) prefix of \( S_f \)

\( \mathcal{P}^A_1 \) — The distribution induced over length \( n \) knowledge sequences

by running \( A \) on a randomly drawn pair \((f, D_f)\) from \( \mathcal{F}_1 \)

\( \mathcal{P}^A_2 \) — The distribution induced over length \( n \) knowledge sequences

by running \( A \) on a randomly drawn pair \((f, D_f)\) from \( \mathcal{F}_2 \)

\( \mathcal{P}_{mono}(G, A) \) — The class of functions \( f : V \to A \) that are monotone with respect to \( G \)

\( v \times G_2 \) — The subgraph of \( G_1 \times G_2 \) induced by \( \{(v, u) : u \in V_2\} \)

\( f_v \) — The function induced by \( f \) on \( v \times G_2 \)

\( P_T(\epsilon) \) — The detection probability of the tester \( T \)

\( \epsilon_{1D}(f) \) — The expected distance of the one-dimensional functions induced by \( f \)
Chapter 1

Introduction

The classical notion of decision problems requires an algorithm to distinguish objects having some property $\mathcal{P}$ from those objects which do not have the property. Property testing is a relaxation of decision problems, where algorithms are only required to distinguish objects having the property $\mathcal{P}$ from those which are at least $\epsilon$-far from every such object. The notion of property testing was introduced by Rubinfeld and Sudan [48] and since then attracted a considerable amount of attention. Property testing algorithms were introduced for problems in graph theory (e.g. [3, 28, 31, 43]), monotonicity testing (e.g. [15, 19, 20, 25, 27]) and other properties (e.g. [2, 6, 10, 13, 16, 18, 21, 23, 29, 39, 41, 42, 44, 45, 47]) (the reader is referred to surveys by Ron [46], Goldreich [26], and Fischer [22] for a presentation of some of this work, including some connections between property testing and other topics). The main goal of property testing algorithms is to avoid “reading” the whole object (which requires complexity at least linear in the size of its representation); i.e., to make the decision by reading a small (possibly, selected at random) fraction of the input (e.g., a fraction of size polynomial in $1/\epsilon$ and poly-logarithmic in the size of the representation) and still having a good (say, at least $2/3$) probability of success.

A crucial component in the definition of property testing is that of the distance between two objects. For the purpose of this definition, it is common to think of objects as being functions over some domain $\mathcal{X}$. For example, a graph $G$ may be thought of as a function $f_G : V \times V \to \{0, 1\}$ indicating for each possible edge $e$ whether it exists in the graph. The distance between functions $f$ and $g$ is then measured by considering the set $\mathcal{X}_{f\neq g}$ of all points $x$ where $f(x) \neq g(x)$ and comparing the size of this set $\mathcal{X}_{f\neq g}$ to that of $\mathcal{X}$; equivalently, one may introduce a uniform distribution over $\mathcal{X}$ and measure the probability of picking $x \in \mathcal{X}_{f\neq g}$. Note that property-testing algorithms access the input function (object) via membership queries (i.e., the algorithm gives a value $x$ and gets $f(x)$).

Our study of property testing goes in two different directions. The first direction is generalizing the testing model to deal with arbitrary distribution measures, while the second direction is looking for general techniques for constructing testers. We elaborate on both directions below.

Distribution-free testing: It is natural to generalize the definition of distance between two functions that was presented above, to deal with arbitrary probability distributions $D$ over $\mathcal{X}$, by measuring the probability of $\mathcal{X}_{f\neq g}$ according to $D$. Ideally, one would hope to get distribution-free property-testing algorithms. A distribution-free tester for a given property $\mathcal{P}$ accesses the function using membership queries, as above, and by randomly sampling the fixed but unknown distribution $D$ (this mimics similar definitions from learning theory and...
is implemented via an oracle access to $D$; see, e.g., [40]. As before, the testing algorithm is required to accept the given function $f$ with probability of at least $\frac{2}{3}$ if $f$ satisfies the property $\mathcal{P}$, and to reject it with probability of at least $\frac{2}{3}$ if $f$ is at least $\epsilon$-far from $\mathcal{P}$ with respect to the distribution $D$.

Indeed, this definition of distance with respect to an arbitrary distribution $D$ was already considered in the context of property testing [28]. However, prior to our work, no distribution-free property-testing algorithm was known for any (non-trivial) property (besides testing algorithms that follow the existence of proper learning algorithms in learning-theory [28]). Moreover, discouraging impossibility results, due to [28], show that for many graph-theoretic properties (for which testers that work with respect to the uniform distribution are known) no such distribution-free algorithm exists. As a result, most previous work focused on testing algorithms for the uniform distribution; some of these algorithms can be generalized to deal with certain (quite limited) classes of distributions (e.g., product distributions [28]), and very few can be modified to be testers with respect to any known distribution (as was observed by [22] regarding the tester presented in [41]), but none is shown to be a distribution-free tester.

A natural question that arises when dealing with distribution-free testing, beyond the mere existence of such testers, is whether the testing can be done using the same query complexity as in the uniform setting. That is, whether whenever distribution-free testing is possible using non-trivial query complexity, it can be done using the same query complexity as in the uniform setting (up to a constant factor), or whether there exist problems for which there is a gap between the query complexity required for the different settings.

**General techniques for constructing testers:** A common approach used in the construction of algorithms in other fields of computer science theory is modular design. That is, using algorithms that were designed to solve specific problems, as black boxes in the solution of other problems. It is interesting to know whether such an approach can be applied also in property testing. That is, can we use known testers for specific properties as black boxes for the testing of other properties? Specifically, we study the possibility to use known monotonicity testers for given graphs to test monotonicity of functions defined over their product.

### 1.1 Previous Work

We start by reviewing some of the central problems studied in the context of property testing, which are relevant to the current work.

**Low-degree tests for polynomials.** The first problem studied in the field of property testing is that of low-degree testing for multivariate polynomials over a finite field, where one wishes to test whether a given function can be represented by a multivariate polynomial of total degree $d$, or it is $\epsilon$-far from any such polynomial. Later, the problem of low-degree testing played a central role in the development of probabilistic checkable proofs (PCP), where the goal is to probabilistically verify the validity of a given proof. For the problem of low-degree testing, Rubinfeld and Sudan [48] presented a tester with query complexity of $O(d^2 + d \cdot \epsilon^{-1})$. This test was further analyzed in [14]. The reader is also referred to [16], where a linearity test

---

1 More precisely, distribution-free property testing is the analogue of the PAC+MQ model of learning (that was studied by the learning-theory community mainly via the EQ+MQ model), standard property-testing is the analogue of the uniform+MQ model.
(which tests whether a given function acts as an homomorphism between groups) is presented, and to [12], [13], [29] and [37] for other related work.

**Monotonicity testing.** Monotonicity has also been a subject of a significant amount of work in the property-testing literature (e.g. [15, 19, 20, 21, 24, 25, 27]). In monotonicity testing, the domain \( \mathcal{X} \) is usually the \( d \)-dimensional cube \([n]^d\). A partial order is defined on this domain in the natural way (for \( \vec{y}, \vec{z} \in [n]^d \), we say that \( \vec{y} \leq \vec{z} \) if each coordinate of \( \vec{y} \) is bounded by the corresponding coordinate of \( \vec{z} \)). A function \( f \) over the domain \([n]^d\) is monotone if whenever \( \vec{z} \geq \vec{y} \) then \( f(\vec{z}) \geq f(\vec{y}) \). Testing algorithms were developed to deal with both the low-dimensional and the high-dimensional cases (with respect to the uniform distribution over the domain). In what follows, we survey some of the known results on this problem which are most relevant to our work.

- The low-dimensional case: In this case, \( d \) is considered to be small compared to \( n \) (and, in fact, it is typically a constant); a successful algorithm for this case is typically one that is polynomial in \( 1/\epsilon \) and in \( \log n \). The first paper to deal with this case is by Ergün et al. [20] which presented an \( O(\frac{\log n}{\epsilon}) \) algorithm for the line (i.e., the case \( d = 1 \)), and showed that this query complexity cannot be achieved without using membership queries (this algorithm was generalized for any fixed \( d \) in [15]). For the case \( d = 1 \), there is a lower bound showing that testing monotonicity (for some constant \( \epsilon \)) indeed requires \( \Omega(\log n) \) queries [21].

- The high-dimensional case: In this case, \( d \) is considered as the main parameter (and \( n \) might be as low as 2); a successful algorithm is typically one that is polynomial in \( 1/\epsilon \) and \( d \). This case was first considered by Goldreich et al. [27] that showed an algorithm for testing monotonicity of functions over the boolean \((n = 2)\) \( d \)-dimensional hyper-cube to a boolean range using \( O(\frac{1}{\epsilon} d) \) queries. This result was generalized in [19] to arbitrary values of \( n \), showing that \( O(\frac{d \log^2 n}{\epsilon}) \) queries suffice for testing monotonicity of general functions over \([n]^d\), which is the best known result so far.

- Testing monotonicity over graph products (modular design of monotonicity testers): A product of two graphs \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \) is a graph denoted by \( G_1 \times G_2 \) with vertex set \( V_1 \times V_2 \) and edge set \( E \) that consists of edges of the form \(((u_1, v_1), (u_2, v_2))\) for \((u_1, u_2) \in E_1\) or \(((u_1, v_1), (u_2, v_2))\) for \((v_1, v_2) \in E_2\) (see Figure 1.1 for an example of a product of two graphs). As mentioned, most previous work in the area of monotonicity testing, focused on functions defined over the \( d \)-dimensional hypercube, \([n]^d\). One feature of \([n]^d\) is that it can be viewed as a product of two lower dimensional hypercubes, or as the \( d^{th} \) power of the line graph \([n]\). Indeed, in [19, Lemma 6], dimension reduction is used for the case of boolean functions; they show a connection between the distance of a given boolean function over \([n]^d\) from being monotone to the average distance from being monotone of the one dimensional functions it induces. Based on this connection, they use results regarding monotonicity testing over the line to get a tester for \([n]^d\). Their approach applies only to specific kind of testers (see Section 8.3.3); nevertheless, it gives a first indication that graph products may be a useful tool for constructing monotonicity testers.

- Other related work: Lower bounds for monotonicity testing of functions \( f : \{0,1\}^d \rightarrow \{0,1\} \) were shown in [24]: \( \Omega(\sqrt{d}) \) lower bound for non-adaptive, one-sided error algo-

\(^{2}\)In the case \( d = 1 \) this yields a linear order.
Figure 1.1: An example for a product of two graphs $G_1 \times G_2$

...
incidence lists of length \(d\) have also been studied in [17] and [30].

A different testing model was introduced by Parnas and Ron [43] for testing properties of all sparse graphs, rather than only the bounded-degree ones. In this case, the graphs are represented by incidence lists of varying length. For each vertex \(v\) the tester can query both the degree of \(v\) and the \(i^{th}\) neighbor of \(v\). The distance between two graphs is then measured by the fraction of edge modifications necessary to obtain the property, defined with respect to an upper bound \(m\) on the number of edges in the graph. They present a testing algorithm for graph diameters, in this model. In addition, they explain how many of the results proved in [31] for the bounded-degree case, can be transformed to this model. Among these properties are the testing algorithms for \(k\)-connectivity. This model was further studied in [5, 38], where the model is adjusted to deal with dense graphs as well, by allowing the tester to query whether there exists an edge in the graph between two given vertices. The authors study testing of graph bipartiteness and triangle freeness in this model.

Connectivity is a central property in sparse graphs; and as such, as mentioned above, it has been previously studied in the context of testing with respect to the uniform distribution. However, although extending the testing models for graph problems to the distribution-free setting seems natural, the problem of connectivity testing in the distribution-free setting has not been dealt with before. Moreover, as mentioned, the only known results for distribution-free testing of graph properties in general are impossibility results for graphs represented by adjacency matrix [28]. They proved that it is impossible to test a variety of partition problems (for which they showed testers with respect to the uniform distribution) in a distribution-free manner. The generalization of the adjacency matrix model to a distribution-free one, is straightforward; on the other hand, every graph is close to be connected in this representation with respect to any distribution\(^3\), hence testing connectivity is trivial. However, unlike the dense graph case, when dealing with testing models for sparse graphs, their generalization to the distribution-free case is not so straightforward, and the testing is not obvious.

1.2 Our Contributions

Our main contributions are distribution-free testers for some of the properties mentioned above: low-degree multivariate polynomials, low-dimensional monotone functions and connectivity of sparse graphs. All testers use similar query complexity to the one known for the uniform setting. We observe that the approach that stands behind the low-degree test can also be applied to the testing of other properties such as dictatorship and juntas functions [23, 45]. These algorithms are the first known distribution-free testers for non-trivial properties. By this, we answer a natural question that has already been raised explicitly by Fischer [22, Subsection 9.3] and is implicit in [28]. We emphasize that our algorithms work for any distribution \(D\) without having any information about \(D\).

On the other hand, we show that, though in the uniform setting testing of boolean functions defined over the boolean hypercube can be done using query complexity that is polynomial in \(\frac{1}{\epsilon}\) and in the dimension \(d\), in the distribution-free setting such testing requires a number of queries that is exponential in the dimension \(d\). By this we show an inherent exponential gap in the query complexity between the uniform and the distribution-free settings for a natural testing problem, for which the testing is possible in both settings using non-trivial query complexity.

\(^3\)Under any distribution measure on \(V \times V\), the total probability of a path of length \(n\) with minimal probability is at most \(\frac{1}{n}\).
As for the uniform setting, we present a testing approach that enables us to use known monotonicity testers for given graphs $G_1, G_2$, to test monotonicity over their product $G_1 \times G_2$, and show that it can be applied to allow modular design of testers in many interesting cases: this approach works whenever the functions are boolean, and also in certain cases for functions with general range. We demonstrate the usefulness of our results by showing how a careful use of this approach improves the query complexity of known testers.

**Distribution-free low-degree testing for polynomials (and more).** We show how to generalize the tester presented in [48] to a distribution-free tester with the same (up to a multiplicative constant factor of 2) query complexity ($O(d^2 + d \cdot \epsilon^{-1})$). The algorithm and its analysis are presented in Chapter 3.

The generalization of the uniform testing algorithm to a distribution free one is done, in this case, by adding another stage to the uniform tester. In this new stage, after verifying that the input function $f$ is close to some low-degree polynomial $g$ with respect to the uniform distribution, we check that $f$ is also close to this specific polynomial $g$ with respect to the given distribution $D$. For this purpose, our approach requires that we will be able to calculate the values of $g$ efficiently based on the values on $f$. This is a generalization of the notion of self-correctors for single functions (see [16]) to classes of functions (which was previously introduced in [48]). We observe that the same testing approach can be used for every property that is testable in the uniform distribution and has a self-corrector in the above sense (Chapter 4).

**Distribution-free connectivity testing for sparse graphs.** We consider three models for testing properties of sparse graphs in the distribution-free setting. These models are generalizations of the ones presented previously for sparse graphs in the context of uniform distribution testing. We show that for each of these models, connectivity can be tested in a distribution-free manner, using a number of queries that is independent of the size of the graph.

- The first model is for bounded-degree graphs [31]; we refer to this model as the *bounded degree model*. We show that it is possible to test graph connectivity, in this model, in a distribution-free manner, using a number of queries which is polynomial in $\frac{1}{\epsilon}$. The test is similar to the one presented in [31] for the uniform setting, however the analysis required for the distribution-free case is different (Section 5.1).

- The second model is for graphs with a bound $m$ on the total number of edges [43]. We refer to this model as the *edge-bounded model*. We show that in this model, it is possible to test connectivity in a distribution-free manner using a poly($\frac{1}{\epsilon}$) number of queries, whenever $m \geq n(1 + c)$ for some constant $c$ (Subsection 5.2.1).

- We then combine the two previous testing models, and deal with bounded-degree graphs with at most $m$ edges. We refer to this model as the *combined model*. This model was not studied before even in the uniform setting. Hence, we first present a uniform distribution tester for this model for every $m$; then, using the distribution-free tester for the edge-bounded model, we prove that there exists a distribution-free tester for graph connectivity in the combined model whenever $m \geq n(1 + c)$, for some constant $c$ (Subsection 5.2.2).
Distribution-free monotonicity testing in the low-dimensional case. We present a distribution-free monotonicity tester in the low-dimensional hyper-cube case. Specifically, we present an algorithm whose complexity is $O\left(\frac{\log n}{\epsilon} \cdot 2^d\right)$ queries. This is done by first considering the one-dimensional case (the “line”). In this case, we prove that an algorithm of [20] can be slightly modified to deal with the distribution-free case with the same query complexity of $O\left(\frac{\log n}{\epsilon}\right)$. Though it is possible to modify the original analysis for the distribution-free case, we choose to present a whole different analysis. We then show how to appropriately generalize this algorithm to deal with higher (yet, low) dimensions (a similar generalization approach was used in [15] for the uniform distribution case). The tester for the one-dimensional case and its generalization for higher dimensions appear in Chapter 6.

It is typical for known property-testing algorithms to be quite simple and the analysis of why these algorithms work is where the property $P$ in question requires understanding; indeed, Goldreich and Trevisan [32] proved that in certain settings this is an inherent phenomena: they essentially showed (with respect to the uniform distribution) that any graph-theoretic property that can be tested can also be tested (with a small penalty in the complexity) by a “generic” algorithm that samples a random subgraph and decides whether it has some property. Our work is no different in this aspect: our algorithms are similar to previously known algorithms and the main contribution is their analysis; in particular, that for the distribution-free case. Moreover, it is somewhat surprising that our distribution-free testers require no dramatically-different techniques than those used in the construction and the analysis of previous algorithms (that work for the uniform distribution case). We remark, however, that although all the distribution-free testers presented in this work can be viewed as variations of testers for the uniform distribution, in each of the problems the modification of the uniform-distribution test is different.\footnote{Indeed, in light of the negative result in [28], there can be no generic transformation of uniform-distribution testers into distribution-free ones.}

Lower bound for distribution-free monotonicity testing in the high-dimensional case. We show that while $O\left(\frac{d}{\epsilon}\right)$ queries suffice for testing monotonicity of boolean functions in the uniform case [27], in the distribution-free case it requires a number of queries that is exponential in the dimension $d$ (that is, there exists a constant $c$ such that $\Omega(2^{cd})$ queries are required for the testing). Hence, such testing is infeasible in the high-dimensional case. Our bound can be trivially extended to monotonicity testing of boolean functions defined over the domain $[n]^d$ for general values of $n$. By this, showing a gap between the query complexity of uniform and distribution-free testers for a natural testing problem, for which both uniform and distribution-free testing are possible using non-trivial query complexity (Chapter 7).

Monotonicity testing over graph products. We study the use of graph products in monotonicity testing from several aspects. Our results go in two main directions: First, we show that the approach of testing monotonicity of functions over graph products using testers for the original graphs, can be used in a variety of cases, and not just for $[n]^d$ or for specific types of tests. In addition, we further study the dimension reduction for $[n]^d$ and show that a more careful use of this approach may yield better results than what was known prior to our work.

Let $G_1$ and $G_2$ be arbitrary graphs. Denote by $\epsilon$ the distance of a function defined over the graph product $G_1 \times G_2$ from being monotone, and by $\epsilon_1$ (respectively, $\epsilon_2$) the average distance of functions it induces on copies of $G_1$ (respectively, $G_2$) from being monotone. Our
results establish certain relations between $\epsilon$, $\epsilon_1$ and $\epsilon_2$, and show that the existence of these relations allows using any known testers for functions defined over the original graphs, $G_1$ and $G_2$, as black boxes to test monotonicity of functions that are defined over their product, $G_1 \times G_2$. At first glance, it may seem as if $\epsilon \leq \epsilon_1 + \epsilon_2$ always holds. We prove that this inequality does not always hold even for Boolean functions; however, in many cases $\epsilon$ can indeed be bounded as a linear combination of $\epsilon_1$ and $\epsilon_2$. Specifically, we show that these relations between $\epsilon, \epsilon_1$ and $\epsilon_2$ hold for Boolean functions defined over graph products. That is, we show that $\epsilon \leq \epsilon_1 + \epsilon_2 + \min\{\epsilon_1, \epsilon_2\}$ for every Boolean function $f$ that is defined over $G_1 \times G_2$. For general range, we prove a linear bound for $\epsilon$ in terms of $\epsilon_1$ and $\epsilon_2$ in restricted types of products; specifically, whenever one of the graphs in question is the line, $\epsilon \leq 4(\epsilon_1 + \epsilon_2)$ for every (possibly non-Boolean) function $f$. An important example for such a product is the $d$-dimensional hypercube $[n]^d$ (see above). Indeed, in the special case of $[n]^d$, based on the relations found between $\epsilon, \epsilon_1$ and $\epsilon_2$ and on properties of the known tester for the line [19, 20], we are able to provide a new analysis of the algorithm of [19]. Our analysis improves the best known upper bound on the query complexity for the low-dimensional case from $O\left(\frac{d \cdot (\log n)^2}{\epsilon}\right)$ [19] to $O\left(\frac{d \cdot 4^d \log n}{\epsilon}\right)$ (this is an improvement for all $d \leq \frac{\log \log n}{2}$) and yields an $O\left(\frac{d^2 \cdot 4^d \cdot (\log n)^2}{\epsilon}\right)$ bound on the running time (Chapter 8).

1.3 Consecutive work

Ailon and Chazelle [1] present a new analysis to the monotonicity tester of [19]. Their analysis uses similar approach to the one presented in Section 8.3.3 and further improves the best known query complexity for the low-dimensional case to $O\left(\frac{d^2 \cdot 4^d \cdot \log n}{\epsilon}\right)$.

1.4 Organization:

In Chapter 2, we formally define the notions we are about to use in this work. In Chapter 3, we present a distribution-free tester for low-degree multivariate polynomials. Then, in Chapter 4 we extend our distribution-free tester to deal with any property that has both a uniform tester and a property self corrector. These results were also published in [33]. Chapter 5 considers connectivity testing for sparse graphs (these results were also published in [35]). Chapter 6 describes the distribution-free monotonicity tester for the low-dimensional case, and Chapter 7 describes the lower bound on the query complexity required for distribution-free testing monotonicity in the high-dimensional case. The monotonicity testing algorithms and their analysis were also published in [33] and the lower bound in [36]. Chapter 8 considers monotonicity testing over graph products (see also [34]) and finally, Chapter 9 presents some open questions.
Chapter 2

Definitions

We formally define the notion of being $\epsilon$-far from a property $P$ with respect to a given distribution $D$ defined over $X$, and of distribution-free testing. Assume that the range of the functions in question is $A$.

Definition 2.1 Let $D$ and $\mathcal{X}$ be as above. The $D$-distance between functions $f, g : \mathcal{X} \to A$ is defined by $\text{dist}_D(f, g) \overset{\text{def}}{=} \Pr_{x \sim D}\{ f(x) \neq g(x) \}$.

The $D$-distance of a function $f$ from a property $P$ (i.e., the class of functions satisfying the property $P$) is $\text{dist}_D(f, P) \overset{\text{def}}{=} \min_{g \in P} \text{dist}_D(f, g)$.

We say that $f$ is $(\epsilon, D)$-far from a property $P$ if $\text{dist}_D(f, P) \geq \epsilon$.

When the distribution in question is the uniform distribution over $\mathcal{X}$, we either use $U$ instead of $D$ or (if clear from the context) we omit any reference to the distribution. E.g., the phrase “$f$ is $\epsilon$-far from $P$” refers to distance according to the uniform distribution.

Next, we define the notion of distribution-free tester for a given property $P$.

Definition 2.2 A distribution-free two-sided error tester for a property $P$ is a probabilistic oracle machine $M$, which is given a distance parameter $\epsilon > 0$, and an oracle access to an arbitrary function $f : \mathcal{X} \to A$ and to samples of a fixed but unknown distribution $D$ over $\mathcal{X}$, and satisfies the following two conditions:

1. (The tester accepts with high probability a function that satisfies $P$.) If $f$ satisfies $P$, then $\Pr\{ M^{f,D} = \text{Accept} \} \geq \frac{2}{3}$.

2. (The tester rejects with high probability if $f$ is far from $P$ w.r.t. $D$.) If $f$ is $(\epsilon, D)$-far from $P$, then $\Pr\{ M^{f,D} = \text{Accept} \} \leq \frac{1}{3}$.

We present also a definition of one-sided error testers that accept every function that satisfies $P$.

Definition 2.3 A distribution-free one-sided error tester for a property $P$ is a probabilistic oracle machine $M$, which is given a distance parameter $\epsilon > 0$, and an oracle access to an arbitrary function $f : \mathcal{X} \to A$ and to samples of a fixed but unknown distribution $D$ over $\mathcal{X}$, and satisfies the following two conditions:

1. (The tester always accepts every function that satisfies $P$.) If $f$ satisfies $P$, then $\Pr\{ M^{f,D} = \text{Accept} \} = 1$. 


2. (The tester rejects with probability at least $\frac{2}{3}$ if $f$ is far from $\mathcal{P}$ w.r.t. $D$.) If $f$ is $(\epsilon, D)$-far from $\mathcal{P}$, then $\Pr\{M^{f,D} = \text{Accept}\} \leq \frac{1}{3}$.

All our testers, like many previously known testers, have one-sided error and always accept any function satisfying the property $\mathcal{P}$ in question.

The definition of a uniform distribution tester for a property $\mathcal{P}$ can be derived from the previous definition by omitting the sampling oracle (since the tester can sample in the uniform distribution by itself) and by measuring the distance with respect to the uniform distribution.

Notice that, because the distribution $D$ in question is arbitrary, it is possible that there are two different functions $f$ and $g$ such that $\text{dist}_D(f, g) = 0$. Specifically, it is possible that $f \in \mathcal{P}$ and $g \notin \mathcal{P}$. Since the notion of testing is meant to be a relaxation of the notion of decision problems, it is required that the algorithm accepts (with high probability) functions that satisfy $\mathcal{P}$, but may reject functions that have distance 0 from $\mathcal{P}$ (but do not satisfy $\mathcal{P}$).

The definition of distribution-free testing was previously introduced in [28].

In addition, note that membership queries allow the algorithm to query the value of the input function also in points with probability 0 (which is also the case with membership queries in learning theory)\footnote{It is not known whether MQ are essential in general for testing even in the uniform case (see [46]); this is known only for specific problems such as monotonicity testing (see [20]).}. 

---

\[^1\text{It is not known whether MQ are essential in general for testing even in the uniform case (see [46]); this is known only for specific problems such as monotonicity testing (see [20]).} \]
Chapter 3

Distribution-free Low-Degree Testers for Polynomials

The first problem studied in the field of property testing was that of testing of multivariate polynomials (see [4, 12, 13, 16, 29, 48]). Let $F$ be a finite field. In the problem of low-degree testing, with respect to the uniform distribution, the tester is given access to a function $f : F^m \rightarrow F$, a distance parameter $\epsilon$, and a degree $d$, and has to decide whether $f$ is a multivariate polynomial of total degree $d$, or is at least $\epsilon$-far (with respect to the uniform distribution) from any degree $d$ multivariate polynomial; this implies that for every degree $d$ multivariate polynomial $g$, the probability that a uniformly drawn point $x$ has a value $g(x)$ different than $f(x)$, is at least $\epsilon$). Rubinfeld and Sudan [48] presented a tester for this problem with query complexity $O(d^2 + d \cdot \epsilon^{-1})$. We show how to modify this tester to a distribution-free one with the same query complexity (up to a constant factor of 2).

3.1 Preliminaries

Fix some value for $d$ and assume from now on that $|F| > 10d$. To describe the testers (both the one for the uniform distribution and our distribution-free one), we use the following terminology, from [48]:

- A line in $F^m$ is a set of $10d + 1$ points of the form $\{x, x + h, \ldots, x + 10dh\}$ for some $x, h \in F^m$. The line defined by $x$ and $h$ is denoted $\ell_{x,h}$.

- We say that a line $\ell_{x,h}$ is an $f$-polynomial, if there exists a univariate polynomial $P_{x,h}(i)$ of degree $d$, such that $f(x + ih) = P_{x,h}(i)$, for every $0 \leq i \leq 10d$.

Notice that if $f$ is a multivariate polynomial of total degree at most $d$, then for every $x$ and $h$, the line $\ell_{x,h}$ is an $f$-polynomial.\footnote{To see that, assume $f(x) = \Sigma_j a_j \prod_{l=1}^{d_j} x_{k_{j_l}}$, where $a_j$ is the coefficient of the $j'$th term in $f$, $d_j$ is the degree ($d_j \leq d$), and $k_{j_l}$ is the index of the $l'$th variable in that term. In this case, for every fixed $x = (x_1, \ldots, x_m)$ and $h = (h_1, \ldots, h_m)$ the value $f(x + ih) = \Sigma_j a_j \prod_{l=1}^{d_j} (x_{k_{j_l}} + ih_{k_{j_l}})$, which, of course, is a degree $d$ univariate polynomial in $i$.} Given the values of $f$ on a line $\ell_{x,h}$, testing whether this line is an $f$-polynomial can be done as follows:
• find, using interpolation, a univariate polynomial $P(i)$ of degree $d$, consistent with the values of $f$ at the $d + 1$ points $x, x + h, \ldots, x + dh$ (i.e., $P(i) = f(x + hi)$ for every $0 \leq i \leq d$).

• check, for every $(d + 1) \leq i \leq 10d$, that $f(x + ih) = P(i)$. If so accept; otherwise reject.

We show how this basic test is used to build a uniform and a distribution-free low-degree test.

### 3.2 Low-degree test for the uniform distribution

The low-degree test for the uniform distribution is done by randomly sampling $O(d + \epsilon^{-1})$ lines (i.e., by uniformly choosing $x, h \in F^m$), and checking that each of these lines is an $f$-polynomial. The correctness of this algorithm follows immediately from the following theorem ([48, Theorem 9]).

**Theorem 3.2.1** There exists a constant $c_U$ such that for $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, if $f$ is a function from $F^m$ to $F$, such that all but at most $\delta$ fraction of the lines $\{\ell_{x,h} | x, h \in F^m \}$ are $f$-polynomials, then there exists a polynomial $g : F^m \rightarrow F$ of total degree at most $d$ such that $\text{dist}_U(f, g) \leq (1 + o(1))\delta$ (provided that $|F| > 10d$).

### 3.3 Distribution-free low-degree testing algorithm

Denote the class of multivariate polynomials of total degree $d$ by $P_{deg}^d$. In this section we show that the tester described in the previous subsection can be modified into a distribution-free tester for low-degree multivariate polynomials. That is, we present an algorithm with query complexity $O(d^2 + d \cdot \epsilon^{-1})$, that given a distance parameter $\epsilon$, a degree parameter $d$, and access to random sampling of $F^m$ according to $D$ and to membership queries of a function $f : F^m \rightarrow F$, distinguishes, with probability of at least $\frac{2}{3}$, between the case that $f$ is in $P_{deg}^d$, and the case that $f$ is $(\epsilon, D)$-far from $P_{deg}^d$.

The natural generalization of the uniform-distribution tester above for the distribution-free case would be to replace the sampling of the tested lines by sampling according to the distribution $D$, i.e., sample the $O(d + \epsilon^{-1})$ lines by choosing $x \sim D$ and $h \sim U$ and check that these lines are $f$-polynomials. However, we do not know whether this modification actually works. Instead, the algorithm we present consists of two stages – in the first stage we simply run the uniform distribution test as is, and check that the function $f$ is $\epsilon$-close to $P_{deg}^d$ with respect to the uniform distribution; the second stage is the generalization suggested above. This combined strategy is presented in Figure 3.1, and we prove that it actually works.

**Theorem 3.3.1** Algorithm $\text{Poly}(\epsilon, d)$ is a distribution-free one-sided error tester for $P_{deg}^d$; its query complexity is $O(d^2 + d \cdot \epsilon^{-1})$.

The correctness of the algorithm relies on the following lemma:

**Lemma 3.3.2** Let $c_U$ be the constant as above. For every $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, if $f$ is a function from $F^m$ to $F$ such that

- $\Pr_{x,h \sim U}\{\ell_{x,h} \text{ is not an } f \text{ polynomial} \} \leq \delta$, and
- $\Pr_{x \sim D, h \sim U}\{\ell_{x,h} \text{ is not an } f \text{ polynomial} \} \leq \delta$,
It was shown that the function $f$ every

**Proof:** We prove that if the same way as in [48]:

- Choose $x, h \in R F^m$. If the line $\ell_{x,h}$ is not an $f$-polynomial, return **FAIL**.
- Choose $x \in D F^m, h \in R F^m$. If the line $\ell_{x,h}$ is not an $f$-polynomial, return **FAIL**.

\[ \text{return PASS} \]

Figure 3.1: Distribution-free low degree tester

then there exists a polynomial $g : F^m \rightarrow F$ of total degree at most $d$ such that $\text{dist}_D(f, g) \leq \delta$ (provided that $|F| > 10d$).

**Proof:** We prove that if $f$ satisfies the two conditions for $0 \leq \delta \leq \frac{1}{cd}$, then it is indeed close to $P_{\text{deg}}^d$. For the purpose of the analysis, we construct a function $g$ based on $f$, in the same way as in [48]:

- For every $x, h \in F^m$, denote by $P_{x,h}(i)$ the univariate polynomial that satisfies $P_{x,h}(i) = f(x + i h)$ for at least $6d$ of the values $i \in \{0, \ldots, 10d\}$. If no such polynomial exists, define $P_{x,h}$ to be “error”. Notice that there can be at most one such polynomial, since every two polynomials that satisfy the demand have to agree on at least $2d$ points, implying that they are identical.

- For every $x \in F^m$, define $g(x) = \text{plurality}_h\{P_{x,h}(0)\}$ where the plurality is taken over all values of $h$ for which $P_{x,h}$ is not an “error” ($g(x)$ is well defined for every $x$ since, by [48, Lemma 12], $\Pr_{h_1, h_2}\{P_{x,h_1}(0) = P_{x,h_2}(0)\} \geq 1 - 20\delta$).

It was shown that the function $g$ is indeed a degree $d$ multivariate polynomial (see the proof of [48, Theorem 9] and notice that $f$ satisfies the condition of that theorem). Moreover, for every $x \in F^m, i \in \{0, \ldots, 10d\}$, $\Pr_{h \sim U}[g(x + i h) = P_{x,h}(i)] \geq 1 - 40\delta$ (see [48, Corollary 13]).

To complete the proof, it remains to show that $\text{dist}_D(f, g) \leq \frac{\delta}{1 - 40\delta}$. Let $B$ be the set of $x$'s that satisfy $f(x) \neq P_{x,h}(0)$ for at least $1 - 40\delta$ fraction of the $h$'s in $F^m$ (including the values of $h$ for which $P_{x,h}$ was defined to be “error”). Notice that if $x \notin B$ then $f(x) = P_{x,h}(0)$ for more than $40\delta$ fraction of the $h$’s in $F^m$ (this is due to the fact that $f(x) \neq P_{x,h}(0)$ for less than $1 - 40\delta$ fraction of the $h$'s), hence it has to be equal to $P_{x,h}(0)$ for more than $40\delta$ fraction of the $h$’s). Therefore, for every $x \notin B$ we can claim the following:

1. $g(x) = P_{x,h}(0)$ for at least $1 - 40\delta$ fraction of the $h$’s (this holds for every $x$, and thus for every $x \notin B$). Denote this fraction by $g_x$.

2. $f(x) = P_{x,h}(0)$ for more than $40\delta$ fraction of the $h$’s in $F^m$. Denote this fraction by $f_x$.

Therefore, for at least one value of $h$ both $f(x) = P_{x,h}(0)$ and $g(x) = P_{x,h}(0)$ (otherwise, $f_x + g_x > 1$, since $g_x \geq 1 - 40\delta$ and $f_x > 40\delta$). Hence, we conclude that for every $x \notin B$, $f(x) = g(x)$. Let us bound the probability of picking a point in $B$ according to the distribution $D$.

\[ \delta \geq \Pr_{x \sim D, h \sim U}\{\ell_{x,h} \text{ is not an } f \text{ polynomial}\} \geq (1 - 40\delta) \Pr_{x \sim D}\{x \in B\}, \]
where the first inequality is due to the second condition of the lemma, and the second inequality follows the previous discussion, implying that

\[ \Pr_{x \sim D} \{ x \in B \} \leq \frac{\delta}{1 - 40\delta} . \]

\[ \square \]

**Proof of theorem 3.3.1:** To prove that the algorithm is indeed a distribution-free tester for \( \mathcal{P}_{\text{deg}}^d \), we prove the following two facts:

1. If \( f \) is in \( \mathcal{P}_{\text{deg}}^d \), then the algorithm accepts \( f \) with probability 1.
2. If \( f \) is \((\epsilon, D)\)-far from \( \mathcal{P}_{\text{deg}}^d \), then the algorithm \( \text{Poly}(\epsilon, d) \) rejects \( f \) with probability of at least \( \frac{2}{3} \).

As explained before, if \( f \) is indeed a multivariate polynomial of total degree \( d \), then every line is an \( f \)-polynomial. Hence, it follows that such \( f \) is accepted by the tester with probability 1. Assume from now on that \( f \) is \((\epsilon, D)\)-far from \( \mathcal{P}_{\text{deg}}^d \). Notice that, by the definition of \( k \), for \( \epsilon' = \frac{1}{k} \), \( f \) is \((\epsilon', D)\)-far from \( \mathcal{P}_{\text{deg}}^d \). By Lemma 3.3.2, either \( \Pr_{x,h \sim U} \{ \ell_{x,h} \text{ is not an } f \text{ polynomial} \} > \frac{\epsilon'}{2+40\epsilon'} \), or \( \Pr_{x \sim D, h \sim U} \{ \ell_{x,h} \text{ is not an } f \text{ polynomial} \} > \frac{\epsilon'}{2+40\epsilon'} \). (otherwise, it follows that there exists a degree \( d \) polynomial \( g \) such that \( \text{dist}_D(f, g) \leq \frac{\epsilon'}{2+40\epsilon'} \), contradicting the fact that the \( D \)-distance of \( f \) from any such polynomial is at least \( \epsilon' \)). Assume that the first event occurs. Therefore, the probability that a randomly chosen line \( \ell_{x,h} \) is an \( f \)-polynomial is at most \( (1 - \frac{\epsilon'}{2+40\epsilon'})^k \). Hence, the probability that the algorithm accepts \( f \) is at most \( (1 - \frac{\epsilon'}{2+40\epsilon'})^k = (1 - \frac{1}{2k+40})^k \leq \frac{4}{e} \leq \frac{4}{3} \) (the first inequality follows the fact that \( c_U \geq 100 \) implying that \( k \geq 100 \)). Similarly, if the second event occurs, the probability that a randomly chosen line \( \ell_{x,h} \), where \( x \sim D \) and \( h \sim U \), is an \( f \)-polynomial is at most \( (1 - \frac{\epsilon'}{2+40\epsilon'})^k \). Hence, as before, the probability that the algorithm accepts \( f \) is at most \( (1 - \frac{\epsilon'}{2+40\epsilon'})^k \leq \frac{1}{3} \). \[ \square \]
Chapter 4

Distribution-Free Testing of Properties with Self-Correctors

A careful examination of the distribution-free tester presented in the previous section, shows that, in fact, the only two features of low-degree multivariate polynomials used in the construction are:

- the existence of a one-sided error uniform distribution tester for low-degree polynomials, and
- the ability to efficiently compute (with high probability), in every point $x$ of the domain, the correct value of the polynomial $g$ that is close to the input function $f$, if $f$ is indeed close to a multivariate low-degree polynomial. We refer to this ability as “property self-correction”.

We argue that it is possible to construct a distribution-free tester for every property $P$ that satisfies these two conditions. We first define the notion of a “property self-correction” formally (it has already been defined implicitly and used in [48]), and then introduce a general scheme for obtaining distribution-free testers for a variety of properties that satisfy the conditions.

The notion of “property self-corrector” is a generalization of the notion of self-correctors for functions that was introduced by Blum, Luby and Rubinfeld in [16]. A self-corrector for a specific function $f$ is a randomized algorithm that given oracle access to a function $g$ which is $\epsilon$-close to $f$, is able to compute the value of $f$ in every point of the domain. This definition can be generalized to classes of functions, specifically demanding that all the functions in the class are self-correctable using the same algorithm.

**Definition 4.0.3** An $\epsilon$ self-corrector for a property $P$ is a probabilistic oracle machine $M$, which is given an oracle access to an arbitrary function $f : X \rightarrow A$ and satisfies the following conditions:

- If $f \in P$, then $\Pr\{M^f(x) = f(x)\} = 1$ for every $x \in X$.
- If there exists a function $g \in P$ such that $\text{dist}_U(f, g) \leq \epsilon$ (i.e., $f$ is $\epsilon$-close to $P$), then $\Pr\{M^f(x) = g(x)\} \geq \frac{2}{3}$, for every $x \in X$.

Note that the definition of “property self-corrector” refers to distance measured only with respect to the uniform distribution, however, we still use these correctors for the construction
of distribution-free testers. Observe that a necessary condition for the existence of an $\epsilon$-self-corrector for a property $P$ is that for every function $f$ such that $\text{dist}_U(f, P) \leq \epsilon$ (i.e., $f$ is $\epsilon$-close to $P$ with respect to the uniform distribution), there exists a unique function $g \in P$ that is $\epsilon$-close to $P$. Notice that the property of monotonicity does not fulfill this requirement. Hence, the distribution-free monotonicity tester that is presented in the next section requires a different approach.

Next, in Figure 4.1, we describe the generalized distribution-free testing scheme. Let $P$ be a property, let $T_P$ be a uniform distribution tester for $P$ with query complexity $Q_T$ that has one-sided error, and let $C_P$ be an $\epsilon'$ property self-corrector for $P$ with query complexity $Q_C$. Let $\epsilon \leq \epsilon'$, and $f : X \rightarrow A$.

<table>
<thead>
<tr>
<th>Algorithm Tester$_D(\epsilon)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run $T_P^f(\epsilon)$. If $T_P^f(\epsilon) = \text{FAIL}$, then return $\text{FAIL}$</td>
</tr>
<tr>
<td>Repeat $\frac{2}{3}$ times:</td>
</tr>
<tr>
<td>Choose $x \in D$.</td>
</tr>
<tr>
<td>Repeat twice: Run $C_P^f(x)$; If $f(x) \neq C_P^f(x)$, then return $\text{FAIL}$</td>
</tr>
<tr>
<td>return $\text{PASS}$</td>
</tr>
</tbody>
</table>

Figure 4.1: Generalized distribution-free testing scheme

Theorem 4.0.4 Algorithm Tester$_D(\epsilon)$ is a one-sided error distribution-free tester for $P$ with query complexity $Q_T(\epsilon) + \frac{2}{3} \cdot Q_C$.

Proof: It is obvious that the query complexity of the algorithm Tester$_D(\epsilon)$ is indeed as required. Hence, we only have to prove the correctness of the algorithm. To do so, we prove the following two facts:

- if $f \in P$ then $f$ is accepted by the algorithm with probability 1.
- if $f$ is $(\epsilon, D)$-far from $P$, then $f$ is rejected by Tester$_D(\epsilon)$ with probability of at least $\frac{2}{3}$.

If $f$ is indeed in $P$, then it passes the uniform test with probability 1, and the value returned by the self-corrector is always identical to the value of $f$. Hence, $f$ is accepted by the algorithm. Assume from now on that $f$ is $(\epsilon, D)$-far from $P$. In this case we distinguish between two possibilities:

- If $f$ is $(\epsilon, U)$-far from $P$, then the probability that it passes the uniform test is at most $\frac{1}{3}$.
- If $f$ is $(\epsilon, U)$-close to $P$, then there exists a function $g \in P$ such that $\text{dist}(f, g) \leq \epsilon$. However, since $\text{dist}_D(f, P) \geq \epsilon$, we deduce that $\text{dist}_D(f, g) \geq \epsilon$ (in other words, $\Pr_{x \sim D}\{f(x) \neq g(x)\} \geq \epsilon$). If $f$ was accepted by the algorithm then one of the two following events happened: either we failed to sample a point in which $f$ and $g$ differ, or we succeeded to sample such a point, but both runs of the self-corrector failed to compute the value of $g$ in this point. The probability of the first event is at most

\footnote{Consider for example the following function $f : [n] \rightarrow \{0, 1\}$: for every $1 \leq i \leq \frac{n}{2}$ set $f(i) = 1$, and for every $\frac{n}{2} + 1 \leq i \leq n$ set $f(i) = 0$. $f$ is $\frac{1}{2}$-far from monotone, and it is $\frac{1}{2}$-close to both constant functions: 0 and 1.}
By the definition of a property self-corrector, the probability of the second event is at most $\frac{1}{3} < \frac{1}{6}$, hence, in both cases the probability that $f$ is accepted by the algorithm is at most $\frac{1}{3}$. 

Remark 4.0.1: We used the assumption that there exists a uniform distribution tester for the property $P$ that has one-sided error. However, the same transformation can be applied also when the uniform distribution tester has two-sided error, only that the resulting distribution-free tester as well has two-sided error.

As was previously stated, the algorithm that was explicitly presented in Chapter 3 can actually be described as an application of this scheme for the class of low-degree multivariate polynomials. Hence, instead of fully describing the distribution-free tester and proving its correctness, it was enough to show that this property can be tested in the uniform distribution and that it can be self-corrected.

This scheme also implies the existence of distribution-free property testers for other properties. Among these properties are juntas and dictatorship functions. A function $f : \{0,1\} \rightarrow \{0,1\}$ is said to be a $k$-junta if there exists a subset of $\{x_1, \ldots, x_n\}$ of size $k$ that determines the value of $f$ (i.e., $f$ is independent of the other variable). A special case of juntas are dictatorship functions, where a single variable determines the value of the function. These properties (and other related properties) have uniform distribution testers, as was shown in [23, 45]. In addition, they are subsets of the class of low-degree polynomials (for example, $k$ juntas are a special case of degree $k$ multivariate polynomials), and thus are self correctable. Therefore, we can apply the scheme described in this section to obtain distribution-free testers for these properties.

Remark 4.0.2: Given two properties $P$ and $P'$ such that $P' \subseteq P$, the fact that $P$ is testable in the uniform distribution does not imply that $P'$ is thus testable (to see this observe that every property is a subset of the class of all functions that is clearly testable). However, the fact that $P$ is self-correctable implies that $P'$ is self-correctable (using the same correction algorithm).
Chapter 5

Distribution-Free Connectivity Testing for Sparse Graphs

This chapter deals with distribution-free testing of connectivity for sparse graphs. As described in the introduction, we consider three representations of sparse graphs in the distribution-free setting, and show that for each of these representations, connectivity can be tested in a distribution-free manner, using a number of queries independent of the size of the graph. The chapter is organized as follows: In Sections 5.1 we focus on testing of bounded degree graphs. Section 5.2 deals with the edge-bounded and the combined models, described previously. That is, general and bounded degree graphs with a bounded number of edges.

5.1 Connectivity testing of bounded-degree graphs

The representation for bounded-degree graphs which is the subject of this section is a generalization of [31] to deal with arbitrary probability distributions. First, we generalize the notion of a function $f_G : V \times [d] \rightarrow V \cup \{\perp\}$ that represents a degree $d$ (undirected) graph $G$. Since the $d$ outgoing edges of a vertex $v$ may have different probabilities, we allow $f_G(v, i) = \perp$ even if $f_G(v, i + 1) \neq \perp$ (i.e., we do not assume that a node of out-degree $d'$ uses the first $d'$ entries in its incidence list).

**Definition 5.1.1** A function $f_G : V \times [d] \rightarrow V \cup \{\perp\}$ represents a graph $G = (V, E)$, if the following holds:

1. for every edge $(u, v) \in E$ there exist unique $i_u$ and $i_v$ such that $f_G(u, i_u) = v$ and $f_G(v, i_v) = u$; and
2. for every vertex $v \in V$ and $i \in [d]$, if there exists no neighbor $u$ of $v$ such that $f_G(v, i) = u$, then $f_G(v, i) = \perp$. In this case, we say that the pair $(v, i)$ is free in $f_G$.

Denote by $P^d$ the class of functions $f$ that represent connected graphs with bounded-degree $d$. Given a probability measure $D : V \times [d] \rightarrow [0, 1]$ and a function $f_G$ that represents a graph $G$, we are interested in the $D$-distance of $f_G$ from $P^d$. Hence, we examine possible ways to transform $f_G$ into a function $f_G'$ that represents a connected graph $G'$ with degree $d$ (i.e., $f_G' \in P^d$). To do so, we have to connect all the connected components of $G$; that is, if

\[\text{In order to allow parallel edges in the definition, we refer to the } j^{th} \text{ copy of the edge } (u, v) \text{ as a distinct edge } (u_j, v_j).\]
the connected components of $G$ are $C_1, \ldots, C_k$, then we wish to add an edge between $C_i$ and $C_{i+1}$ while, at the same time, preventing the graph’s degree from exceeding $d$, and keeping the connectivity of each of the connected components. In addition, we want the total probability of the modified (i.e., added and removed) edges to remain small. Therefore, we look, in any connected component $C_i$ of $G$, either for an edge that is unnecessary to the connectivity, or for two distinct free pairs $(v, i_v)$ and $(u, i_u)$. In other words, we find a list of possible connecting points between the connected components of $G$. We define more accurately the notion of such a list. For this purpose, we need the following definition.

**Definition 5.1.2** Let $G = (V, E)$ be a graph, let $e = (u, v) \in E$ be an edge, and let $C$ be a connected component of $G$ containing $e$. We say that $e$ is redundant, if removing $e$ from $E$ does not affect the connectivity of $C$.

The following definition formally states the idea of a list of possible connections between the connected components of a graph. The $m^{th}$ edge in this list, $(u_m, v_m)$, will connect the components $C_m$ and $C_{m+1}$.

**Definition 5.1.3** Let $f_G$ be a function that represents a graph $G$ with bounded-degree $d$, and let $C_1, \ldots, C_k$ be the connected components of $G$. We say that a list of $k$ quad-tuples $L = ((u_1, i_1), (v_1, j_1)), \ldots, ((u_k, i_k), (v_k, j_k))$ connects $f_G$ if the following holds, for every $m \in [k]$:

1. $u_m \in C_m$ and $v_m \in C_{m+1}$ (in case $m = k$, then $m + 1$ refers to 1).
2. $(v_m, j_m) \neq (u_{m+1}, i_{m+1})$.
3. one of the following holds:
   - (a) $(u_{m+1}, i_{m+1})$ and $(v_m, j_m)$ are free pairs in $f_G$; or
   - (b) $(u_{m+1}, v_m)$ is redundant in $C_{m+1}$, $f_G(u_{m+1}, i_{m+1}) = v_m$, and $f_G(v_m, j_m) = u_{m+1}$.

Indeed, given a list $L$ that connects $f_G$, we can construct from $f_G$ a function $f_G'$, that represents a connected degree $d$ graph $G'$, by setting $f_G'(u_m, i_m) = v_m$ and $f_G'(v_m, j_m) = u_m$ (notice that the graph $G'$ is obtained from $G$ by removing at most one redundant edge from each connected component, thereby not damaging the component’s connectivity) $^2$.

Let $f_G$ be a function that represents a graph $G$. For every edge $(u, v) \in E$, define the $D$-probability of the edge $(u, v)$ to be the total probability of its endpoints with respect to $D$ (it can be seen as the cost of a change in the edge $e = (u, v)$, since a change in $e$ requires a change in the incidence lists of both $u$ and $v$); equivalently, let $i_u$ and $i_v$ be such that $f_G(u, i_u) = v$ and $f_G(v, i_v) = u$, then the $D$-probability of the edge $(u, v)$ is $D(u, v) \overset{\text{def}}{=} D(u, i_u) + D(v, i_v)$. In addition, define the $D$-probability of a vertex $v \in V$ to be $D(v) \overset{\text{def}}{=} \sum_{i \in [d]} D(v, i)$ and the $D$-probability of a list $L$ to be $D(L) \overset{\text{def}}{=} \sum_m (D(u_m, i_m) + D(v_m, j_m))$.

**Observation 1:** If a list $L$ connects $f_G$, then the $D$-distance of $f_G$ from $P_d^d$ is at most $D(L)$.

---

$^2$The above definition actually describes a cycle connecting all the connected components of the graph. It is possible to define a path from $C_1$ to $C_k$; however, in such a case, not all the quad-tuples are symmetric, causing the definition to be slightly more complicated.
5.1.1 Testing connectivity for graphs with bounded-degree $d \geq 3$

In this section, we present an $O(\epsilon^{-2})$ distribution-free tester for connectivity of bounded-degree graphs that, given access to random samples of $V \times [d]$ according to the distribution $D$ and to membership queries of a function $f_G$, distinguishes, with probability of at least $2/3$, between the case that $f_G$ represents a connected graph with bounded-degree $d$ (i.e., $f_G \in \mathcal{P}^d$), and the case that $f_G$ is $(\epsilon, D)$-far from $\mathcal{P}^d$.

The tester is similar to the one presented in [31] for the uniform distribution, in the sense that it also looks for small connected components in $G$. However, while the original analysis is based on the fact that the number of small connected components in a graph $G'$ which is far from being connected, is big, this claim no longer holds when dealing with arbitrary distributions\(^3\). Hence, a whole new analysis is required for the distribution-free case. A natural generalization of the tester for the uniform case may seem to be seeking for connected components where the total probability of their vertices is small (note that, there is no correlation between the number of vertices in a connected component and their total probability). However, there are some drawbacks to this approach. First of all, in the distribution-free setting we have no knowledge of the actual probability $D$ of the sampled points, and are only allowed to sample the domain according to $D$; hence, we are only able to estimate their probability. In addition, the size of such components may be very large, therefore finding out whether two vertices lie in the same component may not be possible using a number of queries that is independent of the size of the graph. Thus, a different generalization is required. Our tester works in phases. In each phase it samples an edge $e$ in the graph according to the distribution $D$, and checks whether the edge $e$ belongs to a small connected component. The algorithm appears in Figure 5.1.

```
Algorithm - connectivity(\epsilon, d)
Repeat \(4\epsilon\) times:
- Choose, using the sampling oracle, \((v, i) \sim D\).
- Perform BFS starting from $v$ until \(96\epsilon d\) vertices have been reached, or no new vertex can be reached. If the search was ended since no new vertex can be reached, return FAIL\(^a\).

return PASS
```

\(^a\)Assume $|V| \geq 96\epsilon d$. If $|V| < 96\epsilon d$, then it is possible to decide whether the graph is connected in query complexity $O(\epsilon^{-1})$.

Figure 5.1: Distribution-free tester for $d \geq 3$

To prove the correctness of this algorithm, we use the following notation. Let $f_G$ be a function that represents a graph $G$, and let $C$ be a connected component of $G$; denote by $n_C$ the number of vertices in $C$ and define $w_C = \sum_{v \in C} D(v)$ (i.e., the total probability of all the vertices in $C$). Similarly, denote by $n_{\chi(C)}$ the number of free pairs $(v, i)$ in $f_G$ such that $v \in C$ (note that $n_{\chi(C)} \leq n_C \cdot d$ and that $n_{\chi(C)}$ is independent of the specific representation $f_G$) and

\[^3\]Consider for example a graph $G$ that consists of two connected components, one of size $|V| - 1$ and the other containing a single vertex $v_0$, and the distribution $D$ is set to be $D(v_0, i) = \frac{1}{d}$ for every $1 \leq i \leq d$. In this case, $G$ is $(\frac{1}{d}, D)$-far from being connected, while it contains only one small connected component.

25
define \( w_{\chi(C)}^G = \sum_{v \in C, \, i \in [d]} f_G(v, i) = D(v, i) \), i.e. the total probability of free pairs in \( C \) (clearly, \( 0 \leq w_{\chi(C)}^G \leq w(C) \)).

The following lemma shows that if there are few free pairs in \( C \), then there is a redundant edge in \( C \) with low \( D \)-probability.

**Lemma 5.1.4** Let \( f_G \) be a function that represents a graph \( G \), and let \( C \) be a connected component of \( G \). If \( n_{\chi(C)} < \frac{nc_d}{4} \), then there exists a redundant edge \((u, v)\) in \( C \) such that \( D(u, v) \leq \frac{24w_c}{nc_d} \).

**Proof:** Let \( C \) be a connected component of \( G \) for which \( n_{\chi(C)} < \frac{nc_d}{4} \). There are at least \( \frac{3}{4} \cdot n_C \cdot \frac{d}{2} \) edges in \( C \) (otherwise there will be more free pairs). Since at most \( n_C \) of these edges are essential for the connectivity of \( C \), at least \( \frac{3nc_d}{8} - n_C \) of these edges are redundant. That is, at least \( n_C(\frac{3d}{8} - 1) \geq n_C \cdot d(\frac{3}{4} - \frac{1}{3}) = \frac{nc_d}{24} \) of the edges in \( C \) are redundant, where the last inequality follows the fact that \( d \geq 3 \). Denote the number of redundant edges by \( x \), and their total probability by \( w \). Therefore, there is at least one edge \((u, v)\) for which \( D(u, v) \leq w \leq \frac{w_c}{x} \leq \frac{24w_c}{nc_d} \).

The next lemma deals with the case that there are many free pairs in \( C \) and shows that in this case there are two free pairs in \( C \) with low probability.

**Lemma 5.1.5** Let \( f_G \) be a function that represents a graph \( G \), and let \( C \) be a connected component of \( G \). If \( n_{\chi(C)} \geq \frac{nc_d}{4} \geq 2 \), then there exist two distinct free pairs \((v_1, i_1)\) and \((v_2, i_2)\) in \( f_G \), such that \( v_1, v_2 \in C \) and \( D(v_1, i_1), D(v_2, i_2) \leq 8 \frac{w_c}{nc_d} \).

**Proof:** Consider the set of free pairs \((v, i)\) in \( f_G \) such that \( v \in C \). Clearly, there exist two different elements in this set \((v_1, i_1)\) and \((v_2, i_2)\) for which \( D(v_1, i_1), D(v_2, i_2) \leq 2w_{\chi(C)}^G/n_{\chi(C)} \) (otherwise, \( w_{\chi(C)}^G > \left(n_{\chi(C)} - 1\right) \cdot 2w_{\chi(C)}^G/n_{\chi(C)} \geq w_{\chi(C)}^G \)). Since \( n_{\chi(C)} \geq \frac{nc_d}{4} \) and \( w_{\chi(C)}^G \leq w_c \), we deduce that \( 2w_{\chi(C)}^G/n_{\chi(C)} \leq 8 \frac{w_c}{nc_d} \).

The next lemma states that if \( f_G \) is \((\epsilon, D)\)-far from \( P^d \), then the total weight of the small connected components in the graph \( G \) is at least \( \frac{1}{2} \). Denote by \( S_D(f_G) \) the total probability of connected components in \( G \) whose size is smaller than \( \frac{96}{cd} \); i.e., \( S_D(f_G) \equiv \sum_{C: n_C < \frac{96}{cd}} w(C) \).

**Lemma 5.1.6** Let \( f_G \) be a function \((\epsilon, D)\)-far from \( P^d \). Then, \( S_D(f_G) \geq \frac{1}{2} \).

**Proof:** We show that if \( S_D(f_G) < \frac{1}{2} \), then \( dist_D(f_G; P^d) < \epsilon \). By Observation 1, it suffices to construct a list \( L \) that connects \( f_G \), such that \( D(L) < \epsilon \). Assume that \( C_1, \ldots, C_k \) are the connected components of \( G \), that \( C_1, \ldots, C_l \) are of size less than \( \frac{96}{cd} \), and \( C_{l+1}, \ldots, C_k \) are of size at least \( \frac{96}{cd} \). We construct the list \( L = ((u_1, i_1), (v_1, j_1)), \ldots, ((u_k, i_k), (v_k, j_k)) \) as follows:

- For \( 1 \leq m \leq l \), we distinguish between the following two cases:
  1. If there are at least two free pairs \((v, i)\) and \((v', i')\) in \( f_G \) such that \( v, v' \in C_m \), set \((u_m, i_m) = (v, i)\) and \((v_{m-1}, j_{m-1}) = (v', i')\) (i.e., we arbitrarily pick two free pairs to play the role of the endpoints of the edges connecting \( C_m \)).
  2. If there are less than two free pairs in \( C_m \), then there must be a redundant edge \((u, v)\) in \( C_m \). This is because we have at least \((d \cdot n_{C_m} - 1)/2 \) edges in \( C_m \); while \( n_{C_m} - 1 \) edges suffice for the connectivity of \( C_m \). Assume \( f_G(u, i) = v \) and \( f_G(v, i') = u \); we set \((u_m, i_m) = (u, i)\) and \((v_{m-1}, j_{m-1}) = (v, i')\).
For $l + 1 \leq m \leq k$, we distinguish between the following two cases:

1. If $n_{\chi(C_m)} < nC_m d/4$, let $(u, v)$ be the redundant edge guaranteed by Lemma 5.1.4, and let $i_u$ and $i_v$ be such that $f_G(u, i_u) = v$ and $f_G(v, i_v) = u$. We set $(u_m, i_m) = (u, i_u)$ and $(v_{m-1}, j_{m-1}) = (v, i_v)$.

2. If $n_{\chi(C_m)} \geq nC_m d/4$ (and we know $nC_m d/4 \geq 2$), let $(v, i)$ and $(v', i')$ be the two free pairs in $f_G$ guaranteed by Lemma 5.1.5, and set $(u_m, i_m) = (v, i)$ and $(v_{m-1}, j_{m-1}) = (v', i')$.

It is easily verified that the list $L$ indeed connects $f_G$. It is left to show that $D(L) < \epsilon$. Define the following partition of $[k]$: $M_1 = \{l\}$, $M_2 = \{m : l + 1 \leq m \leq k\}$, $n_{\chi(C_m)} < \frac{ncd}{4}$, and $M_3 = \{m : l + 1 \leq m \leq k, n_{\chi(C_m)} \geq \frac{ncd}{4}\}$.

$$D(L) = \sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) + \sum_{m \in M_2} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

Since $S_D(f_G) < \frac{\epsilon}{2}$, we have $\sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \leq S_D(f_G) < \frac{\epsilon}{2}$.

By Lemma 5.1.4 (and the construction), $\sum_{m \in M_2} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \leq \sum_{m \in M_2} 24 \frac{w_{C_m}}{nc_m d}$, and since for every $m \in M_2$ we have $n_{C_m} \geq \frac{96}{ed}$, then each summand is at most $24 \frac{w_{C_m} \cdot \epsilon}{4} = w_{C_m} \cdot \epsilon$.

Similarly, by Lemma 5.1.5 (and the construction), $\sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \leq \sum_{m \in M_3} 16 \frac{w_{C_m}}{nc_m d} \leq \sum_{m \in M_3} \frac{w_{C_m} \cdot \epsilon}{4}$.

All together, $D(L) < \frac{\epsilon}{2} + \sum_{m \in M_1 \cup M_2} \frac{w_{C_m} \cdot \epsilon}{4} < \epsilon$.

We can now prove the correctness of the algorithm `connectivity`.

**Theorem 5.1.7** Algorithm `connectivity`$(\epsilon, d)$ is a distribution-free tester for connectivity of bounded-degree graphs with degree $d \geq 3$; its query complexity is $O(\epsilon^{-2})$.

**Proof:** By the definition of a function representing a graph, in every stage of the BFS the algorithm looks for the next neighbor of the current vertex, which causes the BFS step to cost $O(d)$ queries (in the worst case). Since there are $O(\frac{\epsilon^{-2}}{d})$ BFS steps, the query complexity is as required. By the definition of the algorithm, if $f_G \in \mathcal{P}^d$, then it is accepted by the algorithm with probability 1. Let $f_G$ be $(\epsilon, D)$-far from $\mathcal{P}^d$. By Lemma 5.1.6, $S_D(f_G) \geq \frac{\epsilon}{2}$. Hence, the probability of the algorithm to randomly pick a pair $(v, i)$ such that $v \in C$ and $C$ is a connected component of size less than $\frac{96}{ed}$, is at least $\frac{\epsilon}{2}$. Therefore, the probability that the algorithm fails to find a small component is at most $\left(1 - \frac{\epsilon}{2}\right)^2 \leq \frac{12}{e} \leq \frac{1}{3}$. \hfill \Box

### 5.1.2 Testing connectivity for graphs with degree $d = 2$

This section focuses on degree-2 graphs, i.e., the class $\mathcal{P}^2$. Unlike the case of graphs with degree at least 3, it is not true that if $dist_D(f_G, \mathcal{P}^2)$ is large, then so is the total probability of the small connected components of $G$. To see this, consider a graph $G = (V, E)$ that consists of two lines of $\frac{1}{22}$ nodes each, with endpoints $v_1, v_2, v_3$ and $v_4$, and set the distribution $D$ to be $D(v_i, j) = 1/8$ for every $1 \leq i \leq 4$ and $j \in \{1, 2\}$, and 0 otherwise; $dist_D(f_G, \mathcal{P}^2) \geq \frac{1}{4}$.
while there are no small connected components in \( G \). Therefore, the tester described above no longer works.

We present an \( O(\epsilon^{-2}) \) distribution-free connectivity tester for degree 2 graphs. The algorithm is based on the following trivial observation.

**Observation 2:** A connected graph of degree 2 can be either a line or a circle.

Following this observation, in addition to a search for small connected components (which is not enough on its own for degree 2 graphs), our algorithm looks for three endpoints of lines in the graph, implying that it consists of more than one connected component. The algorithm for the case \( d = 2 \) appears in Figure 5.2.

Algorithm - connectivity2(\( \epsilon \))

- Repeat \( \frac{8}{\epsilon} \) times:
  - Choose, using the sampling oracle, \( (v, i) \sim D \).
  - Perform BFS starting from \( v \) until \( \frac{4}{\epsilon} \) vertices have been reached, or no new vertex can be reached. If the search was ended since no new vertex can be reached, return \( \text{FAIL} \).
- Choose \( \frac{120}{\epsilon} \) pairs \( (v_j, i_j) \) according to the distribution \( D \). If among these pairs there are at least 3 vertices with out-degree 1, then return \( \text{FAIL} \).

return \( \text{PASS} \).

**Figure 5.2:** Distribution-free tester for \( d = 2 \)

Let \( f_G \) be a function that represents a degree 2 graph \( G \). Similarly to what was done before, define \( S_D(f_G) \) to be the sum of \( D(v) \) over all vertices \( v \) that are in connected components of size less than \( \frac{4}{\epsilon} \) in \( G \), and \( S_1(f_G) \) to be the sum of \( D(v) \) over all vertices \( v \) whose degree in \( G \) is 1. The next lemma shows that if in the graph \( G \), the total weight of both the small connected components and the degree 1 vertices is small, then \( f_G \) is close to \( P^2 \).

**Lemma 5.1.8** Let \( f_G \) be a function that represents a graph \( G \) with bounded-degree 2. If \( S_D(f_G) < \frac{\epsilon}{4} \) and \( S_1(f_G) < \frac{\epsilon}{4} \), then there exists a function \( f_{G'} \) that represents a degree 2 connected graph \( G' \), such that \( \text{dist}_D(f_G, f_{G'}) < \epsilon \).

**Proof:** Let \( C_1, \ldots, C_k \) be the connected components of \( G \). As before, define \( w_{C_m} = \sum_{v \in C_m} D(v) \) and denote by \( n_{C_m} \) the number of vertices in \( C_m \). Again, we construct a list \( L \) that connects \( f_G \): \( L = ((u_1, i_1), (v_1, j_1)), \ldots, ((u_k, i_k)), (v_k, j_k)) \) such that \( D(L) < \epsilon \). In the construction we distinguish between three cases:

- \( C_m \) consists of a single vertex \( v \) – set \( (u_m, i_m) = (v, 1) \) and \( (v_{m-1}, j_{m-1}) = (v, 2) \).
- \( C_m \) is a line (i.e., there are exactly two vertices in \( C_m \) whose out degree is 1) – let \( u \) and \( v \) be the two endpoints of \( C_m \) and let \( i_u \) and \( i_v \) be such that \( f_G(u, i_u) = f_G(v, i_v) = \perp \). Set \( (u_m, i_m) = (u, i_u) \) and \( (v_{m-1}, j_{m-1}) = (v, i_v) \).
• $C_m$ is a cycle (i.e., for every vertex $v \in C$ both $f_G(v,1)$, $f_G(v,2) \neq \perp$) – let $(u,v)$ be the edge with minimal $D$-probability in $C_m$, and let $i_u$ and $i_v$ be such that $f_G(u,i_u) = v$ and $f_G(v,i_v) = u$. Set $(u_m,i_m) = (u,i_u)$ and $(v_{m-1},j_{m-1}) = (v,i_v)$.

Clearly, $L$ indeed connects $f_G$. It is left to show that $D(L) < \epsilon$. Define the following partition of $[k]$; $M_1 = \{m : n_{C_m} < \frac{1}{4}\}$, $M_2 = \{m : n_{C_m} \geq \frac{1}{4}, C_m \text{ is a line}\}$, and $M_3 = \{m : n_{C_m} \geq \frac{4}{\epsilon}, C_m \text{ is a cycle}\}$. As before, we have

$$D(L) = \sum_{m \in M_1} (D(u_m,i_m) + D(v_{m-1},j_{m-1})) + \sum_{m \in M_2 \cup M_3} (D(u_m,i_m) + D(v_{m-1},j_{m-1}))$$

$$\leq SD(f_G) + \sum_{m \in M_2 \cup M_3} (D(u_m,i_m) + D(v_{m-1},j_{m-1}))$$

$$\leq SD(f_G) + S_1(f_G) + \sum_{m \in M_3} (D(u_m,i_m) + D(v_{m-1},j_{m-1})).$$

By the construction, the $D$-probability of an edge selected from a big connected components which is a cycle, is at most the average $D$-probability of an edge in that cycle, which is $\frac{w_{C_m}}{n_{C_m}}$. Since $n_{C_m} \geq \frac{4}{\epsilon}$, then the $D$-probability of this edge is at most $\frac{w_{C_m}}{4\epsilon}$, implying that the third summand in the last equation is at most $\frac{w_{C_m}}{4\epsilon} \leq \frac{4}{\epsilon}$. Summing up, $D(L) \leq SD(G) + S_1(G) + \frac{4}{\epsilon} < \epsilon$.

Next, we show that if in the graph $G$, represented by $f_G$, the total $D$-probability of edges that belong to small connected components is less than $\frac{4}{\epsilon}$, and $S_1(G) \geq \frac{4}{\epsilon}$, then there are three relatively heavy disjoint sets of vertices of degree 1 in $G$. To prove this, we first claim that the total weight of the degree 1 vertices is not concentrated on two vertices.

Lemma 5.1.9 Let $f_G$ be a function $(\epsilon, D)$-far from $P^2$, and let $G$ be the graph represented by it. Let $V^1$ be the set of all vertices in $G$ whose degree is 1. If $S_1(f_G) \geq \frac{4}{\epsilon}$ and $S_D(f_G) < \frac{4}{\epsilon}$, then there are no two vertices $v_1, v_2 \in V^1$ such that $D(v_1) + D(v_2) \geq S_1(f_G) - \frac{4}{\epsilon}$.

The proof of the above lemma is done by showing that if there exist two such vertices, then there exists a function $f' \in P^2$ such that $dist_D(f_G, f') < \epsilon$, contradicting the fact that $f_G$ is $(\epsilon, D)$-far from $P^2$. The construction of the function $f'$ is done by connecting all the connected components of $G$ into a line with $v_1$ and $v_2$ as its endpoints. The full details of the proof appear in Subsection 5.1.3.

Lemma 5.1.10 Let $f_G$ be a function $(\epsilon, D)$-far from $P^2$, and let $G$ be the graph represented by it. Let $V^1$ be the set of all vertices in $G$ whose degree is 1. If $S_1(f_G) \geq \frac{4}{\epsilon}$ and $S_D(f_G) < \frac{4}{\epsilon}$, then there exist three disjoint sets $V_1, V_2, V_3 \subseteq V^1$ such that $\sum_{v \in V_i} D(v) \geq \frac{4}{30}$, for every $1 \leq i \leq 3$.

Proof: By Lemma 5.1.9, there are no two vertices $v_1, v_2 \in V^1$ such that $D(v_1) + D(v_2) \geq S_1(f_G) - \frac{4}{\epsilon}$. We distinguish between the following three cases:

1. There are more than two vertices $v, u \in V^1$ such that $D(v), D(u) \geq \frac{4}{30}$. In this case, set $V_1 = \{v\}$, $V_2 = \{u\}$, and $V_3 = V^1/\{v, u\}$. Based on our assumption, $S_1(f_G) - D(v) - D(u) \geq \frac{4}{\epsilon}$, and therefore, the three sets satisfy the requirements of the lemma.
We now prove that our algorithm is indeed a distribution-free tester for $\mathcal{P}^2$.

**Theorem 5.1.11** Connectivity2 is a distribution-free tester for $\mathcal{P}^2$, with query complexity $O(\epsilon^{-2})$.

**Proof:** Clearly, the query complexity is $O(\epsilon^{-2})$, and if $f_G$ indeed represents a connected graph $G$, then it is accepted by the algorithm with probability 1. We claim that if $f_G$ is $(\epsilon, D)$-far from $\mathcal{P}^2$, then it is rejected by the algorithm with probability of at least $\frac{2}{3}$. Assume that $f_G$ is $(\epsilon, D)$-far from $\mathcal{P}^2$. By Lemma 5.1.8, either $S_D(f_G) \geq \frac{\epsilon}{4}$ or $S_1(f_G) \geq \frac{\epsilon}{4}$. Let us distinguish between the two possibilities:

- If $S_D(f_G) \geq \frac{\epsilon}{4}$, then the probability that after the first stage of the algorithm it fails to detect a small connected component is at most $(1 - \frac{4}{\epsilon})^2 \leq \frac{1}{3}$.

- Otherwise, we know that $S_D(f_G) < \frac{\epsilon}{4}$ and $S_1(f_G) \geq \frac{\epsilon}{4}$. Hence, by Lemma 5.1.10, there are three sets of degree 1 vertices, $V_1$, $V_2$ and $V_3$ such that $\sum_{v \in V_i} D(v) \geq \frac{\epsilon}{30}$ for $1 \leq i \leq 3$. Therefore, the probability that in the second stage, the algorithm fails to sample at least one of these sets is at most $3 \cdot (1 - \frac{30}{\epsilon})^2 \leq 3 \cdot \left(\frac{1}{\epsilon}\right)^4 \leq \frac{1}{3}$.

5.1.3 Proof of Lemma 5.1.9:

We show that if there exist two vertices $v_1, v_2 \in V^1$ such that $D(v_1) + D(v_2) \geq S_1(f_G) - \frac{\epsilon}{5}$, then there exists a function $f' \in \mathcal{P}^2$ such that $\text{dist}(f_G, f') < \epsilon$, contradicting the fact that $f_G$ is $(\epsilon, D)$-far from $\mathcal{P}^2$. This is done by connecting all the connected components of $G$ into a line with $v_1$ and $v_2$ as its endpoints. The construction of the function $f'$ is done by first constructing from $f_G$ a new function $f_G'$, and then constructing a list $L$ that connects $f_G'$. The function $f_G'$ is obtained from $f_G$ in the following manner. Let $C_1, \ldots, C_k$ be the connected components of $G$. We distinguish between two possible cases:

1. If $v_1$ and $v_2$ do not belong to the same connected component, then set $f_G' = f_G$. Assume without loss of generality that $v_1 \in C_1$ and $v_2 \in C_k$, and define $m' = k$.
2. Otherwise, \( v_1 \) and \( v_2 \) belong to the same connected component, \( C_i \). In this case, \( C_i \) is a line having \( v_1 \) and \( v_2 \) as its endpoints. Let \( e' = (u, v) \) be the edge with minimal D-probability in \( C_i \) and let \( i_u \) and \( i_v \) be such that \( f_G(u, i_u) = u \) and \( f_G(u, i_v) = v \). The function \( f_{G'} \) is obtained from \( f_G \) by setting \( f_{G'}(v, i_v) = f_{G'}(u, i_u) = \perp \). In other words, the graph \( G' \) that is represented by \( f_{G'} \) is obtained from \( G \) by removing from \( C_i \) the edge \( e' = (u, v) \) with minimal D-probability. Denote by \( C', C'' \) the two connected components obtained from \( C_i \) by the removal of the edge \( e \). Let \( C_1', \ldots, C_{k+1}'' \) be the connected components of \( f_{G'} \) and, again, assume without loss of generality that \( C' = C_i' \) and \( C'' = C_i'' \). Define \( m' = k + 1 \).

We now construct a list \( L = ((u_1, i_1), (v_1, j_1)), \ldots, ((u_{m'-1}, i_{m'-1}), (v_{m'-1}, j_{m'-1})) \) that connects \( f_{G'} \), in a similar manner to what was done in the proof of Lemma 5.1.8. However, unlike previous constructions, the list \( L \) will be a path from \( C_i' \) to \( C_i'' \) going through all the components rather than a cycle. In the construction, we distinguish between several possible cases for a connected component \( C_m \):

- If \( m = 1 \) then by our assumption \( v_1 \in C_m \) and \( C_m \) is a line. Let \( u, v_1 \) be the endpoints of \( C_1 \), and let \( i_u \) be such that \( f_G(u, i_u) = \perp \). Set \((u_1, i_1) = (u, i_u) \). (Observe that if \( f_G \neq f_{G'} \), then \( u \) is one of the endpoints of the edge \( e' \).)

- If \( m = m' \) then by our assumption \( v_2 \in C_m \) and \( C_m \) is a line. Again, let \( u \) be the other endpoint of \( C_m \) besides \( v_2 \) and let \( i_u \) be such that \( f_G(u, i_u) = \perp \). Set \((v_{m'-1}, i_{m'-1}) = (u, i_u) \), and observe that as before, if \( f_{G'} \) was obtained from \( f_G \) by removing the edge \( e' \), then \( u \) is one of the endpoints of \( e' \).

- If \( 2 \leq m \leq m' - 1 \):
  - \( C_m \) consists of a single vertex \( v - \) set \((u_m, i_m) = (v, 1) \) and \((v_{m-1}, j_{m-1}) = (v, 2) \).
  - \( C_m \) is a line – let \( u \) and \( v \) be the two endpoints of \( C_m \) and let \( i_u \) and \( i_v \) be such that \( f_{G'}(u, i_u) = f_{G'}(v, i_v) = \perp \). Set \((u_m, i_m) = (u, i_u) \) and \((v_{m-1}, j_{m-1}) = (v, i_v) \).
  - \( C_m \) is a cycle – let \((u, v) \) be the edge with minimal D-probability in \( C_m \), and let \( i_u \) and \( i_v \) be such that \( f_{G'}(u, i_u) = v \) and \( f_{G'}(v, i_v) = u \). Set \((u_m, i_m) = (u, i_u) \) and \((v_{m-1}, j_{m-1}) = (v, i_v) \).

Clearly, the list \( L \) indeed connects \( f_{G'} \). Denote by \( f' \) the function obtained from \( f_{G'} \) by adding the edges in \( L \). In addition, observe that if \( f_G \neq f_{G'} \), then both endpoints of \( e' \) appear in the list \( L \). Hence, \( dist_D(f_G, f') \leq D(L) \). It is left to show that \( D(L) < \epsilon \). As before, define \( w_{C_m} = \sum_{v \in C_m} (D(v)) \) and denote by \( n_{C_m} \) the number of vertices in \( C_m \).

Define the following partition of \([2, m'-1] \): \( M_1 = \{ m : n_{C_m} < \frac{4}{\epsilon}, C_m \text{ is a line} \} \), \( M_2 = \{ m : n_{C_m} \geq \frac{4}{\epsilon}, C_m \text{ is a cycle} \} \), and \( M_3 = \{ m : n_{C_m} \geq \frac{4}{\epsilon}, C_m \text{ is a cycle} \} \). We distinguish between the two possible cases.

1. \( f_G = f_{G'} \). In this case,

\[
D(L) = D(u_1, i_1) + D(v_{m'-1}, j_{m'-1}) + \sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \\
+ \sum_{m \in M_2} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))
\]
If $C_1$ is a connected component containing only the vertex $v_1$, then $D(u_1, i_1)$ contributes to $S_D(f_G)$. Otherwise, $u_1$ is a 1-degree vertex other than $v_1$ and $v_2$, and hence $D(u_1, i_1)$ contributes to $S_1(f_G) - (D(v_1) + D(v_2))$. The same holds for $D(v_{m'-1}, j_{m'-1})$. Therefore,

$$D(L) \leq S_D(f_G) + S_1(f_G) - (D(v_1) + D(v_2)) + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

$$\leq S_D(f_G) + \frac{\epsilon}{8} + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})).$$

As before, the $D$-probability of an edge selected from a big connected component which is a cycle, is at most the average $D$-probability of an edge in that cycle, which is $\frac{w_{C_m}}{n_{C_m}}$. Since $n_{C_m} \geq \frac{4}{\epsilon}$, then the $D$-probability of this edge is at most $\frac{w_{C_m} \cdot \epsilon}{4}$, implying that the third summand in the last equation is at most $\sum_{m \in M_3} \frac{w_{C_m} \cdot \epsilon}{4} \leq \frac{\epsilon}{4}$. Summing up, $D(L) \leq S_D(G) + \frac{\epsilon}{8} + \frac{\epsilon}{4} < \epsilon$.

2. $f_G'$ is obtained from $f_G$ by removing the edge $e'$. In this case,

$$D(L) = D(e') + \sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) + \sum_{m \in M_2 \cup M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})).$$

Let $C_i$ be the connected component of $f_G$ that both $v_1$ and $v_2$ belong to. Consider two possibilities.

(a) If $n_{C_i} \leq \frac{4}{\epsilon}$, then

$$D(L) = D(e') + \sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

$$+ \sum_{m \in M_2 \cup M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

$$\leq S_D(f_G) + \sum_{m \in M_2 \cup M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

$$\leq S_D(f_G) + S_1(f_G) - (D(v_1) + D(v_2)) + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$$

$$\leq S_D(f_G) + \frac{\epsilon}{8} + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})).$$

As before, $\sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1}))$ can be bounded by $\frac{\epsilon}{4}$, implying that $D(L) < \epsilon$.

(b) Otherwise, $n_{C_i} \geq \frac{4}{\epsilon}$. In this case, $D(e') \leq \frac{w_{C_i}}{n_{C_i} - 1} \leq \frac{w_{C_i}}{3}$ (since $n_{C_i} - 1 \geq \frac{3}{\epsilon}$).
Therefore, using the same arguments as before,

\[ D(L) = D(e') + \sum_{m \in M_1} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \]

\[ + \sum_{m \in M_2} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \]

\[ \leq SD(f_G) + S_1(f_G) - (D(v_1) + D(v_2)) + D(e') + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \]

\[ < \frac{\epsilon}{4} + \frac{\epsilon}{8} + D(e') + \sum_{m \in M_3} (D(u_m, i_m) + D(v_{m-1}, j_{m-1})) \]

\[ \leq \frac{\epsilon}{4} + \frac{\epsilon}{8} + \frac{\epsilon}{3}. \]

Where the last term is due to \( e' \) and to the edges removed from connected components \( C_m \) for \( m \in M_3 \). Implying that indeed \( D(L) < \epsilon \) as required.

\[ \square \]

### 5.2 Alternative testing models for sparse graphs

In this section, we consider additional testing models for sparse graphs, as discussed in the Introduction. For each of these testing models, we define the class of functions that represent graphs, the queries that the tester is allowed to ask, and prove the existence of a distribution-free tester in that model. Section 5.2.1 deals with graphs with a bounded number of edges. Then, Section 5.2.2 deals with bounded-degree graphs with a bound also on the total number of edges.

#### 5.2.1 The edge-bounded model

The edge-bounded testing model described in this section is a generalization of the model presented in [43] for testing with respect to the uniform distribution: There is no upper bound on the degree, only a bound \( m \) on the total number of edges in the graph; as in the uniform model, the tester can query the degree of a given vertex \( v \) or the \( i' \)th neighbor of \( v \). Equivalently, we require that the list of edges is of length at most \( m \). In the uniform variant of this model, we are interested in distinguishing between the case that a graph that contains at most \( m \) edges is connected, and the case that it is far from any connected graph containing at most \( m \) edges. As explained in [43], connectivity can be tested with respect to the uniform distribution.

To generalize this model to the distribution-free setting, we view a graph \( G = (V, E) \) as a function \( f : [m] \to \{0\} \cup \{\perp\} \), indicating for each \( i \in [m] \) the \( i' \)th entry in the edge list if such an edge exists, or \( \perp \) if that entry is empty. As before, a graph \( G \) can have more than one representation, due to different orderings of the edge list. Note that the differences between the weights of the entries in the list are more significant when removing edges (in order to make room for new edges) and less relevant when talking about edges additions (where one can choose the entry with minimal probability).

Denote by \( P_m \) the class of functions that represent connected graphs with at most \( m \) edges. The next lemma, shows that if \( m \geq (1 + c)n \) for some constant \( c \), then it is possible to use
any tester known in this model for the uniform distribution, to construct a distribution-free tester.

**Lemma 5.2.1** Let \( m \geq (1+c)n \), for some constant \( c \). Then, there exists a distribution-free tester for \( \mathcal{P}_m \) with query complexity of \( O(\epsilon^{-3}) \).

**Proof:** We show that a distribution-free tester for \( \mathcal{P}_m \) can be constructed based on the existence of a uniform one, with the same query complexity up to a constant factor. To do so, we prove that if \( f_G \) is \( \epsilon \)-close to \( \mathcal{P}_m \) with respect to the uniform distribution, then \( f_G \) is \( g(\epsilon) \)-close to \( \mathcal{P}_m \) with respect to any distribution \( D \), when \( g(\epsilon) \) is a linear function in \( \epsilon \). It follows that, if a function \( f_G \) is \( (\epsilon, D) \)-far from \( \mathcal{P}_m \) for some distribution \( D \), it is also \( g^{-1}(\epsilon) \)-far from \( \mathcal{P}_m \) with respect to the uniform distribution. Thus, it is possible to test whether a given function \( f_G \) is in \( \mathcal{P}_m \) or is \( (\epsilon, D) \)-far from \( \mathcal{P}_m \), for some fixed but unknown distribution \( D \), using the known uniform tester [43] whose query complexity is \( O(\epsilon^{-3}) \), it follows that the query complexity of the distribution-free tester is as required.

Assume from now on that \( f_G \) is indeed \( \epsilon \)-close to \( \mathcal{P}_m \), with respect to the uniform distribution, and let \( G \) be the graph it represents. There exists a set \( X \) that contains at most \( cm \) edges, such that adding those edges to the graph \( G \) transforms it into a connected one (though the \( cm \) modifications required to transform \( G \) into a connected graph may include edge removals as well, these modifications contribute nothing to the connectivity and are only done to prevent exceeding the bound on the number of edges). In addition, there exists a set \( Z \subseteq E \), of size at most \( n-1 \), that spans each of the connected components of \( G \). Thus, removal of any edge that does not belong to \( Z \), has no effect on the connected components of the graph. In other words, there are at least \( m-n \) values \( 1 \leq i \leq m \), such that a change of \( f_G(i) \) does not affect the connected components of the graph. Denote this set of values by \( Y \). By assigning the edges in \( X \) to the values in \( Y \), we transform \( G \) to a connected graph. That is, the function obtained from \( f_G \) after the additions, belongs to \( \mathcal{P}_m \). Let \( D \) be a distribution over \([m]\), and let \( X' \) denote the \( cm \) values in \( Y \) with minimal probability according to \( D \) (assume for now that \( cm \leq m-n \)). The total probability of \( X' \) is the minimal possible cost for such an addition. This probability is at most \( \frac{cm}{m-n} = \frac{\epsilon}{1 - \frac{1}{c+1}} \), and since \( m \geq (1+c)n \) it is at most \( \frac{\epsilon}{1 - \frac{1}{c+1}} \). To summarize, if \( dist_U(f_G, \mathcal{P}_m) \leq \epsilon \), then \( dist_D(f_G, \mathcal{P}_m) \leq \frac{\epsilon}{1 - \frac{1}{c+1}} \) for any distribution \( D \). Hence, if \( dist_D(f_G, \mathcal{P}_m) \geq \epsilon \) for some distribution \( D \), then \( dist_U(f_G, \mathcal{P}_m) \geq \epsilon \cdot (1 - \frac{1}{1+c}) \).

It remains to show that, for every possible distance parameter \( \epsilon' \) that we will run the uniform tester with, indeed \( \epsilon' m \leq m-n \). It is easy to verify that since \( \epsilon \leq 1 \) and \( m \geq n(1+c) \), then \( \epsilon' m = \epsilon(1 - \frac{1}{1+c})m \leq m-n \), enabling the choice of \( \epsilon' m \) edges out of \( X' \). This concludes our proof. \( \square \)

### 5.2.2 The combined model

Next, we consider a combination of the two previous testing models. One may think of graphs where there are at most \( m \) edges and the degree of each vertex is at most \( d \). This model can be seen as an intermediate model between the above two models, since it contains all bounded-degree graphs and is contained in the set of all graphs with a bound on the total number of edges. As in the previous model, graphs are represented by functions \( f : [m] \rightarrow \{\emptyset\} \cup \{\bot\} \), and the distribution is defined over the list of edges, i.e. the set \([m]\). However, in this case we are only interested in functions that represent degree-\( d \) graphs with at most \( m \) edges. Denote

\[ \epsilon' = \epsilon(1 - \frac{1}{1+c}) \leq \epsilon \leq (m-n) \leq (m-n). \]
the class of these functions by $\mathcal{P}_m^d$. As in the previous model, the tester is able to query the $i'\text{th}$ neighbor of a given vertex $v$.

Though we believe that this is a very natural testing model for sparse graphs, it was not considered before even in the uniform setting. We present both a uniform and a distribution-free tester for $\mathcal{P}_m^d$, using similar approaches. When dealing with the uniform distribution, the distance between two functions is measured as in [43]; that is, by the fraction of edge modifications necessary to transform the graph into a connected one, measured with respect to $m$.

**Lemma 5.2.2** There exists a uniform tester for $\mathcal{P}_m^d$ with query complexity $\text{poly}(\frac{1}{d})$.

**Proof:** To prove the existence of such a tester, we show that if a function $f_G$ is $\epsilon$-close to $\mathcal{P}_m$, then it is $3\epsilon$-close to $\mathcal{P}_m^d$ (note that both testing models use the same graph representation).

It follows, that if $f_G$ is $\epsilon$-far from $\mathcal{P}_m^d$, it is $\frac{3}{5}\epsilon$-far from $\mathcal{P}_m$. Hence, the existence of a uniform tester for $\mathcal{P}_m^d$ follows the existence of such a tester for $\mathcal{P}_m$ [43]. This tester works by performing BFS walks in the graph, and thus does not use queries of a vertex degree. Therefore, testing in this model can be performed with the same query complexity $\text{poly}(\frac{1}{d})$.

Assume that $f_G$ is $\epsilon$-close to $\mathcal{P}_m$, and let $G$ be the graph represented by $f_G$. Then, there exists a set of $\epsilon m$ edge modifications that transforms $G$ into a connected graph with at most $m$ edges. Denote this set by $X$. Note that this set of edge modifications $X$ may not respect the degree bound $d$. Therefore, we find, based on these edge modifications, a way to transform $G$ into a connected graph while keeping both the bound on the degree and the bound on the number of edges. Denote the set of edge removals in $X$ by $X_{\text{remove}}$, and the set of edge additions in $X$ by $X_{\text{add}}$. Assume that, for every connected component in $G$, there exist at most two edges in $X_{\text{add}}$ such that one of the vertices in the edge belongs to this component (otherwise, it is possible to construct such an $X$ with the same size). In addition, assume that all the edges in $X_{\text{remove}}$ are redundant in the connected component they belong to (otherwise, we can construct a set $X$ of the same size that does not contain these removals; this is since for every edge in $X_{\text{remove}}$ that disconnects a connected component in the graph, an additional edge has to be added in order to connect the graph). It is clear that removing the edges in $X_{\text{remove}}$ can be done without exceeding the degree bound. Thus, we are left with the addition of the edges in $X_{\text{add}}$. We will add the edges to the graph iteratively, showing that in every stage at most 3 edge modifications are needed to enable the addition of an edge in $X_{\text{add}}$, without violating the degree bound. Let $e = (u, v)$ be an edge in $X_{\text{add}}$ that we wish to add to the graph. Clearly, adding the edge $(u, v)$ is equivalent to adding an edge $(u', v')$ such that $u$ and $u'$, and similarly $v$ and $v'$, are in the same connected component. Note that since all the edges in $X_{\text{remove}}$ were redundant in their components, the connected components of the graph obtained from $G$ after the removals are identical. If there exist vertices $u'$ and $v'$ in the connected components of $u$ and $v$, respectively, with degree less than $d$, then we can add the edge $(u', v')$ at a cost of one edge modification. Otherwise, assume that there is no vertex $u'$ in the connected component of $u$ with degree less than $d$. Hence, all the vertices in the component of $u$ have a degree of $d$, and we can deduce that none of the edges in $X_{\text{remove}}$ belonged in the same connected component with $u$. Assume there are $k$ vertices in that component. Hence, there are $\frac{k+1}{2}$ edges in this component, and only $k - 1$ of them are necessary to span the component. Therefore, there is at least one edge that can be removed.

---

5$\mathcal{P}_m^d = \mathcal{P}_m^d \subseteq \mathcal{P}_m$.  
6In case that the tester performs queries of vertices degrees, these queries can be simulated in this model using $O(d)$ queries (in the worst case), causing a factor of $d$ in the query complexity.
without effecting the connectivity, resulting with a vertex with degree less than \( d \). The same can be done in case there is no vertex \( v' \) in the connected component of \( v \) with degree less than \( d \). We observe that since at most two edges with endpoints in a specific connected component have to be added, then at most one edge has to be removed from every connected component in \( G \). To summarize, the addition of the edge \((u,v)\) (or an equivalent edge) costs at most 3 edge modifications, implying the desired claim.

We now turn to the distribution-free setting.

**Theorem 5.2.3** Let \( m \geq (1+c)n \), for some constant \( c \). There exists a distribution-free tester for \( P_m^d \) with query complexity \( \text{poly}(\frac{1}{\varepsilon}, d) \).

To construct the distribution-free tester, we want to use the tester presented for bounded-degree graphs in Section 5.1. Hence, we first need a way to transform graphs in our representation to the bounded-degree representation. We describe how, given a function \( f_G : [m] \to \binom{[d]}{2} \cup \{\perp\} \) that represents a graph \( G \) with degree of at most \( d \), we transform it to a function \( T(f_G) : V \times [d] \to V \cup \{\perp\} \) that represents the same graph \( G \), as follows: for every \( v \in V \), assume an ordering on the edges outgoing from \( v \) (i.e., the same ordering according to which the tester’s queries are answered). For every \( i \in [d] \), define the value of \( T(f_G)(v,i) = u \) if \( u \) is the \( i \)'th neighbor of \( v \) and \( \perp \) if \( v \) has less than \( i \) neighbors. Note that necessarily if \( T(f_G)(v,i) = \perp \), then \( T(f_G)(v,i+1) = \perp \).

Note that, given access to queries for the function \( f_G \), one can simulate queries for the function \( T(f_G) \) (this is due to the choice of the same ordering on the outgoing edges of a vertex \( v \)). However, this is not enough to use the distribution-free tester for bounded-degree graphs. We also need to find a way to translate distribution measures from our model to the bounded-degree model.

Given the functions \( f_G \) and \( T(f_G) \), we describe a transformation from the distribution \( D \) defined over \([m]\) to a distribution \( T(D, f_G) \) defined over \( V \times [d] \). Let \( W \) be the total probability of all non-empty entries in \([m]\) under the distribution \( D \). That is, \( W = \sum_i 1_{f(i) \neq \perp} D(i) \). The distribution \( T(D, f_G) \) is now defined as follows. For every \( v \in V \) and \( i \in [d] \):

- If \( T(f_G)(v,i) = u \) and \((v,u)\) is the \( j \)'th edge in the list, define \( T(D, f_G)(v,i) = D(j) / 2W \). If \( W = 0 \), then \( T(D, f_G) \) is defined to be the uniform distribution over all non-empty entries in \( V \times [d] \).
- Otherwise, define \( T(D, f_G)(v,i) = 0 \).

Note that, sampling of \( V \times [d] \) with respect to the distribution \( T(D, f_G) \) using sampling of \([m]\) according to the distribution \( D \) is not possible. This is due to the possibility to sample empty entries in \([m]\), that cannot be translated to a pair \((v,i)\) in \( V \times [d] \).

**Sampling of \( T(D, f_G) \):** Set \( c = 8 \).

- Sample, at most \( \frac{8c}{\varepsilon} \log \frac{1}{\varepsilon} \) times, the distribution \( D \) until getting \( i \) such that \( f(i) \neq \perp \).
- If all \( \frac{8c}{\varepsilon} \log \frac{1}{\varepsilon} \) samples were of empty entries, return FAIL.
- Otherwise, let \( i \) be the sampled non-empty entry, and assume \( f(i) = (v,u) \). Find the location of \( u \) in \( v \)'s neighbors list, denote it by \( i_v \). Similarly, find \( v \)'s place in \( u \)'s neighbors list, denoted by \( i_u \). Then, with probability \( \frac{1}{2} \) return \((v,i_v)\), otherwise return \((u,i_u)\).
Each sampling of $T(D, f_G)$, using the above sampling procedure, requires $poly(\frac{1}{\varepsilon}, d)$ queries. In addition, if the sampling procedure indeed returns a value $(v, i)$, then $(v, i) \sim T(D, f_G)$. However, there is also the possibility that the sampling procedure returns FAIL, which does not happen when given the possibility of direct sampling of $T(D, f_G)$. Let $A$ be a connectivity distribution-free testing algorithm in the bounded-degree model. Denote by $A'$ the algorithm obtained from $A$ by adding the following rule: if any sampling query of $A$, during its run on a function $f$, was answered FAIL, then $A$ accepts $f$. We will show that we can still use the tester after this modification for our construction.

Assume for now, that we actually have an ability to sample $T(D, f_G)$. One may wonder whether the distribution-free tester for $P^d$, that was presented in Section 5.1, can be used as a tester for $P^d_m$ as well. It is easy to see, however, that in some cases the function $T(f_G)$ is very close to $P^d$ with respect to $T(D, f_G)$, while $f_G$ is far from $P^d_m$ with respect to $D$. Therefore, a different test is required. The distribution-free tester for $P^d_m$ appears in Figure 5.3.

Algorithm Connectivity$_m(\varepsilon, d)$:
Repeat twice:
- Run connectivity'(\varepsilon, d) (obtained from the algorithm presented in Section 5.1) on $T(f_G)$ using the sampling procedure for $T(D, f_G)$ with distance parameter $\frac{\varepsilon}{2}$.
- Run the distribution-free tester for $P_m$ on $f_G$ and $D$ with distance parameter $\frac{\varepsilon}{2}$.
return PASS

Figure 5.3: Distribution-free tester for $P^d_m$

Theorem 5.2.3 now follows immediately the correctness of the above tester. This, in turn, is based on Lemma 5.2.4 and Lemma 5.2.5 below, and the existence of the distribution-free testers for $P_m$ (Section 5.2.1). Assume that $d \geq 3$; the proof for $d = 2$ is similar.

As in Section 5.1, we denote by $S_D(f)$ the total weight of the small connected components in $G$. That is, the total weight of connected components of size less than $\frac{6\varepsilon}{d^2}$. Lemma 5.2.4 below shows that if the total weight of small connected components in $G$, measured with respect to $D$, is large then, with high probability (over the possible executions of the tester), the function $f_G$ fails to pass the first stage of the above tester. Lemma 5.2.5 then shows that if the total weight of small connected components in $G$ is small, and the function is far from $P^d_m$ then, with high probability over the possible executions of the tester, it fails to pass the second stage of the above tester. Therefore, if a function $f_G$ is $(\varepsilon, D)$-far from $P^d_m$ then, with high probability, it is rejected by the tester.

Lemma 5.2.4 Let $f_G : [m] \rightarrow (\frac{1}{2}) \cup \{\bot\}$ be a function and $D$ a distribution over $[m]$. If $S_D(f_G) > \frac{\varepsilon}{7}$, then the probability that $f_G$ passes the first stage of the tester is at most $\frac{2}{5}$.

The proof of this lemma uses the following observation, which is based on the analysis of the tester for $P^d$ (Section 5.1).

Observation 3: Let $f_G$ be a function in the bounded-degree model. If $S_D(f_G) > \frac{\varepsilon}{7}$, then the probability that $f$ passes the distribution-free test presented in Section 5.1, with respect to a distribution $D$ over $V \times [d]$ with distance parameter $\varepsilon$, is at most $\frac{1}{3}$.

---

7Let $m = nd/4$, let $D$ be the uniform distribution over $[m]$, and consider the following graph $G$ on $n$ nodes. $G$ consists of $n/2$ isolated vertices and a connected component of size $n/2$ that contains $nd/4$ edges. It is easy to see that $dist_{P_m}(f_G, P^d_m) \geq 1/d$, while $dist_{T(D, f_G)}(T(f_G), P^d) = 4/nd$.

8For the case $d = 2$, we define $S_D(f_G)$ to be the total weight of connected components of size at most $\frac{4}{7}$, as in Section 5.1.
Proof of Lemma 5.2.4: Let \( f : [m] \rightarrow (\{1\} \cup \{\bot\}) \) be a function and \( D \) be a distribution over \([m]\), and assume that \( S_D(f_G) > \frac{\epsilon}{4} \). If \( f_G \) passed the first stage of the tester, then one of the following two events occurred. Either the function \( T(f_G) \) passed both runs of the bounded degree tester, or one of the tester sampling queries was answered by \text{FAIL}. Since the probability of any connected component, with respect to the distribution \( T(D, f_G) \), is at least as big as its probability with respect to \( D \), we deduce that \( S_{T(D, f_G)}(T(f_G)) > \frac{\epsilon}{4} \). Therefore, by Observation 3, the probability of the first event (i.e., that \( T(f_G) \) passed the test in both times), is at most \( \frac{1}{3} = \frac{1}{9} \). It remains to show that the probability of the second event is also at most \( \frac{1}{9} \). Since \( S_D(f_G) > \frac{\epsilon}{4} \), we have \( W > \frac{\epsilon}{4} \). Hence, the probability that an execution of the sampling procedure fails to sample a non-empty entry in \([m]\) is at most \( (1 - \frac{\epsilon}{4})^{\frac{9}{4}} \frac{1}{2} \log \frac{1}{e} \leq \frac{1}{e^{c_{\epsilon}}} \). Since in both executions of the bounded degree tester there are at most \( \frac{\epsilon}{2} \) sampling queries then, using the union bound, the probability that at least one of these queries was answered by \text{FAIL} is at most \( \frac{\epsilon}{e^{c_{\epsilon}}} \frac{1}{2} \leq \frac{\epsilon}{e^{c_{\epsilon}}} \). 

Lemma 5.2.5 Let \( f : [m] \rightarrow (\{1\} \cup \{\bot\}) \) be a function and \( D \) be a distribution over \([m]\). If \( f \) is \( \frac{\epsilon}{2} \)-close to \( P_m \), and \( S_D(f_G) \leq \frac{\epsilon}{4} \), then \( f \) is \( \epsilon \)-close to \( P_m^d \), where both distances are measured with respect to \( D \).

Proof: Since \( f \) is \( \frac{\epsilon}{2} \)-close to \( P_m \) with respect to the distribution \( D \), there exists a set of edge modifications of total weight of at most \( \frac{\epsilon}{4} \) that transforms \( G \) into a connected graph with at most \( m \) edges. Denote this set by \( X \). As in the case of the uniform tester, we are interested in performing these modifications in the graph \( G \) while respecting the degree bound \( d \). Let \( X_{\text{remove}} \) and \( X_{\text{add}} \) be as in the proof of Lemma 5.2.2. As before, the modifications in \( X_{\text{remove}} \) can be performed without exceeding the vertices degrees and we can assume that all the edges in \( X_{\text{remove}} \) are redundant in their connected component. Hence, the connected components of the graph obtained from \( G \) after the removal of the edges in \( X_{\text{remove}} \) are still identical. However, the modifications in \( X_{\text{add}} \) have to be handled more carefully since besides the need to avoid exceeding the degree bound, we have to make sure that the total probability of the modifications is small. Again, adding an edge \((u, v)\) to the graph is equivalent to adding every edge \((u', v')\) for \( u' \) a vertex in the same connected component as \( u \) and \( v' \) a vertex in the same connected component as \( v \). In case the addition of the edge is possible without removing any edges, we proceed as in the proof of Lemma 5.2.2, and add the edge with the same cost as in \( X_{\text{add}} \). Otherwise, assume that there is no vertex in the connected component of \( u \) with degree less than \( d \). Hence, we can deduce that none of the edges in \( X_{\text{remove}} \) belong to this connected component. If the size of the connected component of \( u \) is \( k \), then there are \( \frac{kd}{4} \) edges in that component. However, in the case of testing with respect to an arbitrary distribution measure, we will not be able to remove any edge that in not necessary for the connectivity as before, and the removal has to be done while considering the edges probabilities. Since \( d \geq 3 \), there exist at least \( \frac{kd}{6} \) edges in the component that are not essential for connectivity. We remove the edge with the minimal probability out of these edges.

It is left to prove that indeed the total probability of the edge modifications made in \( G \) is at most \( \epsilon \). This probability is the sum of the total probability of the edges modifications in \( X \), which is \( \frac{\epsilon}{2} \), and the total cost of the edge removals made to allow the addition of the edges in \( X_{\text{add}} \). Using arguments similar to the ones used in Section 5.1, we can show that the total probability of the edges removals is at most \( \frac{\epsilon}{4} + \frac{6\epsilon}{96} \), where the first term is due to the small connected components, with size less than \( \frac{96\epsilon}{3d} \), and the second term is due to the big connected components. Hence, the total probability of edge modifications needed to transform \( G \) to a connected graph with respect to \( D \) is at most \( \epsilon \).
Remark 5.2.1: The analysis for the case $d = 2$ is similar. Note that Observation 3 holds also for the case $d = 2$, and hence so does the proof of Lemma 5.2.4. As for the proof of Lemma 5.2.5, the problem of being unable to add an edge from $X_{add}$ can only arise in a connected component which is a cycle. In such case, we remove the edge with minimal $D$-probability and proceed as in the case of $d \geq 3$. 
Chapter 6

Distribution-Free Monotonicity Testing on the d-Dimensional Cube

In this chapter, we present testers for monotonicity over the d-dimensional hyper-cube with respect to an arbitrary distribution D. For simplicity, we begin our discussion with the case d = 1, and show that given access to random samples according to D and to membership queries, there is a distribution-free tester for monotonicity over [n], whose query complexity is $O(\frac{\log n}{\epsilon})$. In Section 6.3, we generalize this algorithm to a distribution-free tester for monotonicity over the d-dimensional hyper-cube whose query complexity is $O\left( (\log n)^d \cdot 2^d \frac{1}{\epsilon} \right)$ (this algorithm for the case $d = 1$ is the basic algorithm shown in Section 6.2).

6.1 Preliminaries

We begin with a few notations and definitions. Denote by $[n]$ the set $\{1, \ldots, n\}$, and by $[n]^d$ the set of d-tuples over $[n]$. For every two points $\vec{i}$ and $\vec{j}$ in $[n]^d$ we say that $\vec{i} \leq \vec{j}$ if for every $1 \leq k \leq d$, $i_k \leq j_k$. Let $(A, \prec_A)$ be some linear order.

**Definition 6.1.1** We say that a function $f : [n]^d \rightarrow A$ is monotone if for every $\vec{i}$ and $\vec{j}$ if $\vec{i} \leq \vec{j}$ then $f(\vec{i}) \leq_A f(\vec{j})$.

**Definition 6.1.2** Let $f : [n]^d \rightarrow A$ be a function. A pair $(\vec{i}, \vec{j})$ is said to be an $f$-violation if $\vec{i} \prec_A \vec{j}$ and $f(\vec{i}) >_A f(\vec{j})$.

Let $D$ be any distribution on $[n]^d$, and let $S$ be a subset of $[n]^d$. Define $Pr_D\{\vec{i}\} \overset{\text{def}}{=} Pr_{X \sim D}\{X = \vec{i}\}$, and $Pr_D\{S\} \overset{\text{def}}{=} \sum_{\vec{i} \in S} Pr_D\{\vec{i}\}$.

6.2 Testing monotonicity for the line ($d = 1$)

In this part we consider the case $d = 1$. Our algorithm is a variant of the algorithm presented in [20] for testing monotonicity (with respect to the uniform distribution). The analysis presented here for this algorithm, however, is quite different. The algorithm works in phases, in each phase a center point is selected according to the distribution $D$ (in the original algorithm, the center point is selected uniformly), and the algorithm looks for a violation of the monotonicity with this center point. The search for a violation is done by randomly sampling in growing
neighborhoods of the center point. In other words, in the case \( d = 1 \), the only change made in the original algorithm in order to adjust it to be distribution-free is that the choice of center points is made according to \( D \). However, the search for violations remains unchanged. It is important to observe that, when dealing with an arbitrary distribution, there is no connection between the distance of the function from monotone (or the probability of the violation) and the number of pairs that form a violation of monotonicity. Hence, the correctness of the algorithm for the uniform distribution does not imply its correctness for the general case.

As stated above, we present a tester for monotonicity in the one-dimensional case with query complexity \( O\left(\frac{\log n}{\epsilon}\right) \). The algorithm appears in Figure 6.1.

![Algorithm-monotone-1-dimD(f, \epsilon)](algorithm-monotone-1-dimD(f, \epsilon):
repeat \( \frac{n}{2} \) times
  choose \( i \in D \subseteq [n] \)
  for \( k \leftarrow 0 \ldots \left\lfloor \log i \right\rfloor \) do
    repeat 8 times
      choose \( a \in R \subseteq [2^k] \)
      if \( f(i-a) >_A f(i) \) then return FAIL
  for \( k \leftarrow 0 \ldots \left\lfloor \log(n-i) \right\rfloor \) do
    repeat 8 times
      choose \( a \in R \subseteq [2^k] \)
      if \( f(i) >_A f(i+a) \) then return FAIL
return PASS

Figure 6.1: Distribution-free monotonicity tester for \( d = 1 \)

**Theorem 6.2.1** Algorithm \( \text{monotone-1-dim}_D \) is a one-sided error distribution-free monotonicity tester over the line.

To prove this theorem, we need the following definitions and lemmas.

**Lemma 6.2.2** Let \( f : [n] \rightarrow A \) be a function, and let \( S \subseteq [n] \) be a set. If for every \( f \)-violation \((i, j)\) either \( i \in S \) or \( j \in S \), then there exists a monotone function \( f' \) that differs from \( f \) only on points in \( S \).

**Proof:** We will show that, by modifying the value of \( f \) only on points in \( S \), we can obtain a monotone function \( f' \). Define \( \bar{S} = [n] \setminus S \). The construction of \( f' \) is done through the following iterative process. In each step, modify the value of the smallest point in \( S \) that was not yet modified. The process ends after redefining the values of all the points in \( S \).

The value of point \( i \in S \) is set in the following manner. We distinguish between two cases: \( i = 1 \) and \( i \in S \setminus \{1\} \). If \( i = 1 \) then we set \( f'(i) = f(i) \) if \( S = [n] \), and \( f'(i) = \min_{j \geq 1 \ldots j \in S \setminus \{f(j)\}} \) (note that the maximum is taken over all points before \( i \) and not only those in \( S \)). The values of points in \( \bar{S} \) are unchanged (i.e., for all these points \( f'(i) = f(i) \)).

It is left to prove that \( f' \) is indeed monotone, or equivalently that there are no \( f \)-violations. Consider a pair of points \( i, j \) s.t. \( i < j \); we will prove that \( f'(i) \leq_A f'(j) \). There are three possibilities:

**Case 1:** \( i, j \in \bar{S} \) — since \( f' \) equals \( f \) for all points not in \( S \), and there were no \( f \)-violations with both endpoints in \( \bar{S} \), then \( f'(i) = f(i) \leq_A f'(j) = f'(j) \).
The fact that every monotone function from monotone then $P_{D}$ immediately from its definition. From now on, assume that there are no $f'$-violations other than the ones referred to in this case, this is the minimal $f'$-violation. By the definition of $f'$, clearly if $i = 1$ then $f'(i) = \min_{j > 1, j \in S} \{f(j)\}$ and therefore $f'(i) \leq A f(j) = f'(j)$. Hence $i \neq 1$. By the construction of $f'$, the value of $f'$ at point $i$ is the maximal value of $f'$ at points $k$ s.t. $k < i$. Thus $f'(i) = f'(k)$, for some $k < i$, implying that $(k, j)$ is also an $f'$-violation, contradicting the minimality of $(i, j)$.

A similar argument was used in [19]. An immediate conclusion of the above lemma is the following:

**Lemma 6.2.3** Let $f: [n] \to A$ be a function $(\varepsilon, D)$-far from monotone. Given $S \subseteq [n]$, if for every $f$-violation $(i, j)$ either $i \in S$ or $j \in S$, then $P_{D}\{S\} \geq \varepsilon$.

**Definition 6.2.4** For an $f$-violation $(i, j)$, we say that $i$ is active in this violation if

$$|\{k : i < k < j, f(i) > A f(k)\}| \geq \frac{j - i - 1}{2},$$

similarly, $j$ is active in this violation if $|\{k : i < k < j, f(j) < A f(k)\}| \geq \frac{j - i - 1}{2}$.

That is, $i$ is active in an $f$-violation $(i, j)$, if for at least half of the points $i < k < j$, $(i, k)$ is also an $f$-violation (i.e., $f(i) > A f(k)$).

**Observation 4:** For every $f$-violation $(i, j)$, at least one of $i$ and $j$ is active in $(i, j)$.

**Proof:** Let $(i, j)$ be an $f$-violation. It is known that $i < j$ and $f(i) > A f(j)$. For every $i < k < j$, if $f(i) \leq A f(k)$ (i.e., $(i, k)$ is not an $f$-violation) then $f(k) > A f(j)$, and hence $(k, j)$ is an $f$-violation. Similarly, if $(k, j)$ is not an $f$-violation, then $(i, k)$ is. Therefore, each of the points between $i$ and $j$ forms an $f$-violation with either $i$ or $j$.

Define the active set of $f$ (denoted $A_f$) as the set of all points that are active in some $f$-violation. Following this observation and applying Lemma 6.2.3 to the set $A_f$, if $f$ is $(\varepsilon, D)$-far from monotone then $P_{D}\{A_f\} \geq \varepsilon$. We now prove Theorem 6.2.1.

**Proof:** The fact that every monotone function $f$ is accepted by the algorithm follows immediately from its definition. From now on, assume that $f$ is $(\varepsilon, D)$-far from monotone; we prove that $f$ is rejected with probability of at least $\frac{\varepsilon}{2}$. Our algorithm may fail to detect that $f$ is not monotone if either one of the following two events occurs:

1. None of the points sampled by the algorithm according to $D$ is in $A_f$.
2. The algorithm picked at least one point $i \in A_f$, but failed to detect that $i$ belongs to some $f$-violation.

It is easily verified that the probability of the first event is at most $(1 - \varepsilon)^3 \leq \frac{1}{2^3} \leq 1/6$. We now bound the probability of the second event. By the definition of $A_f$, for every $i \in A_f$ there is a $j$ such that either $(i, j)$ or $(j, i)$ is an $f$-violation and $i$ is active in this violation. Assume w.l.o.g. that $(i, j)$ is an $f$-violation. For $k = \min\{l : 2^l \geq j - i\}$ (i.e., $k$ is the smallest integer s.t. $j \leq i + 2^k$), we can claim that $|\{l : i < l \leq i + 2^k, f(i) > A f(l)\}|$ is more than $\frac{1}{2} : 2^k$. This is due to the fact that $j - i > 2^{k-1}$, and since $i$ is active in the $f$-violation $(i, j)$, for at least half the points $l$ between $i$ and $j$ (i.e., at least $\frac{2^{k-1}}{2}$ points) the pair $(i, l)$ is
an $f$-violation. The probability that the algorithm fails to find an $f$-violation for this $k$ is at most $(\frac{3}{4})^8 \leq \frac{1}{8}$, and hence the probability of the second event is at most $\frac{1}{8}$, implying that the total probability that the algorithm wrongly accepts $f$ is at most $\frac{1}{3}$.

**Remark 6.2.1:** In the journal version of [20], an additional testing algorithm for monotonicity on the line, called “Sort-Check-I”, is presented. This algorithm can also be transformed to be a distribution-free monotonicity tester over the line. However, we do not know if it can be generalized to higher dimensions.

### 6.3 Distribution-free monotonicity testing for general $d$

In the previous subsection we saw how to test monotonicity over the one-dimensional hypercube (the line) when the distance is measured with respect to an arbitrary distribution.

We show how to generalize this algorithm to the $d$-dimensional case. We begin by adjusting the definitions and notations of the previous subsection to the $d$-dimensional case. Notice that unlike the one-dimensional case, in the $d$-dimensional case $[n]^d$ is a partial order (not every two points are comparable). Denote by $\vec{i}$ the point $i_1, \ldots, i_d \in [n]^d$. For every two points $\vec{i}$ and $\vec{a}$, we denote by $\vec{i} + \vec{a}$ (respectively $\vec{i} - \vec{a}$) the point $i_1 + a_1, \ldots, i_d + a_d$ (respectively $i_1 - a_1, \ldots, i_d - a_d$), where every coordinate smaller than 1 is set to 1 and every coordinate larger than $n$ is set to $n$.

We will show that given access to random samples according to $D$ and to membership queries, there is a distribution-free tester for monotonicity, whose query complexity is $O(\frac{(\log n)^d 2^d}{\epsilon})$. The algorithm is presented in Figure 6.2.

![Algorithm monotone-d-dimD(d, f, \epsilon)](algorithm-monotone-d-dimD-1.png)

Figure 6.2: Distribution-free monotonicity tester for general $d$

Clearly, the query complexity of this algorithm is $O(\frac{(\log n)^d 2^d}{\epsilon})$. We show that this algorithm is indeed a distribution-free monotonicity tester. As before, we can state the following lemma.

---

1A similar generalization for the uniform setting was suggested by [15]
Lemma 6.3.1 For every function \((\epsilon, D)\)-far from monotone, \(f : [n]^d \to A\), if \(S\) is a subset of \([n]^d\) such that for every \(f\)-violation \((\vec{i}, \vec{j})\) either \(\vec{i}\) or \(\vec{j}\) is in \(S\), then there exists a monotone function \(f'\) that differs from \(f\) only at points in \(S\), and therefore \(\Pr_D\{S\} \geq \epsilon\).

The proof of this lemma for the \(d\)-dimensional case is an immediate generalization of the proof for the one-dimensional case. The changes are made to cope with the fact that we are no longer dealing with a linear order, but a partial one. The modification of \(f\) into \(f'\) is again done in an iterative process, in each step we modify the value of one of the points \(\vec{i} \in S\) that has not yet been modified, and the values of all the points below it (i.e., all \(\vec{k} < \vec{i}\)) have already been set. The value of the point \(\vec{i}\) is set in a manner very similar to the one used for the one-dimensional case: We again distinguish between two cases: \(\vec{i} = (1, \ldots, 1)\), and \(\vec{i} \in [n]^d \setminus (1, \ldots, 1)\). In the first case, we set \(f'(\vec{i}) = f(\vec{i})\) if \(S = [n]^d\), and \(f'(\vec{i}) = \min_{\vec{j} \in S} \{f(\vec{j})\}\) otherwise. In the second case, the value \(f'(\vec{i})\) is set to be \(\max_{\vec{j} \leq \vec{i}} \{f(\vec{j})\}\). By the same arguments used in the proof for the one-dimensional version, the resulting function is indeed monotone.

Once again we define an active point in an \(f\)-violation. As before, we say that \(\vec{i}\) (respectively \(\vec{j}\)) is active in the \(f\)-violation \((\vec{i}, \vec{j})\) if for at least half of the values \(\vec{k}\) s.t. \(\vec{i} < \vec{k} < \vec{j}\), the pair \((\vec{i}, \vec{k})\) (respectively \((\vec{k}, \vec{j})\)) is an \(f\)-violation. It is easily seen that indeed for every \(f\)-violation \((\vec{i}, \vec{j})\), at least one of \(\vec{i}\) and \(\vec{j}\) is active in this violation. Therefore if, as before, we denote by \(A_f\) the set of points that are active in some \(f\)-violation, we can deduce that if \(f\) is \((\epsilon, D)\)-far from monotone, then \(\Pr_D\{A_f\} \geq \epsilon\).

Theorem 6.3.2 Algorithm monotone-d-dim\(_D\) is a one-sided error distribution-free monotonicity tester over \([n]^d\).

Proof: To prove this theorem, the following two facts have to be shown: (a) if \(f\) is monotone, it is accepted by the algorithm with probability 1; and (b) if \(f\) is \((\epsilon, D)\)-far from monotone, it is rejected by the algorithm with probability of at least \(\frac{2}{3}\).

As for (a), it follows immediately the definition of the algorithm. Therefore, assume from now on that \(f\) is \((\epsilon, D)\)-far from monotone; we show that the probability that the algorithm accepts \(f\) is at most \(\frac{2}{3}\). As before, there are two possible events in which \(f\) is accepted by the algorithm: the first event is that none of the points sampled by the algorithm according to \(D\) is in \(A_f\), and the second event is that the algorithm picked at least one point \(\vec{i} \in A_f\), but failed to detect that \(\vec{i}\) belongs to some \(f\)-violation.

As in the one-dimensional case, the probability of the first event can be bounded by \(\frac{1}{6}\). To estimate the probability of the second event, let \(\vec{i}\) be a point in \(A_f\) and assume w.l.o.g. that there exists \(\vec{j}\) s.t. \((\vec{i}, \vec{j})\) is an \(f\)-violation and \(\vec{i}\) is active in this violation. Denote by \(k_i\), \(1 \leq l \leq d\), the minimal value such that \(i_l + 2^{k_l} \geq j_l\). By the minimality of \(k_i\), it follows that \(j_l - i_l > \frac{1}{2} \cdot 2^{k_l}\), therefore \((j_1 - i_1) \cdot \ldots \cdot (j_d - i_d) > \frac{1}{2^d} \cdot 2^{k_1} \cdot \ldots \cdot 2^{k_d}\). We know that \(\vec{i}\) is active in the \(f\)-violation \((\vec{i}, \vec{j})\), hence at least half of the values between \(\vec{i}\) and \(\vec{j}\) form an \(f\)-violation with \(\vec{i}\). It follows that at least \(\frac{1}{2^{2d}}\) of the values \(\vec{a}\) in \([2^{k_1}] \times \ldots \times [2^{k_d}]\) will reveal a violation of monotonicity. Since for every possible value for \(k_1, \ldots, k_d\) we sample \(4 \cdot 2^d\) points in \([2^{k_1}] \times \ldots \times [2^{k_d}]\), the probability that no \(f\)-violation is detected is at most \((1 - \frac{1}{2^{2d}})^{4 \cdot 2^d} = (1 - \frac{1}{2})^{2^{2d+1}} \leq \frac{1}{2} \leq \frac{1}{6}\). Therefore, the total probability for the algorithm to fail to detect a violation, and hence to accept \(f\), is at most \(\frac{1}{3}\).

Remark 6.3.1: A similar distribution-free monotonicity tester can be shown for boolean functions of the form \(f : B \to \{0, 1\}\), where \((B, <_B)\) is a partial order that can be described as an "almost balance" tree (i.e., a tree of depth \(O(\log n)\), when \(n = |B|\)).
Chapter 7

A Lower Bound for Distribution-Free Monotonicity Testing in the High-Dimensional Case

In this chapter we show that distribution-free testing of boolean functions, defined over the boolean hypercube, requires a number of queries that is exponential in the dimension. In Section 7.1, we give an overview of the lower-bound proof and present some families of functions used for this proof. Then, for simplicity, we first prove, in Section 7.2, the lower bound for one-sided error testing and later, in Section 7.3, we extend our lower bound for the two-sided error case.

Hereafter, we identify any point \( x \in \{0, 1\}^d \) with the corresponding set \( x \subseteq [d] \). This allows us to apply set theory operations, such as union and intersection, to points. In addition, for any two points \( p, p' \in \{0, 1\}^d \), denote by \( p \| p' \) the point \( x \in \{0, 1\}^d \) that is the concatenation of \( p \) and \( p' \) (i.e., it is identical to \( p \) in its first \( \frac{d}{2} \) coordinates and to \( p' \) in its last \( \frac{d}{2} \) coordinates). Given a point \( x \in \{0, 1\}^d \), we say that \( p \in \{0, 1\}^d \) is the prefix of \( x \) if \( x = p \| p' \) for some \( p' \).

For every two points \( x, y \in \{0, 1\}^d \), denote by \( H(x, y) \) the hamming distance between \( x \) and \( y \). Denote by \( B^d_\lambda \) the set of points in \( \{0, 1\}^d \) of weight \( \lambda d \), that is \( B^d_\lambda = \{ x \in \{0, 1\}^d : |x| = \lambda d \} \).

7.1 The Two Families

To prove the lower bound on the query complexity, we show that any distribution-free monotonicity tester with sub-exponential query complexity is unable to distinguish between monotone functions to functions that are \( \frac{1}{2} \)-far from monotone. To this aim, we consider two families \( \mathcal{F}_1 \) and \( \mathcal{F}_2 \) of pairs \( (f, D_f) \), where \( f \) is a boolean function and \( D_f \) is a probability distribution corresponding to \( f \), both defined over \( \{0, 1\}^d \), such that the following holds:

1. Every function in \( \mathcal{F}_1 \) is monotone. Hence, every tester has to accept every pair \((f, D_f) \in \mathcal{F}_1 \) with probability of at least \( \frac{2}{3} \).

2. Every function \( f \) in \( \mathcal{F}_2 \) is \( \frac{1}{2} \)-far from monotone with respect to \( D_f \). Therefore, every tester has to reject every pair \((f, D_f) \in \mathcal{F}_2 \) with probability of at least \( \frac{2}{3} \) (regardless of the choice of the distance parameter \( \epsilon \)).

3. There exists a constant \( c \) such that there exists no algorithm \( A \) that asks less than \( 2^{cd} \) membership queries and samples the distribution less than \( 2^{cd} \) times, and satisfies:

47
• A accepts every pair \((f, D_f)\) in \(\mathcal{F}_1\) with probability of at least \(\frac{2}{3}\);
• A rejects every pair \((f, D_f)\) in \(\mathcal{F}_2\) with probability of at least \(\frac{2}{3}\).

We conclude that there exists no two-sided error distribution-free tester for monotonicity of boolean functions defined over \(\{0,1\}^d\) with sub-exponential query complexity. (A similar approach was previously used for lower bounds proofs in property testing with respect to the uniform distribution by [31]. However, since the previous proofs considered the uniform setting, the two families in those proofs consist of functions and not pairs of functions and distributions.)

In the rest of this section, we describe the two families of functions (and corresponding distributions), \(\mathcal{F}_1\) and \(\mathcal{F}_2\), defined over \(\{0,1\}^d\). Let \(\alpha < \frac{1}{16}\) be a parameter used for the construction of both families and define \(m \equiv 2^{\alpha d}\). In the construction, we use a set \(\mathcal{M} \subseteq B^{d/2}_{1/2-2\alpha}\) of size \(2m\) such that every two points in \(\mathcal{M}\) are “far apart”. This property of \(\mathcal{M}\) will be used in the proof of the third requirement. We first define the exact requirements from such a set \(\mathcal{M}\) and claim its existence.

**Lemma 7.1.1** There exists a set \(\mathcal{M} \subseteq B^{d/2}_{1/2-2\alpha}\) that satisfies the following conditions:

• \(|\mathcal{M}| = 2m\).
• \(H(p, p') > 2\alpha d\), for every two distinct points \(p, p' \in \mathcal{M}\).

**Proof:** First, note that the second condition of the lemma is equivalent to asking that \(|p \cap p'| < (\frac{1}{4} - 2\alpha)d\) for every two distinct points \(p, p' \in \mathcal{M}\). This is since \(H(p, p') = 2((\frac{1}{4} - \alpha)d - |p \cap p'|)\); hence, \(|p \cap p'| < (\frac{1}{4} - 2\alpha)d\) iff \(H(p, p') > 2\alpha d\).

We show that a randomly drawn set of \(2m\) points in \(B^{d/2}_{1/2-2\alpha}\) satisfies the small intersection condition with positive probability. Let \(P\) be the probability that two randomly drawn points \(x, y \in B^{d/2}_{1/2-2\alpha}\) have intersection of size at least \((\frac{1}{4} - 2\alpha)d\). By the union bound, the probability that a set of \(2m\) randomly drawn points does not satisfy the small intersection condition is at most \(4m^2P\). We prove the existence of such a set \(\mathcal{M}\), by showing that \(4m^2P < 1\). It is also possible to see \(P\) as the probability for a given point \(x \in B^{d/2}_{1/2-2\alpha}\) that a randomly drawn point \(y \in B^{d/2}_{1/2-2\alpha}\) has an intersection of size at least \((\frac{1}{4} - 2\alpha)d\) with \(x\), hence

\[
P = \frac{\sum_{i=(\frac{1}{4}-2\alpha)d}^{(\frac{3}{4}-\alpha)d} \binom{(\frac{1}{4}-\alpha)d}{i} \binom{(\frac{3}{4}+\alpha)d}{d-i}}{\binom{(\frac{3}{4}-\alpha)d}{d}} < (\alpha d + 1) \frac{\binom{\frac{1}{4}-\alpha d}{d}}{\binom{\frac{1}{4}+\alpha d}{d}} = \frac{(\alpha d + 1)((\frac{1}{4} - \alpha)d)!((\frac{1}{4} + \alpha)d)!}{d!(\frac{3}{4} - 2\alpha d)!(\frac{1}{4}d)!(\alpha d)!^2}.
\]

By Stirling formula, \(P < \frac{1}{4m^2}\) for sufficiently large \(d\), implying the desired result. \(\square\)

Using such \(\mathcal{M}\), we define the two families. For each pair \((f, D_f)\), we first define the function \(f : \{0,1\}^d \rightarrow \{0,1\}\) and then, based on the definition of \(f\), define the corresponding distribution \(D_f\) over \(\{0,1\}^d\).
7.1.1 The family $\mathcal{F}_1$

Each function $f$ is defined by first choosing (in all possible ways) two sets $X_1$ and $X_2$, each of size $m$, such that $X_1 \subset B_{1/2-\alpha}^d$ and $X_2 \subset B_{1/2+\alpha}^d$. The choice of the two sets is done as follows:

- Choose a set of $m$ points $M' \subset M$.
- For each point $p$ in $M'$, randomly choose a point $y \in B_{d/2}^d$ and add the point $x = p || y$ to $X_1$ (as needed, $|x| = (1/2 - \alpha)d$).
- For each point $p$ in $M \setminus M'$, randomly choose a point $y \in B_{d/2+4\alpha}^d$ and add the point $x = p || y$ to the set $X_2$ (as needed, $|x| = (1/2 + \alpha)d$).

The function $f$ is now defined in the following manner:

- For every point $x_1 \in X_1$ and every point $y$ such that $y \geq x_1$, set $f(y) = 1$.
- For every point $x_2 \in X_2$ and every point $y$ such that $y \leq x_2$, set $f(y) = 0$.
- For every point $y$, such that $f(y)$ was not defined above, if $|y| \leq d/2$ then $f(y) = 0$, otherwise $f(y) = 1$.

See Figure 7.1 for an example of such a function for $m = 2$.

The distribution $D_f$ corresponding to the function $f$ is the uniform distribution over the $2m$ points in $X_1 \cup X_2$. That is, $D_f(x) = \frac{1}{2m}$ for every $x \in X_1 \cup X_2$ and $D_f(x) = 0$ for any other point in $\{0, 1\}^d$.

The fact that $\mathcal{F}_1$ is not empty follows the existence of such a set $M$. To see that the function $f$ is well defined, and that every function in $\mathcal{F}_1$ is monotone (and hence that the family $\mathcal{F}_1$ satisfies the first requirement of the construction), we observe the following simple lemma.

**Lemma 7.1.2** \{y : y \geq x_1\} \cap \{y : y \leq x_2\} = \emptyset, for every $x_1 \in X_1$ and $x_2 \in X_2$.

**Proof:** If there exists a point $z \in \{y : y \geq x_1\} \cap \{y : y \leq x_2\}$, then $x_1 < x_2$. By the construction of $f$, there exists two different points $p_1, p_2 \in M$ such that $x_1 = p_1 || y_1$ and $x_2 = p_2 || y_2$ for some $y_1$ and $y_2$. Thus, $p_1 \leq p_2$, contradicting the fact that $p_1 \neq p_2$ and $|p_1| = |p_2|$.

Figure 7.1: An example for a function in $\mathcal{F}_1$ for $m = 2$. 

49
7.1.2 The family $\mathcal{F}_2$

Each function $f$ in $\mathcal{F}_2$ is also defined by first choosing two sets $X_1$ and $X_2$, each of size $m$ such that $X_1 \subset B_{1/2-\alpha}^d$ and $X_2 \subset B_{1/2+\alpha}^d$. The choice of the two sets is done as follows:

- Choose a set of $m$ points $M' \subset M$.
- For each point $p$ in $M'$, choose a pair of points $(y_1, y_2)$ such that:
  - $y_1 \in B_{d/2}^d$,
  - $y_2 \in B_{d/2+4\alpha}^d$,
  - $y_1 < y_2$.
- Add the point $x_1 = p||y_1$ to $X_1$ and the point $x_2 = p||y_2$ to $X_2$. The two points $x_1$ and $x_2$ will be referred to as a couple in $f$.

The function $f$ is now defined as follows:

- For every point $x_1 \in X_1$ and every point $y$ such that $y \in U(x_1)$, set $f(y) = 1$, where $U(x) \overset{\text{def}}{=} \{y : y \geq x, |y| \leq \frac{d}{2}\}$.
- For every point $x_2 \in X_2$ and every point $y$ such that $y \in L(x_2)$, set $f(y) = 0$, where $L(x) \overset{\text{def}}{=} \{y : y \leq x, |y| > \frac{d}{2}\}$.
- For every point $y$ such that $f(y)$ was not defined above, if $|y| \leq \frac{d}{2}$ then $f(y) = 0$, otherwise, $f(y) = 1$.

See Figure 7.2 for an example of such a function for $m = 2$.

The distribution $D_f$ is again the uniform distribution over $X_1 \cup X_2$, as in the definition of $\mathcal{F}_1$.

The fact that $\mathcal{F}_2$ is not empty follows immediately the existence of the set $M$. We now argue that the family $\mathcal{F}_2$ satisfies the second requirement of the construction; that is, every function $f$ in $\mathcal{F}_2$ is $\frac{1}{2}$-far from monotone with respect to $D_f$.

**Lemma 7.1.3** $f$ is $\frac{1}{2}$-far from monotone with respect to $D_f$, for every $(f, D_f) \in \mathcal{F}_2$. 

![Figure 7.2: An example for a function in $\mathcal{F}_2$ for $m=2$.](image)
Proof: To transform $f$ into a monotone function, we have to alter the value of $f$ either in $x_1$ or in $x_2$, for every couple $(x_1, x_2)$ in $f$. By this, increasing the distance by $\frac{1}{m}$. Since there are $m$ disjoint couples, the total distance is at least $\frac{1}{2}$. (In fact, since the distance of a boolean function from monotone can be at most $\frac{1}{2}$, the distance is exactly $\frac{1}{2}$.) □

Remark 7.1.1: The fact that the points in $M$ are far apart was not used so far, and it will be used later in the proof of the third requirement. The first two requirements from $F_1$ and $F_2$ are satisfied for every choice of a set $M$.

7.2 A lower bound for one-sided error testing

In this section, to give some intuition of the proof and better understanding of the two families, we first prove our lower bound for the simpler case of one-sided error testers. The arguments used in this section will be later generalized to the general case of two-sided error tester.

Specifically, we prove that there exists no one-sided error distribution-free monotonicity tester with sub-exponential query complexity that accepts every pair $(f, D_f)$ in $F_1$, with probability 1, and rejects every pair $(f, D_f)$ in $F_2$, with high probability. To this aim, we show that for every tester $A$, there exists a pair $(f, D_f) \in F_2$, such that with high probability, the execution of $A$ on $(f, D_f)$ is also consistent with some monotone function from $F_1$. Since $A$ has to accept every monotone function, then $A$ has to accept $f$ with high probability.

The above claim is simple if the tester is not allowed to use membership queries, but only to sample the distribution $D_f$. In this case, to distinguish between the two families, $F_1$ and $F_2$, the tester has to detect a couple in the function from $F_2$. Given a size $i$ sample of the distribution, the probability that the $(i + 1)^{th}$ sample will be a couple of one of the already known $i$ points is at most $\frac{i}{m}$. Therefore, the probability to distinguish between the two families using $n$ samples is $\sum_{i=1}^{n} \frac{i}{m} = O\left(\frac{n^2}{m}\right)$. Since $m$ is exponential in $d$, then distinguishing between the two families requires an exponential number of queries. However, the difference between one tester to the other is in the tester’s choice of membership queries, and dealing with the tester’s membership queries is where the difficulty of this proof lays.

To state our claim formally, we introduce some notation. Let $A$ be a tester with query and sample complexity $n = n(d, \frac{1}{c})$. Such a tester can also be viewed as a mapping from a sequence $\{(p_i, v_i)\}_{i=1}^{j}$ of labelled points to either “sample the distribution” or “query $p_{j+1}$” if $j < n$, and to “accept” or “reject” if $j = n$. We refer to such a sequence of labelled points obtained by $A$ as a knowledge sequence. Given a pair $(f, D_f)$, denote by $S_f$ the knowledge sequence learnt by $A$ during its run on $(f, D_f)$ (for some possible execution of $A$ on $(f, D_f)$). We say that a function $f$ is consistent with a knowledge sequence $S$ if $f(p_i) = v_i$, for every $1 \leq i \leq j$.

As stated above, we will show that for some constant $c$, for every tester $A$ with query complexity $n < 2^{cd}$, there exists a pair $(f, D_f) \in F_2$ such that with high probability, over the possible executions of $A$ and the sampling of the domain according to $D_f$, the sequence $S_f$ is consistent with some monotone function $f' \in F_1$. Hence, with high probability $S_f$ causes $A$ to accept, contradicting the requirement that $A$ must reject $(f, D_f)$ with probability of at least $\frac{2}{3}$.

For this, we show that for every tester $A$ and for every possible choice of random coins for $A$, the probability that, for a randomly drawn pair $(f, D_f)$ from $F_2$, the sequence $S_f$ is consistent with some monotone function from $F_1$ is very high (where the probability is taken over the choice of $(f, D_f)$ and the sampling of $D_f$). In other words, for every tester $A$ and
for every choice of random coins for \( A \), most pairs in \( \mathcal{F}_2 \) are such that, with high probability (over the sampling of \( D_f \)), \( S_f \) is consistent with a monotone function. Hence, for every tester \( A \), there exists a pair \((f, D_f) \in \mathcal{F}_2\) such that for most choices of random coins for \( A \), the probability that \( S_f \) is consistent with a monotone function is very high.

**Lemma 7.2.1** For every tester \( A \), there exists a pair \((f, D_f) \in \mathcal{F}_2\) such that

\[
\Pr\{S_f \text{ is consistent with a monotone function}\} \geq 1 - \frac{2n^2}{m},
\]

where the probability is taken over the choice of random coins for \( A \) and the samplings of the distribution \( D_f \).

Based on the above lemma we can state our lower bound.

**Theorem 7.2.2** Let \( c = \alpha/3 \). For every one-sided error monotonicity tester \( A \) that asks less than \( 2^d \) membership queries and samples the distribution less than \( 2^d \) times, there exists a pair \((f, D_f) \in \mathcal{F}_2\) such that \( \Pr\{A \text{ accepts } (f, D_f)\} > \frac{1}{3} \).

**Proof:** Set \( n = 2 \cdot 2^d \). By Lemma 7.2.1, there exists a pair \((f, D_f) \in \mathcal{F}_2\) such that \( \Pr\{S_f \text{ is consistent with a monotone function}\} \geq 1 - \frac{2n^2}{m} \). By our choice of \( c \) and \( n \), for this pair \((f, D_f) \in \mathcal{F}_2\) we have \( \Pr\{S_f \text{ is consistent with a monotone function}\} \geq 1 - \frac{4 \cdot 2^{2d/3}}{2^d} = 1 - \frac{4 \cdot 2^{d/3}}{3} > \frac{1}{3} \) (for sufficiently large \( d \)).

It remains to prove Lemma 7.2.1. To do so, we first state the condition that a knowledge sequence has to satisfy in order to be consistent with some function in \( \mathcal{F}_1 \). Note that in order for a knowledge sequence to be extendable to a monotone function in \( \mathcal{F}_1 \), it is not enough that \( S_f \) is consistent with some monotone function. If for some couple \((x_1, x_2)\) with prefix \( p \), the sequence \( S_f \) contains points \( y_1 \in U(x_1) \) and \( y_2 \in L(x_2) \) both with the same prefix \( p \), then we can deduce that \( f \in \mathcal{F}_2 \) regardless of whether \( y_1 \) and \( y_2 \) are comparable or not. Therefore, we have to define a relaxed notion of a witness for non-monotone functions for our case, such that every sequence that does not contain a witness is indeed extendable to a function in \( \mathcal{F}_1 \). (In general, it is enough to show that there exists an arbitrary monotone function that is consistent with the knowledge sequence, and not necessarily a function in \( \mathcal{F}_1 \). However, we will use this fact later on when extending the proof of the lower bound to the two-sided error case.)

**Definition 7.2.3** Let \( S_f \) be a knowledge sequence learnt by \( A \) while running on the pair \((f, D_f) \in \mathcal{F}_2\). We say that \( S_f \) contains a witness if there exist points \( y_1, y_2 \in S_f \) such that \( y_1 \in U(x_1) \) and \( y_2 \in L(x_2) \) for some couple \((x_1, x_2)\) in \( f \).

**Lemma 7.2.4** If the knowledge sequence \( S_f \), learnt by \( A \) while running on a pair \((f, D_f) \in \mathcal{F}_2\), does not contain a witness, then there exists a function \( f' \in \mathcal{F}_1 \) that is consistent with \( S_f \).

**Proof:** To prove the existence of such a function \( f' \), we show how to construct the sets \( X'_1 \) and \( X'_2 \) such that \( f' \), the function induced by \( X'_1 \) and \( X'_2 \), is in \( \mathcal{F}_1 \) and is consistent with \( S_f \). The construction of \( X'_1 \) and \( X'_2 \) is as follows:

1. For every couple \((x_1, x_2)\) in \( f \), if \( S_f \) contains points in \( U(x_1) \) then add \( x_1 \) to \( X'_1 \); otherwise, if \( S_f \) contains points in \( L(x_2) \), then add \( x_2 \) to \( X'_2 \). Let \( \mathcal{M}_S \) be the set of \( \frac{d}{3} \) length prefixes of all the points that were added to \( X'_1 \) and to \( X'_2 \). Note that since \( S_f \) does not contain a witness, \( \mathcal{M}_S = |X'_1| + |X'_2| \).

52
2. Choose \( m - |X'_1| \) points in \( \mathcal{M} \setminus \mathcal{M}_S \); denote the set of selected points by \( \mathcal{M}_1 \). For each prefix \( p \in \mathcal{M}_1 \), choose a point \( y \in B_{1/2}^{d/2} \), and add \( p\|y \) to \( X'_1 \).

3. Denote by \( \mathcal{M}_2 \) the set \( \mathcal{M} \setminus (\mathcal{M}_S \cup \mathcal{M}_1) \). For each prefix \( p \in \mathcal{M}_2 \), choose a point \( y \in B_{1/2}^{d/2} \) and add \( p\|y \) to \( X'_2 \).

Clearly, the function \( f' \) is consistent with \( S_f \). It remains to show that it is indeed a function in \( \mathcal{F}_1 \). Since \( S_f \) does not contain a witness, none of the prefixes in \( \mathcal{M} \) was used more than once in the construction of \( X'_1 \) and \( X'_2 \). Hence, the sets \( X'_1 \) and \( X'_2 \) can also be chosen in the process described in the construction of \( \mathcal{F}_1 \). Therefore, the function \( f' \) defined by the two sets \( X'_1 \) and \( X'_2 \) is indeed in \( \mathcal{F}_1 \).

We now prove that the probability that the knowledge sequence \( S_f \), learnt by \( A \) during its run on a randomly chosen pair \( (f, D_f) \in \mathcal{F}_2 \), contains a witness, is very small. To do so, we prove that the probability for any step of \( A \), that the knowledge sequence will contain a witness after that step is very small. The following technical lemma will be used later in the proofs.

**Lemma 7.2.5** Let \( f \) be a function in \( \mathcal{F}_2 \). Then, \( U(x_1) \cap U(x'_1) = \phi \) for every \( x_1, x'_1 \in X_1 \) such that \( x_1 \neq x'_1 \). Similarly, \( L(x_2) \cap L(x'_2) = \phi \) for every \( x_2, x'_2 \in X_2 \) such that \( x_2 \neq x'_2 \).

**Proof:** Consider \( x_1, x'_1 \in X_1 \) such that \( x_1 \neq x'_1 \) and assume, towards a contradiction, that there exists a point \( y \in U(x_1) \cap U(x'_1) \). Let \( p \) and \( p' \) be the prefixes of \( x_1 \) and \( x'_1 \), respectively, and denote by \( p_y \) the \( \frac{d}{2} \) length prefix of \( y \). Since \( y \in U(x_1) \) then \( H(x_1, y) \leq \alpha d \), and hence \( H(p, p_y) \leq \alpha d \). Similarly, \( H(p', p_y) \leq \alpha d \). Therefore, \( H(p, p') \leq 2\alpha d \), contradicting the requirements from \( \mathcal{M} \). The proof for \( X_2 \) is similar.

An immediate conclusion of the above lemma is that, for every query that \( A \) may ask, there can be only one prefix in \( \mathcal{M} \) such that this query belongs to \( U(x_1) \) or \( L(x_2) \) for some couple \((x_1, x_2)\) with that prefix. That is,

**Lemma 7.2.6** For every point \( z \in \{y \in \{0,1\}^d : (\frac{1}{2} - \alpha)d \leq |y| \leq \frac{d}{2}\} \), there exists at most one prefix \( p \in \mathcal{M} \) such that \( z \in U(x) \) for some point \( x \in B_{1/2}^{d/2-\alpha} \) with prefix \( p \). Similarly, for every point \( z \in \{y \in \{0,1\}^d : \frac{d}{2} < |y| \leq (\frac{1}{2} + \alpha)d\} \), there exists at most one prefix \( p \in \mathcal{M} \) such that \( z \in L(x) \) for some point \( x \in B_{1/2+\alpha}^{d/2} \) with prefix \( p \).

Given \( (f, D_f) \in \mathcal{F}_2 \) and a knowledge sequence \( S_f \), denote by \( S_f(i) \) the length \( i \) prefix of \( S_f \). We say that the \( i \)th labelled point in \( S_f(i) \) is destructive if \( S_f(i - 1) \) does not contain a witness and \( S_f(i) \) does.

**Lemma 7.2.7** For every tester \( A \) and every possible sequence of random coins for \( A \), the following holds:

\[
\Pr\{S_f(i + 1) \text{ contains a witness} \mid S_f(i) \text{ does not contain a witness}\} \leq \frac{i + 2}{m},
\]

where the probability is taken over the random choice of \( (f, D_f) \) from \( \mathcal{F}_2 \) that is consistent with \( S_f(i) \), and over the possible samplings of \( D_f \) for the chosen pair \((f, D_f)\).
Proof: We distinguish between the two possible types of steps for $A$. The first possibility is that in step $i + 1$ tester $A$ asks for a sample of the distribution $D_f$, and the second possibility is that $A$ chooses to query the function. In the first case, for every possible choice of $(f, D_f)$ from $F_2$, we show that the probability that $S_f(i + 1)$ contains a witness is at most $\frac{1}{2m}$. Let $(f, D_f)$ be a pair in $F_2$ that is consistent with $S_f(i)$ and let $X_1$ and $X_2$ be the sets used for its construction. The sampled point can be destructive only if $S_f(i)$ already contains a related point. That is, if for some couple $(x_1, x_2)$ in $f$, either the sampled point is $x_1$ and there are points in $S_f(i)$ from $L(x_2)$, or the sampled point is $x_2$ and $S_f(i)$ contains points from $U(x_1)$. By Lemma 7.2.6, every point in $S_f(i)$ can be in $U(x_1)$ for at most one point $x_1 \in X_1$ and similarly, every point can be in $L(x_2)$ for at most one point $x_2 \in X_2$. Hence, for every point in $S_f(i)$, only sampling of the relevant point $x_1$ or $x_2$ can cause the knowledge sequence to contain a witness.

The probability for this event is $\frac{1}{2m}$. Therefore, the probability that the sampled point, that was returned by the oracle, is destructive is bounded by $\frac{1}{2m}$.

In the second case, let $q_{i+1}$ be the query asked by $A$ in step $i + 1$. If $|q_{i+1}| < \left(\frac{1}{2} - \alpha\right)d$ or $|q_{i+1}| > \left(\frac{1}{2} + \alpha\right)d$, then $S_f(i + 1)$ cannot contain a witness. Assume with out loss of generality that $|q_{i+1}| \leq \frac{d}{2}$ (the proof for the case that $|q_{i+1}| > \frac{d}{2}$ is similar). By Lemma 7.2.6, there can be at most one relevant prefix $p \in M$ such that a couple $(x_1, x_2)$ with prefix $p$ can include $q_{i+1}$ in $U(x_1)$. Assume there exists a 0-labelled point $z$ in $S_f(i)$ that can possibly be in $L(x_2)$ for a point $x_2$ with prefix $p$ (otherwise, $q_{i+1}$ is not destructive). The sequence $S_f(i + 1)$ will contain a witness only if $q_{i+1} \in U(x_1)$ for the point $x_1$ corresponding to $x_2$. Note that $S_f$ does not necessarily include $x_2$ itself, but only points in $L(x_2)$. We will bound the probability, over the possible choices of a pair $(f, D_f)$ such that $f$ is consistent with $S_f(i)$, that indeed $q_{i+1} \in U(x_1)$, and show that this probability is at most $\frac{2}{m}$.

The bound is obtained by giving an upper bound on the probability, for a randomly drawn point $x_1$ that is consistent with $S_f$, that indeed $q_{i+1} \in U(x_1)$. Before proving this bound, we explain why such a bound implies a bound on the probability that a randomly drawn pair in $F_2$ that is consistent with $S_f$ satisfies $q_{i+1} \in U(x_1)$. We first observe that all the functions in $F_2$ that are consistent with $S_f$ contain a couple with the prefix $p$. It is also easy to see that, once the prefix of a couple is determined and we consider only functions in $F_2$ that include a couple with that prefix, the distribution induced over the possible choices of this couple in $f$ is uniform over all prefix $p$ couples $(x_1, x_2)$ such that $x_1 < x_2$. In addition, for every such function $f$, the question whether indeed $q_{i+1} \in U(x_1)$ is fully determined by the choice of the prefix $p$ couple $(x_1, x_2)$. Hence, it is enough to bound the probability over the possible choices of the couple $(x_1, x_2)$ in $f$ with the prefix $p$ such that the couple is consistent with $S_f(i)$, that $q_{i+1} \in U(x_1)$. Following the fact that every $(\frac{1}{2} - \alpha)d$-weight point $x$ plays the role of $x_1$ in the same number of couples (note that the weight of $x_1$ and $x_2$ is identical in all the couples), we conclude that the distribution induced by a random choice of a couple with prefix $p$ over the choice of $x_1$ (or $x_2$) is uniform. Therefore, bounding the probability that a random couple that is consistent with $S_f$ satisfies $q_{i+1} \in U(x_1)$ can be done by considering the probability that a random choice of $x_1$, that is consistent with $S_f(i)$, satisfies $q_{i+1} \in U(x_1)$.

We are interested in bounding the probability that, for a random choice of $x_1$, indeed $q_{i+1} \in U(x_1)$. This will be done through the following three steps. In step 1, we give an upper bound on the number of possible choices for $x_1$ such that $q_{i+1} \in U(x_1)$. In step 2, we show a lower bound on the number of choices for $x_1$ that are consistent with $S_f$. Finally, in step 3, we combine these two bounds to get an upper bound on the probability of the desired event.

\[ \text{Note that, in general, if the points in } M \text{ were not “far apart”, it is possible that the couples are very close to one another, and hence almost every sampled point is destructive.} \]
1. Since \( x_1 \) must have the prefix \( p \), we bound the number of points \( x \in B^{d}_{1/2-\alpha} \) with prefix \( p \) such that \( x < q_{i+1} \). Clearly, this number is maximal when \( q_{i+1} \) is also with the prefix \( p \) and \( |q_{i+1}| = \frac{d}{2} \). Therefore, the number of possible choices for \( x_1 \) such that \( q_{i+1} \in U(x_1) \) is at most \( k \defeq \left( \frac{1}{2} + \alpha \right)^d \).

2. Our choice of \( x_1 \) is constrained by the 0-labelled points that we have in \( L(x_2) \), by 1-labelled points that could have possibly been in \( L(x_2) \), and by 0-labelled points that we know that could have been in \( U(x_1) \) for possible choices of \( x_1 \). The number of possible choices for \( x_1 \), with respect to the 0-labelled points known in \( L(x_2) \) and the 1-labelled points that could have been in \( L(x_2) \), can always be bounded from below by the number of such choices, given that the corresponding point \( x_2 \) is known. Hence, the number of choices for \( x_1 \) given only the points known in \( L(x_2) \) is bounded by \( K \defeq \left( \frac{1}{2} + 2\alpha \right)^d \) (the first \( \frac{d}{2} \) coordinates are always \( p \)). However, our choice is also constrained by the 0-labelled points known in \( S_f(i) \) that could have been in \( U(x_1) \). Using arguments similar to the ones that were used in step 1, each of these points can eliminate at most \( k \) possible choices for \( x_1 \) that are consistent with \( S_f \).

3. We can now deduce that the probability that a randomly drawn point \( x_1 \), that is consistent with \( S_f(i) \), satisfies \( q_{i+1} \in U(x_1) \), can be upper bounded by \( \frac{k^i}{K^{i-k}} \). To bound this probability, we first bound the ratio \( \frac{k}{K} \). We have, \( \frac{k}{K} = \frac{((\frac{1}{2} + 2\alpha)^d)!}{((\frac{1}{2} + \alpha)^d + 2)^d} < (\frac{8\alpha}{2})^d \). Therefore, the probability that \( q_{i+1} \in U(x_1) \) is bounded by \( \frac{(8\alpha)^d}{1 - (8\alpha)^d} \). By the choice of \( \alpha < \frac{1}{16} \), we have that \( (8\alpha)^d < \frac{1}{2} = \frac{1}{m} \). Hence, \( \frac{(8\alpha)^d}{1 - (8\alpha)^d} < \frac{\frac{1}{2}}{1 - \frac{1}{m}} = \frac{1}{m-1} \). Since \( i < \frac{m}{2} \), this probability is bounded by \( \frac{2}{m} \).

Based on the above lemma, we can now prove the following proposition.

**Proposition 7.2.8** For every tester \( A \) and every possible choice of random coins for \( A \):

\[
\Pr\{ S_f \text{ contains a witness} \} \leq \frac{n^2}{m},
\]

where the probability is taken over the random choice of the pair \((f, D_f)\) from \( F_2 \) and over the possible sampling of \( D_f \) for the chosen pair \((f, D_f)\).

**Proof:** Given a tester \( A \) and a choice of random coins for \( A \), the probability that \( S_f \) contains a witness for a randomly chosen pair \((f, D_f) \in F_2 \) is bounded by

\[
\sum_{i=1}^{n} \Pr\{ S_f(i+1) \text{ contains a witness} \mid S_f(i) \text{ does not contain a witness} \}.
\]

By Lemma 7.2.7, this probability can be bounded by \( \sum_{i=1}^{n} \frac{i+2}{m} = \frac{n(n+1)+4n}{2m} < \frac{n^2}{m} \). 

Lemma 7.2.1 follows immediately from the above proposition.
7.3 A lower Bound for Two-Sided Error Testing

In this section we extend Theorem 7.2.2 to the two-sided error case; that is, we prove that there exists no two-sided error distribution-free monotonicity tester with sub-exponential query complexity that accepts every pair \((f, D_f) \in \mathcal{F}_1\), with high probability, and rejects every pair \((f, D_f) \in \mathcal{F}_2\), with high probability. To do so, we show that there exists a constant \(c\), such that for every tester \(A\) that asks less than \(2^{cd}\) membership queries and samples \(D_f\) less than \(2^{cd}\) times, the distributions induced on the set of possible knowledge sequences by running \(A\) on a random pair from \(\mathcal{F}_1\) and from \(\mathcal{F}_2\) are statistically close.

Let \(A\) be a tester with query and sample complexity \(n < 2^{cd}\). Denote by \(P_{A_1}\) the distribution induced over length \(n\) knowledge sequences by running \(A\) on a randomly drawn pair \((f, D_f)\) from \(\mathcal{F}_1\) (note that \(P_{A_1}\) is induced by the random choice of the pair \((f, D_f)\) in \(\mathcal{F}_1\), the choice of random coins for \(A\), and the sampling of the domain according to \(D_f\)); \(P_{A_2}\) is defined similarly. We show that for every such tester \(A\), the statistical difference between the two distributions \(P_{A_1}\) and \(P_{A_2}\) is exponentially small.

In the previous section we saw that for every tester \(A\), the probability, measured with respect to \(P_{A_2}\), that a knowledge sequence contains a witness is very small. At first glance, it may seem that this is enough to show that the two distributions \(P_{A_1}\) and \(P_{A_2}\) are statistically close. However, this is not true. In the case of functions in \(\mathcal{F}_1\), for every prefix \(p \in \mathcal{M}\) there exists a point either in \(X_1\) or in \(X_2\) with the prefix \(p\) (\(m\) of the prefixes in \(\mathcal{M}\) are used as prefixes for points in \(X_1\), while the other \(m\) are used as prefixes for points in \(X_2\)). This is not the case for functions in \(\mathcal{F}_2\), where we choose only \(m\) out of the \(2^m\) prefixes, and select a couple \((x_1, x_2)\) with each prefix. Hence, one can suggest the following testing approach:

1. repeat the following two steps:
   (a) choose a prefix \(p \in \mathcal{M}\).
   (b) try to find out, using membership queries, whether there exists a point either in \(X_1\) or in \(X_2\) with the prefix \(p\).

2. if for at least \(\frac{1}{4}\) of the prefixes chosen in step 1, no witness was found for a point in \(X_1\) or in \(X_2\) with that prefix, decide that the function is in \(\mathcal{F}_2\); otherwise, decide that the function is in \(\mathcal{F}_1\).

(Such an approach is not relevant in the case of one-sided error testing, since the tester has to accept every function in \(\mathcal{F}_1\) with probability 1.) Clearly, if the tester is only allowed to sample the distribution \(D_f\), this testing approach requires an exponential number of queries. However, it may be possible to use membership queries to significantly reduce the required query complexity. Hence, showing that with high probability the knowledge sequence \(S_f\) does not contain a witness is not enough, and there are other undesired events we have to eliminate.

Assume with out loss of generality that the tester \(A\) works in two stages. In the first stage, the tester performs \(l\) samplings of the distribution \(D_f\), and in the second stage, \(A\) performs \(k\) membership queries (that is, \(l + k = n\)). We define formally each of the undesired events, and prove that with high probability, both under \(P_{A_1}\) and under \(P_{A_2}\), these events do not occur in the knowledge sequence learnt by \(A\). As before, we denote by \(S_f(i)\) the length \(i\) prefix of \(S_f\), the knowledge sequence learnt by \(A\). The first undesired event is that the tester succeeds to learn useful information about the given pair \((f, D_f)\) in the querying stage (and not only in the sampling stage).
Definition 7.3.1 A knowledge sequence $S_f$, learnt by $A$ while running on a pair $(f, D_f)$, contains non-sampled points (in short, n.s. points) if either for some $x_1 \in X_1$ that was not sampled in the first stage of the tester, $S_f$ contains points in $U(x_1)$, or that for some $x_2 \in X_2$ that was not sampled in the first stage of the tester, $S_f$ contains points in $L(x_2)$.

We first prove that this event is unlikely under $P_{1A}$. The proof is similar to the proof of Lemma 7.2.7.

Lemma 7.3.2 For every tester $A$ and every possible sequence of random coins for $A$, the following holds:

$$Pr\{S_f(i+1) \text{ contains n.s. points } | S_f(i) \text{ does not contain n.s. points} \} \leq \frac{2}{m},$$

where the probability is taken over the random choice of $(f, D_f)$ from $\mathcal{F}_1$ that is consistent with $S_f(i)$, and over the possible samplings of $D_f$ for the chosen pair $(f, D_f)$.

Proof: By definition, in order for $S_f(i+1)$ to contain non-sampled points, $l < i + 1$. That is, in step $i + 1$, the tester queries the function. Let $q_{i+1}$ be the query performed by the tester in step $i + 1$, and assume with out loss of generality that $(\frac{1}{2} - \alpha)d \leq |q_{i+1}| \leq \frac{d}{2}$. We show that the probability that $q_{i+1} \in U(x_1)$ for some point $x_1 \in X_1$ that was not sampled by the tester in the sampling stage is bounded by $\frac{2}{m}$. By Lemma 7.2.6, there exists at most one prefix $p \in \mathcal{M}$ such that $q_{i+1}$ can be in $U(x_1)$ for a point $x_1 \in X_1$ with prefix $p$. Hence, $S_f(i)$ includes no point $x \in X_1 \cup X_2$ with prefix $p$ (otherwise, the probability that $S_f(i+1)$ contains non-sampled points is 0). As in the proof of Lemma 7.2.7, once the prefix of a point $x_1 \in X_1$ is determined, the distribution induced by the random choice of the pair $(f, D_f) \in \mathcal{F}_1$ on the choice of $x_1$ is uniform. Thus, to bound the probability for a random pair $(f, D_f) \in \mathcal{F}_1$ that is consistent with $S_f(i)$ that $q_{i+1} \in U(x_1)$ for $x_1 \in X_1$, it is enough to bound the probability for a randomly chosen $x_1$ with prefix $p$ that is consistent with $S_f(i)$, that $q_{i+1} \in U(x_1)$. The arguments used for this bound are simpler than the ones used in the proof of Lemma 7.2.7, since this time the choice of $x_1$ is only constrained by 0-labelled points that could have possibly been in $U(x_1)$, and not by the choice of the couple of $x_2$ (since the functions in question are all in $\mathcal{F}_1$).

Our bound will be obtained through the following three steps: In step 1, we bound from above the number of possible choices for $x_1$ such that $q_{i+1} \in U(x_1)$. In step 2, we bound from below the number of possible choices for $x_1$ that are consistent with $S_f(i)$. Finally, in step 3, we combine the two bounds to obtain an upper bound on the probability that for a random $x_1$ that is consistent with $S_f(i)$ indeed $q_{i+1} \in U(x_1)$.

1. Since $x_1$ must have the prefix $p$, we bound the number of points $x \in B_{1/2}^{|d_{i+1}|}$ with prefix $p$ such that $x < q_{i+1}$. As before, this number is maximal when $q_{i+1}$ is also with the prefix $p$ and $|q_{i+1}| = \frac{d}{2}$. Therefore, the number of possible choices for $x_1$ such that $q_{i+1} \in U(x_1)$ is at most $k \overset{\text{def}}{=} (\frac{1}{2} + \alpha)d$.

2. The number of possible choices for $x_1$ given only the prefix $p$ is $K \overset{\text{def}}{=} \frac{d}{2}$ (the first $\frac{d}{2}$ coordinates are always $p$). However, our choice of $x_1$ is constrained by the 0-labelled points in $S_f(i)$ that could have possibly been in $U(x_1)$ for possible choices of $x_1$. Using arguments similar to step 1, each of these points can eliminate at most $k$ choices for $x_1$. Hence, there are at least $K - ik$ possible choices for $x_1$ that are consistent with $S_f$. 

57
3. We conclude that the probability that a randomly drawn point \( x_1 \), that is consistent with \( S_f(i) \), satisfies \( q_{i+1} \in U(x_1) \), can be upper bounded by \( \frac{k}{\kappa R^k} \). We have, \( k = \frac{((\frac{1}{2} + \alpha) d)^2 (\frac{d}{2})^2}{(\frac{1}{2})! (\frac{d}{2})! (d!)^2} = \frac{1 - \frac{d}{2}}{1 - \frac{d}{2} - (\frac{1}{2} + \alpha) d - \frac{d}{2}} < (\frac{1}{2})^{(\frac{1}{2} + \alpha) d} < (\frac{1}{2})^{\frac{d}{2}} \). Therefore, the probability that \( q_{i+1} \in U(x_1) \) can be bounded by \( (\frac{1}{2})^{\frac{d}{2}} \). By the choice of \( \alpha < \frac{1}{16} \) and the fact that \( i < m \), we have that \( i \cdot \frac{1}{2}^d < \frac{1}{2} \) for sufficiently large values of \( d \). Hence, \( (\frac{1}{2})^{\frac{d}{2}} < 2(\frac{1}{2})^{\frac{d}{2}} < \frac{2}{m} \).

Based on the above lemma, we now prove the following proposition.

**Proposition 7.3.3** For every tester \( A \) and every possible choice of random coins for \( A \):

\[
Pr\{S_f \text{ contains n.s. points} \} \leq \frac{2n}{m},
\]

where the probability is taken over the random choice of the pair \((f, D_f)\) from \( \mathcal{F}_1 \) and over the possible sampling of \( D_f \) for the chosen pair \((f, D_f)\).

**Proof:** Given a tester \( A \) and a choice of random coins for \( A \), the probability that \( S_f \) contains non-sampled points for a randomly chosen pair \((f, D_f)\) \( \in \mathcal{F}_1 \) can be bounded by

\[
\sum_{i=1}^{n} Pr\{S_f(i+1) \text{ contains non-sampled points} \mid S_f(i) \text{ does not contain non-sampled points} \}.
\]

By Lemma 7.3.2, this probability can be bounded by \( \sum_{i=1}^{n} \frac{2n}{m} = \frac{2n}{m} \).

We now show that this event is unlikely also under \( \mathcal{P}_A \).

**Lemma 7.3.4** For every tester \( A \) and every possible sequence of random coins for \( A \), the following holds:

\[
Pr\{S_f(i+1) \text{ contains n.s. points} \mid S_f(i) \text{ does not contain n.s. points} \} \leq \frac{i + 2}{m},
\]

where the probability is taken over the random choice of \((f, D_f)\) from \( \mathcal{F}_2 \) that is consistent with \( S_f(i) \), and over the possible samplings of \( D_f \) for the chosen pair \((f, D_f)\).

**Proof:** Clearly, the probability that for a randomly drawn pair \((f, D_f)\) \( \in \mathcal{F}_2 \) that is consistent with \( S_f(i) \) the point \( q_{i+1} \) is in \( U(x_1) \) for a point \( x_1 \in X_1 \) that is not known in \( S_f(i) \) can be bounded by the probability that for such a pair \((f, D_f)\), indeed \( q_{i+1} \in U(x_1) \) given that \( x_2 \), the couple of \( x_1 \), appears in \( S_f(i) \) (the knowledge of \( x_2 \) can only raise the probability of the tester to choose a point \( q_{i+1} \in U(x_1) \)). However, this probability was already shown in Lemma 7.2.7 to be bounded by \( \frac{i + 2}{m} \).

The following proposition is proved in a similar way to Proposition 7.2.8, with Lemma 7.3.4 playing the role of Lemma 7.2.7.

**Proposition 7.3.5** For every tester \( A \) and every possible choice of random coins for \( A \):

\[
Pr\{S_f \text{ contains pre-sampling information} \} \leq \frac{n^2}{m},
\]

where the probability is taken over the random choice of the pair \((f, D_f)\) from \( \mathcal{F}_2 \) and over the possible sampling of \( D_f \) for the chosen pair \((f, D_f)\).
The next undesired event is that during the sampling stage, the tester receives the same sample more than once. We show that this event is unlikely both under \( \mathcal{P}_1^A \) and under \( \mathcal{P}_2^A \).

**Definition 7.3.6** A knowledge sequence \( S_f \) is repetitive if there exists a point in \( X_1 \cup X_2 \) that was sampled at least twice in \( S_f \).

**Proposition 7.3.7** For every tester \( A \) and every possible choice of random coins for \( A \):

\[
\Pr\{S_f \text{ is repetitive} \} \leq \frac{n^2}{m},
\]

where the probability is taken over the random choice of the pair \((f, D_f)\) from \( \mathcal{F}_1 \) or from \( \mathcal{F}_2 \) and over the possible sampling of \( D_f \) for the chosen pair \((f, D_f)\).

**Proof:** To prove the lemma, we show that given a set \( L \) of size \( m \), the probability that at least one of the points in \( L \) appears more than once in a random sample of size \( n \) is bounded by \( \frac{n^2}{m} \). The probability that the \( i^{th} \) sample from \( L \) is a point that was sampled before can be bounded by \( \frac{1}{m} \). Hence, using the union bound, the probability that at least one of the \( n \) samples repeats a point that was sampled before is bounded by \( \sum_{i=1}^{n} \frac{i}{m} = \frac{n(n+1)}{2m} \leq \frac{n^2}{m} \). \( \square \)

We conclude that with very high probability, under both distributions, none of the undesired events occurs.

**Definition 7.3.8** We say that a knowledge sequence \( S_f \) is good if \( S_f \) does not contain a witness nor non-sampled points and is not repetitive; otherwise, we say that \( S_f \) is bad.

Note that, if a knowledge sequence \( S_f \) is good, then the sequence is fully determined by the answers received from the sampling oracle in the first stage of the tester. That is, unless a query \( q \) is in \( U(x) \) or \( L(x) \) for a point \( x \) that was sampled in the first stage of the tester, the answer that the tester receives is 0 if \( |q| \leq \frac{d}{2} \), and 1 otherwise.

By Propositions 7.2.8, 7.3.3, 7.3.5, and 7.3.7, for every tester \( A \), with probability of at least \( 1 - \frac{5n^2}{m} \), both with respect to \( \mathcal{P}_1^A \) and with respect to \( \mathcal{P}_2^A \), the knowledge sequence learnt by \( A \) is good. We show that for every good knowledge sequence \( S_f \), the two probabilities \( \mathcal{P}_1^A(S_f) \) and \( \mathcal{P}_2^A(S_f) \) are very close. That is,

**Lemma 7.3.9** For every good knowledge sequence \( S_f \) it holds that

\[
1 - \frac{O(n^2)}{m} \leq \frac{\mathcal{P}_1^A(S_f)}{\mathcal{P}_2^A(S_f)} \leq 1 + \frac{O(n^2)}{m}.
\]

**Proof:** To prove the lemma, we show that, for every step the tester performs, and for every answer it receives, the ratio between the probability of this answer with respect to both a random function from \( \mathcal{F}_1 \) and a random function from \( \mathcal{F}_2 \) is bounded by \( (1 \pm \frac{n}{2m-n}) \). Hence, for every knowledge sequence, the ratio between the probability of this sequence under the two distributions can be bounded from above by \( (1 + \frac{n}{2m-n})^n = (1 + O(n^2)) \) and from below by \( (1 - \frac{n}{2m-n})^n = (1 - O(n^2)) \) (where the equality follows the binomial expansion of \( (1 + \frac{n}{2m-n})^n \)).

\[
2(1 + \frac{n}{2m-n})^n \leq \frac{n}{2m-n} \leq (1 + \frac{n}{2m-n})^n
\]

We show that \( 2m-n > 0 \). Since \( m > n^2 + n \), we have \( \frac{2m-n}{n^2-n} = \frac{2m-n}{2m-n-n^2} = (1 + \frac{n^2}{2m-n}) \leq (1 + \frac{n^2}{m}) \).

The expansion for \( (1 - \frac{n}{2m-n})^n \) is similar.
Recall that we assume that the tester has two stages. The first stage is the sampling stage, while the second stage is dedicated to the membership queries of the tester. Since the sequences in question are all good, the answers that the tester receives in the second stage are fully determined by the answers it received in the sampling stage. Hence, the probability of these answers under both distributions is 1 and thus is identical.

It remains to show that indeed the ratio between the two probabilities holds with respect to the first stage of the tester, the sampling stage. Let \( x \) be the answer returned to the tester in the \((i + 1)^{st}\) sample, and assume with out loss of generality that \(|x| = (1/2 - \alpha)d\). Let \( n_1 \) be the number of weight \((1/2 - \alpha)d\) points sampled before \( x \) (that is, points from \( X_1 \)). Denote by \( K \) the number of points in \( D_{1/2}^{d/2} \). Hence, the probability under \( P_1^A \) that the tester receives a sampling of \( x \), is the probability that in a function \( f \in F_1 \) that is consistent with \( S_f(i) \), the point \( x \) belongs to \( X_1 \) and that \( x \) was sampled according to \( D_f \). It is easy to see that the distribution induced by \( P_1^A \) over the choice of prefix for the sampled point, is the uniform distribution defined over the set of prefixes that have not yet appeared in the points sampled prior to \( x \). In addition, once the prefix of \( x \) is determined, the choice of \( x \) is uniformly distributed over all \( K \) prefix \( p \) points. Hence, the probability is \( (\frac{m - n_1}{m - 1}) \cdot (\frac{1}{K}) \), where the first term is the probability that the next point drawn according to \( D_f \) is in \( X_1 \), the second term is the probability that the prefix used for the next point is indeed the prefix \( p_x \) of \( x \) and the third is the probability to choose \( x \) out of all prefix \( p_x \) points. We now turn to bound the same probability with respect to the distribution \( P_2^A \). In the case of functions from \( F_2 \), the distribution induced over the prefix of the drawn point is again uniform, and so is the distribution induced over the possible choices of the point once the prefix has been determined. However, the probability that the drawn point belongs to \( X_1 \) is \( \frac{1}{2} \), implying that the probability to choose \( x \) in this case is \( \frac{1}{2} \cdot \frac{1}{2m - i} \cdot \frac{1}{K} \). Hence, the ratio between the two probabilities is \( \frac{2m - 2n_1}{2m - i} \) and is bounded from above by \((1 + \frac{n}{2m - n})\) and from below by \((1 - \frac{n}{2m - n})\) as required 3.

Therefore, the statistical difference between the two distributions \( P_1^A \) and \( P_2^A \) is bounded by

\[
\frac{1}{2} \left( \sum_{S_f; S_f \text{ is bad}} \max\{P_1^A(S_f), P_2^A(S_f)\} + \sum_{S_f; S_f \text{ is good}} P_2^A(S_f) \cdot \frac{O(n^2)}{m} \right).
\]

Since under both distributions, the probability that the knowledge sequence is bad is exponentially small, the statistical difference between the two distributions is bounded by \( \frac{O(n^2)}{m} \). Implying the following theorem:

**Theorem 7.3.10** There exists a constant \( c \), such that testing monotonicity of boolean functions defined over the boolean \( d \)-dimensional hypercube, in the distribution-free setting, requires \( 2^c \) queries.

**Acknowledgment**

The lower bound proof for monotonicity testing in the high dimensional case was initiated by a discussion with Nader Bshouty. We wish to thank Nader for sharing his ideas with us; they played a significant role in the development of the results presented in this paper.

---

3Note that, \( \frac{2m - 2n_1}{2m - i} = 1 + \frac{2n_1}{2m - i} \). Since \( 0 \leq n_1 \leq i \leq n \), then \( 2m - n \leq 2m - i \), and \( -n \leq i - 2n_1 \leq n \).
Chapter 8

Testing Monotonicity over Graph Products

In this chapter we consider the problem of monotonicity testing over graph products. We present a testing approach that enables us to use known monotonicity testers for given graphs $G_1, G_2$, to test monotonicity over their product $G_1 \times G_2$. We demonstrate the usefulness of our results by showing how a careful use of this approach improves the query complexity of known testers.

The chapter is organized as follows: In Section 8.1, we formally define graph products and state a general framework for the construction of testers for such graphs. In Section 8.2, we focus on monotonicity testing of boolean functions defined over general graph products. In Section 8.3, we study monotonicity testing of general (non-boolean) functions defined over graph products of a line with any other graph, and use our results to give an improved analysis for the known monotonicity tester of [19].

8.1 Preliminaries and general approach

Let $A$ be some linear order. For a directed graph $G = (V, E)$ and functions $f, g : V \to A$, the distance between $f$ and $g$ is $\text{dist}(f, g) \triangleq |\{v \in V : f(v) \neq g(v)\}|/|V|$. Denote by $\mathcal{P}_{\text{mono}}(G, A)$ the class of functions $f : V \to A$ that are monotone with respect to $G$ (i.e., for all $(u, v) \in E$ we have $f(u) \leq f(v)$). For every function $f : V \to A$ defined over $G$, denote by $\epsilon(f)$ the distance of $f$ from monotone; i.e., $\epsilon(f) = \min_{g \in \mathcal{P}_{\text{mono}}(G, A)} \text{dist}(f, g)$; since $G$ and $A$ will be clear from the context, they do not appear in the notation $\epsilon(f)$.

We first define formally the notion of graph product we focus on in this work.

**Definition 8.1.1** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The product of $G_1$ and $G_2$, denoted by $G_1 \times G_2$, is the graph $G = (V, E)$ where:

1. $V = V_1 \times V_2$.
2. $((v_1, u_1), (v_2, u_2)) \in E$ iff one of the following holds:
   
   (a) $(v_1, v_2) \in E_1$ and $u_1 = u_2$; or
   (b) $(u_1, u_2) \in E_2$ and $v_1 = v_2$.

(See Figure 1.1 for an example of a product of two graphs.)
Denote by \([n]\) the line graph; i.e., \(G = ([1, \ldots, n], \{(i, i+1) : 1 \leq i \leq n - 1\})\). It is easy to see that the two dimensional mesh can be viewed as \([n] \times [n]\), and the \(d\)-dimensional hypercube can be viewed as the \(d\)th power of \([n]\).

As mentioned, we are interested in using testers for \(G_1\) and \(G_2\) to construct a tester for \(G_1 \times G_2\). Therefore, we are looking for connections between the distance from being monotone of a function defined over \(G_1 \times G_2\), to the distance from being monotone of the functions it induces on copies of \(G_1\) and \(G_2\). To state our goal more formally, we introduce a few definitions.

**Definition 8.1.2** Given a graph product \(G_1 \times G_2\) of graphs \(G_1 = (V_1, E_1)\) and \(G_2 = (V_2, E_2)\), define for every vertex \(v \in V_1\) the \(v\)-copy of \(G_2\), denoted by \(v \times G_2\), to be the subgraph of \(G_1 \times G_2\) induced by \(\{(v, u) : u \in V_2\}\) (observe that \(v \times G_2\) is isomorphic to \(G_2\)). (See Figure 8.1 for an example).

Given a function \(f : V_1 \times V_2 \rightarrow A\), denote by \(f_v\), for every \(v \in V_1\), the function induced by \(f\) on \(v \times G_2\). Similarly, for a vertex \(v \in V_2\), define the graph \(G_1 \times v\) and the induced function \(f_v\).

![Figure 8.1: b-copy of \(G_1\) in the product \(G_1 \times G_2\) of Figure 1.1](image)

**Definition 8.1.3** Given a function \(f : V_1 \times V_2 \rightarrow A\) defined over \(G_1 \times G_2\), we say that \(f\) is \(G_1\)-monotone if, for every \(v \in V_2\), the function \(f_v\) is monotone. In other words, all the functions induced by \(f\) on copies of \(G_1\) are monotone. The notion of \(f\) being \(G_2\)-monotone is defined similarly. (See Figure 8.2 for an example of a \(G_1\)-monotone and a \(G_2\)-monotone functions.)

![Figure 8.2: The left function, \(f_1\), is \(G_1\)-monotone, while the right function, \(f_2\), is \(G_2\)-monotone. None of \(f_1, f_2\) is \(G_1 \times G_2\) monotone.](image)

Note that neither \(G_1\)-monotonicity nor \(G_2\)-monotonicity imply monotonicity of the function over \(G_1 \times G_2\) (as can be seen in Figure 8.2). The next observation follows immediately our definition of graph product.
Observation 5: A function defined over $G_1 \times G_2$ is monotone if and only if it is both $G_1$-monotone and $G_2$-monotone.

For every function $f : V_1 \times V_2 \to A$, defined over $G_1 \times G_2$, denote by $\epsilon_1(f)$ the expected distance of a function induced by $f$ on a copy of $G_1$ from being monotone; $\epsilon_2(f)$ is defined similarly. That is, $\epsilon_1(f) = E_{e \in V_2} \epsilon(f_e)$ and $\epsilon_2(f) = E_{e \in V_1} \epsilon(f_e)$ (for example, in Figure 8.2, $\epsilon_1(f_1) = 0$ and $\epsilon_2(f_2) = 1/4$). Equivalently, $\epsilon_1(f)$ (respectively, $\epsilon_2(f)$) is the distance of the function $f$ from the class of $G_1$-monotone (respectively, $G_2$-monotone) functions. This is because transforming $f$ into a $G_1$-monotone function can be performed independently on every induced copy of $G_1$ (note that the requirement that a function is $G_1$-monotone is separate for every induced copy of $G_1$ and is independent of the other copies of $G_1$).

We are interested in bounding $\epsilon(f)$ as a function of $\epsilon_1(f)$ and $\epsilon_2(f)$. Specifically, a linear bound may be useful. Before presenting such bounds, we explain why this kind of (linear) bounds will enable us to use the monotonicity testers for $G_1$ and $G_2$ to construct a monotonicity tester for $G_1 \times G_2$. Assume that for some constant $c$, for every function $f$ defined over $G_1 \times G_2$, indeed $\epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f))$. (At first, it may seem as if always $\epsilon(f) \leq \epsilon_1(f) + \epsilon_2(f)$; however, in the next section we show that this is not the case even for boolean functions.)

We present a general testing scheme for $G_1 \times G_2$, using the testers for $G_1$ and $G_2$ as black boxes. Let $T_1$ be a monotonicity tester for $G_1$, and let $Q_1(\epsilon)$ be its query complexity. Similarly, let $T_2$ be a monotonicity tester for $G_2$, and let $Q_2(\epsilon)$ be its query complexity. The scheme appears in Figure 8.3.

```
General_Tester(f, \epsilon)
repeat 4c/\epsilon times:
    choose i \in \{1, 2\}.
    choose v \in V_i uniformly.
    repeat twice - test, using $T_i$, that $f_v$ is monotone with distance parameter $\frac{\epsilon}{2c}$.
    if $T_i$ rejects, then return FAIL.
return PASS
```

Figure 8.3: General testing scheme for graph products

**Theorem 8.1.4** Let $c$ be a constant. Assume that $\epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f))$ for every function $f : V_1 \times V_2 \to A$ defined over $G_1 \times G_2$. Then, $\text{General}\_\text{Tester}(f, \epsilon)$ is a monotonicity tester for functions defined over $G_1 \times G_2$ with query complexity $O(\frac{c^2}{\epsilon^2}(Q_1(\frac{\epsilon}{2c}) + Q_2(\frac{\epsilon}{2c})))$.

**Proof:** To prove the theorem, we claim: (a) the query complexity of the tester is as required; (b) if a function $f : V_1 \times V_2 \to A$ is indeed monotone, then it passes the test with probability $1$; and (c) if $f$ is $\epsilon$-far from being monotone, then with probability of at least $\frac{2}{3}$ it is rejected by the tester. Claim (a) is obvious by the definition of the tester. Claim (b) follows immediately from the fact that if $f$ is monotone, then it is both $G_1$-monotone and $G_2$-monotone (Observation 5). As for Claim (c), by the fact that $\epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f))$, we deduce that the average distance of a function induced by $f$ on either $G_1$ or $G_2$ is at least $\epsilon/2c$. This implies that at least $\epsilon/2c$ of these functions are in distance of at least $\epsilon/2c$ from monotone.

\[1\] Notice that there are two extreme situations that satisfy the condition and both have to fall into the general framework. One situation is that all the functions induced on copies of either $G_1$ or $G_2$ are $\frac{\epsilon}{2c}$-far from monotone, while the other is that $\frac{\epsilon}{2c}$ of the functions are 1-far from monotone, while the others are monotone.
Lemma 8.2.1

Let \( f \) be a function. Note that, in general, transforming \( f \) into a \( G \)-monotone function may damage \( f \)'s monotonicity. However, we show that such a transformation can be done while preserving monotonicity at most \( \frac{1}{3} \).

The proof of Lemma 8.2.1 is based on the following lemma. This lemma shows that if a function \( f \) defined over \( G_1 \times G_2 \) is \( G_1 \)-monotone, then the number of modifications required to transform \( f \) into a \( G_2 \)-monotone function suffices also to transform it into a monotone function. Note that, in general, transforming \( f \) into a \( G_2 \)-monotone function may damage its \( G_1 \)-monotonicity (see Figure 8.4 for an example of a \( G_1 \)-monotone function defined over \( [n] \times [n] \) and a transformation of it to a \( G_2 \)-monotone function that does not preserve \( G_1 \)-monotonicity). However, we show that such a transformation can be done while preserving the \( G_1 \)-monotonicity.

Lemma 8.2.2

Let \( f : V_1 \times V_2 \rightarrow \{0, 1\} \) be a \( G_1 \)-monotone function. If the distance of \( f \) from being \( G_2 \)-monotone is \( \epsilon \), then the distance of \( f \) from monotone is \( \epsilon \). In other words, \( \epsilon(f) = \epsilon_2(f) \).

Proof: First, we claim the existence of a function \( f' \) which is \( G_2 \)-monotone, \( \text{dist}(f, f') = \epsilon \) and every \( G_2 \)-monotone function \( f'' \) with \( \text{dist}(f, f'') = \epsilon \) satisfies \( f'' \leq f' \) (the inequality holds in every position). For this, observe that if a boolean \( G_2 \)-monotone function \( f' \) is \( \epsilon \)-close to \( f \),
then \( \text{dist}(f_v, f'_v) = \epsilon(f_v) \) for every \( v \in V_1 \). In other words, for every vertex \( v \in V_1 \) the function \( f'_v \) is one of the monotone functions which are closest to \( f_v \). Otherwise, if there exists a vertex \( v \in V_1 \) and a monotone function \( f'' \) such that \( \text{dist}(f''_v, f_v) < \text{dist}(f'_v, f_v) \), then it is possible to construct from \( f' \) a new \( G_2 \)-monotone function that is closer to \( f \) than \( f' \) by setting the values of the function at \( v \times G_2 \) to be the same as in \( f''_v \). Therefore, it is enough to show that for every vertex \( v \in V_1 \) there exists a function \( g \) that is maximal among the monotone functions that are close to \( f_v \).

Let \( f_1 \) and \( f_2 \) be two monotone functions that are \( \epsilon(f_v) \)-close to \( f_v \) for some \( v \in V_1 \). Define \( X = \{ u \in V_2 : f'_1(u) = 1 \text{ and } f'_2(u) = 0 \} \) (see Figure 8.5 for an illustration of two functions \( f_1 \) and \( f_2 \) defined over \( \{0, 1\}^d \) and the set \( X \)). Note that \( X \) is a closed set in the sense that for every \( u_1 \) and \( u_2 \) in \( V_2 \), if both \( u_1 \) and \( u_2 \) are in \( X \) then so is every vertex \( u' \) such that \( u_1 \leq u' \leq u_2 \). In addition, based on the monotonicity of \( f_1 \) and \( f_2 \), for every \( u' \notin X \) that is below \( X \) (i.e., there exists a vertex \( u \in X \) such that \( u' \leq u \)), both \( f'_1(u) = 0 \) and \( f'_2(u) = 0 \). Similarly, for every \( u' \notin X \) that is above \( X \) (i.e., there exists a vertex \( u \in X \) such that \( u \leq u' \)), both \( f'_1(u) = 1 \) and \( f'_2(u) = 1 \). Hence, the function \( f'_1 \) obtained from \( f_1 \) by setting the values of all \( v \in X \) to 0 is monotone. Similarly, the function \( f'_2 \) obtained from \( f_2 \) by setting the values of all \( v \in X \) to 1 is monotone. Since the possible minimal distance of a monotone function from \( f \) is \( \epsilon \), we deduce that the distance of \( f'_1 \) and \( f'_2 \) from \( f \) is at least \( \epsilon \). Based on the fact that the distance of both \( f_1 \) and \( f_2 \) from \( f \) is the minimal possible distance for a monotone function, we deduce that the number of values \( u \in X \) such that \( f(v, u) = 1 \) is identical to the number of values \( u \in X \) such that \( f(v, u) = 0 \) (otherwise, the distance of either \( f'_1 \) or \( f'_2 \) from \( f \) is smaller than \( \epsilon \)). Thus, the function obtained from \( f_2 \) by changing the values of the vertices in \( X \) to 1 is a monotone function with the same distance from \( f_v \). The above argument can be applied repeatedly until we obtain a function \( f''_v \) that is maximal among the monotone functions that are close to \( f_v \).

We now prove that \( f' \) is also monotone. By definition, \( f' \) is \( G_2 \)-monotone, hence it is left to show that it is also \( G_1 \)-monotone. To show this, it suffice to show that for every \( v, v' \in V_1 \) such that \( u \leq v \), the functions \( f'_u, f'_{u'} : V_2 \rightarrow \{0, 1\} \) satisfy \( f'_u \leq f'_{u'} \). Denote \( X_u = \{ w : f''_u(w) = 1 \} \) and \( X_{u'} = \{ w : f''_{u'}(w) = 1 \} \). Note that both \( X_u \) and \( X_{u'} \) are closed sets in the sense described above, and so is \( X_u \setminus X_{u'} \). As before, in both functions, all the values above \( X_u \setminus X_{u'} \) are set to 1 and all the values below \( X_u \setminus X_{u'} \) are set to 0. Thus, the function obtained from \( f'_u \) by setting the values of all the points in \( X_u \setminus X_{u'} \) to 0 is monotone. We need to show that \( X_u \setminus X_{u'} = \phi \).

This set consists of points that were all assigned 0 in \( f'_u \). Hence, by the construction of \( f' \) it follows that more than half of these points were assigned 0 in \( f_u \) (otherwise, since all the values above it are set to 1 in \( f''_u \), there is another monotone function \( f'' \) such that \( f''_u < f'_u \) and \( \text{dist}(f''_u, f_v) \leq \text{dist}(f'_u, f_v) \), contradicting the maximality of \( f' \)). Since \( f \) is \( G_1 \)-monotone, this also holds for \( f_u \), meaning that more than half of these points were assigned 0 in \( f_u \). This contradicts the fact that \( f'_u(w) = 1 \) for every \( w \in X_u \setminus X_{u'} \).
Figure 8.6: The function \( f \).

Figure 8.7: Areas \( S, B, R \) and \( L \).

**Remark 8.2.1:** The fact that the functions in question were boolean was used in the above proof to claim that at least half of the points in \( X_u \setminus X_v \) were assigned 0 in \( f_v \). This is not true in case of general range, where there are more than two possible values.

**Proof of Lemma 8.2.1:** Let \( f \) be a boolean function defined over \( G_1 \times G_2 \). By the definition of \( \epsilon_1(f) \), there exists a \( G_1 \)-monotone boolean function \( f' \) defined over \( G_1 \times G_2 \) such that \( \text{dist}(f, f') = \epsilon_1(f) \), and the distance of \( f' \) from being \( G_2 \)-monotone is at most \( \epsilon_2(f) \). Thus, by Lemma 8.2.2, the distance of \( f' \) from being monotone is at most \( \epsilon_1(f) + \epsilon_2(f) \). By symmetry, \( \epsilon(f) \leq \epsilon_2(f) + (\epsilon_1(f) + \epsilon_2(f)) \), implying that \( \epsilon(f) \leq \epsilon_1(f) + \epsilon_2(f) + \min\{\epsilon_1(f), \epsilon_2(f)\} \). \( \square \)

The reader may be tempted to conjecture that the proof unnecessarily pays \( \epsilon_1(f) \) (or \( \epsilon_2(f) \)) twice, and in fact it is possible to show that \( \epsilon(f) \leq \epsilon_1(f) + \epsilon_2(f) \). However, the next example shows that this is not the case. Consider the following boolean function \( f \) defined over \([n]_2\): 

\[
\text{if } \frac{an+1}{2a+b} \leq i \leq \frac{(a+b)n}{2a+b} \text{ or } \frac{an+1}{2a+b} \leq j \leq \frac{(a+b)n}{2a+b}, \text{ see Figure 8.6.}
\]

Clearly, \( \epsilon_1(f) \) and \( \epsilon_2(f) \) both equal \( 2 \cdot ab/(2a + b)^2 \). On the other hand, we prove that \( \epsilon(f) = (a^2 + 2ab)/(2a + b)^2 \) for \( 2b \leq a < \frac{(2+\sqrt{8})b}{2} \). Thus, by setting \( a = 2.41b \), we have \( \epsilon(f) \geq 1.1(\epsilon_1(f) + \epsilon_2(f)) \).

**Proposition 8.2.3** Let \( f \) be the function defined above, and assume \( 2b \leq a < \frac{(2+\sqrt{8})b}{2} \), then \( \epsilon(f) = (a^2 + 2ab)/(2a + b)^2 \).

**Proof:** Denote by \( S \) the upper left square of 1's, by \( B \) its boundary, and by \( L \) and \( R \) the lower and right rectangles of 0's (see Figure 8.7).

Let \( f_m \) be a monotone function such that \( \text{dist}(f, f_m) \) is minimal. We show that \( f_m \) is the function obtained from \( f \) by setting the values of the points in \( S \) to be 0 and the values of

\[ 2\text{The lemma can also be proven using a generalization of the arguments used in [19, Lemma 5].} \]
the points in $R$ and $L$ to be 1. Hence, $\epsilon(f) = \text{dist}(f, f_m) = ((an)^2 + 2abn^2)/((2a + b)n^2$, as required.

The proof is done in two stages. First we show that in every monotone function $f'$ such that $\text{dist}(f, f') = \epsilon(f)$, the values of $f'$ on points in $S \cup B$ must be 0. That is, $f'(i, j) = 0$ for all $i, j \leq (a + b)n$. Then, we prove that in every such function $f'$, the values of all points outside $S \cup B$ must be set to 1.

Stage 1: Let $f'$ be a monotone function $\epsilon(f)$-close to $f$, and assume to the contrary that there exist $i, j \leq \frac{(a+b)n}{2a+b}$ such that $f'(i, j) = 1$. We show that the function $f''$ obtained from $f'$ by setting the values of all points $(i, j) \in S \cup B$ to 0 is also monotone and $\text{dist}(f, f'') < \text{dist}(f, f')$. The monotonicity is obvious, since in $f'$ all the points not in $S \cup B$ are bigger than the points in $S \cup B$, and assigning the minimal possible value 0 to the points in $S \cup B$ cannot violate monotonicity. Consider the set $I = \{(i, j) \in S \cup B | f'(i, j) = 1\}$ and let $(i_r, j_r)$ be the minimal pair in $I$ in lexicographic order. Let $j_c$ be the minimal column index in $I$ and let $(i_c, j_c)$ be the minimal pair in $I$ in lexicographic order with column index $j_c$. We claim that both $(i_r, j_r)$ and $(i_c, j_c)$ belong to $S$ (see Figure 8.8).

Assume that either $(i_r, j_r) \in B$ or $(i_c, j_c) \in B$. Let $(i_0, j_0)$ be the minimal pair in lexicographic order on $I \cap S$. Let $j_1$ be the minimal column index in $I \cap S$, and let $(i_1, j_1)$ be the minimal pair in lexicographic order in $I \cap S$ with column index $j_1$. If $I \cap S = \emptyset$ then we set $i_0 = j_0 = i_1 = j_1 = \frac{(a+b)n}{2a+b}$. Denote by $f_0$ the function obtained from $f'$ by setting all the values $(i, j)$ such that either $i < i_0$ and $j \leq \frac{(a+b)n}{2a+b}$ or $i \leq \frac{(a+b)n}{2a+b}$ and $j < j_1$ to be 0. Clearly, $f_0$ is monotone and $\text{dist}(f, f_0) < \text{dist}(f, f')$, contradicting the minimality of $\text{dist}(f, f')$. Hence, as claimed, $(i_r, j_r), (i_c, j_c) \in S$.

By the definition of $f'$ it holds that $\text{dist}(f, f'') \cdot ((2a + b)n)^2 = x + (an)^2$, where $x$ is the number of points $(i, j) \not\in S \cup B$ such that $f(i, j) \neq f'(i, j)$ (the number of modifications made outside $S \cup B$). On the other hand,

$$
\text{dist}(f, f') \cdot ((2a + b)n)^2 \geq x + [(bn)^2 + (an - (i - 1))bn + (an - (j - 1))bn] + [(i_r - 1)an + (j_c - 1)an - (i_r - 1)(j_c - 1) + ((i_r - 1)(j_c - 1) - (i_r - 1)(j_c - 1))]
$$

where the first term in the inequality is due to the modifications in $B$ and the second term is due to the modifications in $S$. In addition,

$$(an - bn)[(i_r - 1) + (j_c - 1)] - (i_r - 1)(j_c - 1) \geq (an - bn)(2(an - bn)) - (an - bn)^2$$

$$= (an - bn)^2 \geq 0,$$
where the inequality follows the fact that the maximal value of this function is received for 
\( i_x, j_x = an - bn \). Therefore, 
\( \text{dist}(f, f') \cdot ((2a + b)n)^2 \geq x + (bn)^2 + 2abn^2 \). Based on the fact that 
\( a < \frac{(2 + \sqrt{5})b}{2} \), we have 
\( x + (bn)^2 + 2abn^2 > x + (an)^2 \), and therefore 
\( \text{dist}(f, f') > \text{dist}(f, f'') \). Therefore, we conclude that the values of 
\( S_f \) and \( B_f \) in \( f' \) are 0.

**Stage 2:** We show that for every function \( f' \) such that 
\( \text{dist}(f, f') = \epsilon \), it holds that 
\( f'(i, j) = 1 \) for every \((i, j) \notin S \cup B \). The arguments used in this stage are similar to the 
one used in the previous stage. First, observe that in every such function \( f' \), all the points in 
the lower right square must remain with the value 1. We now show that in every such 
function \( f' \) all the points both in \( L \) and in \( R \) must be set to 1. Then, based on this fact, 
we deduce that indeed in every such function \( f' \) all the points in the square to the left of \( L \) 
and the square above \( R \) must keep their value, which completes our proof. We describe 
the proof that in every such function \( f' \), all the points in \( L \) were set to 1. Assume towards a 
contradiction that there exists a point in \( L \) that remains with the value 0, and let \( r \) be the 
maximal row index of such a point. Let \( f'' \) be the function obtained from \( f' \) by setting the 
values of all the points in \( L \) and the square on its left to 1. The function \( f'' \) is monotone 
since, as mentioned before, all the points in the lower right square keep the value 1. By 
the definition of \( f'' \), we have 
\( \text{dist}(f, f'') \cdot ((2a + b)n)^2 = abn^2 + x \), where \( x \) is the number of 
modifications made in \( f' \) in \( R \) and the square above it and in \( S \cup B \). We now examine the 
distance of \( f' \) from \( f \). At least 
\( an \cdot (r - (a + b)n) \) values of points in the square on \( L \)'s left 
were necessarily changed to 0. While, at the same time, at least 
\( ((2a + b) - r)bn \) of the points in \( L \) were changed to 1. Hence, 
\( \text{dist}(f, f') \cdot (an \cdot (r - (a + b)n) + bn \cdot ((2a + b) - r) + x = x + abn^2 + b^2n^2 - a^2n^2 + r(an - bn) \). Since \( r > (a + b)n \) and \( a > b \) we have 
\( abn^2 + b^2n^2 - a^2n^2 + r(an - bn) > abn^2 + b^2n^2 - a^2n^2 + (a + b)n(an - bn) = abn^2 \). Therefore, 
\( \text{dist}(f, f') > \text{dist}(f, f'') \), contradicting the minimal distance of \( f' \). The arguments used in the 
case of \( R \) are similar. \( \square \)

### 8.3 General functions defined over products of the line

This section deals with monotonicity testing of functions with arbitrary range, that are defined 
over a product of the line (that is, \( [n] \)) with another graph. An example for such a graph is 
\( [n]^d \) which can be described as \( [n] \times [n]^{d-1} \). Indeed, the bound presented in Lemma 8.3.1, will 
be used in the new analysis of the monotonicity tester for \( [n]^d \). We prove the following linear bound:

**Lemma 8.3.1** Let \( f \) be a function defined over \( G_1 \times G_2 \), where \( G_1 = [n] \); then, 
\( \epsilon(f) \leq 4\epsilon_1(f) + 3\epsilon_2(f) \leq 4(\epsilon_1(f) + \epsilon_2(f)) \).

Notice that, although this bound is not as good as the one shown in Lemma 8.2.1, it is no 
longer limited to boolean functions.

For simplicity, we begin with the case of \( [n] \times [n] \) (Section 8.3.1), and then generalize our 
argument to a product of \( [n] \) with an arbitrary graph \( G \) (Section 8.3.2). Then, in Section 8.3.3, 
we use Lemma 8.3.1 and specific properties of the monotonicity tester for the line, to 
improve the upper bound on the query complexity of the algorithm of [19] for general functions defined 
over \( [n]^d \). This is done by showing that specific knowledge of the tester may be used in general 
(i.e., not only in the case of \( [n]^d \)) to reduce the query complexity of the tester.
8.3.1 Monotonicity testing for functions defined over $[n]^2$

In this section we prove Lemma 8.3.1 for $[n]^2$ (i.e., when $G_1 = G_2 = [n]$). In this context, we refer to $G_2$-monotone functions as monotone in the first dimension and similarly to $G_1$-monotone functions as monotone in the second dimension. Equivalently, view $f$ as a two-dimensional array; if $f$ is monotone in the first (second) dimension then each row (column) of the array is sorted. As before, notice that the fact that $f$ is monotone in the first (or second) dimension does not imply that $f$ is monotone; however, monotonicity is equivalent to monotonicity in both dimensions. For the proof, we need the following definition and simple lemma (a similar argument was used in [19]).

**Definition 8.3.2** Let $f : V \rightarrow A$ be a function defined over a graph $G = (V,E)$. A pair $(u,v)$ is said to be an $f$-violation if $u <_G v$ (that is, there is a path in $G$ from $u$ to $v$) and $f(u) >_A f(v)$.

**Lemma 8.3.3** Let $f : V \rightarrow A$ be a function defined over a graph $G = (V,E)$. Given $S \subseteq V$, if for every $f$-violation $(i,j)$ either $i \in S$ or $j \in S$, then there exists a monotone function $f'$ that differs from $f$ only on points in $S$.

**Proof:** We show that, by modifying the value of $f$ only on points in $S$, we can obtain a monotone function $f'$. Define $\bar{S} = V \setminus S$. The construction of $f'$ is done through the following iterative process. In each step, modify the value of a point $v \in S$ such that there is no other point $u \leq_G v \in S$ for which the value of $u$ was not yet modified. The process ends after modifying the values of all points in $S$.

The value of a point $v \in S$ is determined in the following manner. We distinguish between two cases: there is no $u \in V$ such that $u <_G v$, and there is a $u \in V$ such that $u <_G v$. In the first case, we set $f'(v) = f(v)$ if $S = V$, and $f'(v) = \min_{u : u \geq_G v , u \in \bar{S}} \{ f(u) \}$ otherwise. In the second case, set $f'(v) = \max_{u : u <_G v} \{ f(u) \}$ (note that the maximum is taken over all points that are smaller than $v$ and not only those in $S$). The values of points in $\bar{S}$ are unchanged (i.e., for all these points $f'(v) = f(v)$).

We claim that $f'$ is indeed monotone, or equivalently that there are no $f'$-violations. Consider a pair of points $u,v$ s.t. $u <_G v$; we will prove that $f'(u) \leq_A f'(v)$. There are three possibilities:

1. $u,v \in \bar{S}$ – since $f'$ equals $f$ for all points not in $S$, and there were no $f$-violations with both endpoints in $\bar{S}$, then $f'(u) = f(u) \leq_A f(v) = f'(v)$.

2. $v \in S$ and $u$ is either in $S$ or in $\bar{S}$ – by the construction of $f'$, the value of $f'$ at point $v$ was set after the value at point $u$ had already been set (since $u <_G v$), and therefore $f'(u) \leq_A f'(v)$.

3. $u \in S$ and $v \in \bar{S}$ – assume to the contrary that $(u,v)$ is an $f'$-violation; moreover, let $(u,v)$ be the minimal such pair (in lexicographic order). Notice that, since we already know that there are no $f'$-violations other than the ones referred to in this case, this is the minimal $f'$-violation. By the definition of $f'$, it is clear that if there is no $u' <_G u \in V$, then $f'(u) = \min_{u' : u' >_G u} \{ f(u') \}$ and therefore $f'(u) \leq_A f(v) = f'(v)$. Hence, if this is not the case then, by the construction of $f'$, the value of $f'$ at point $u$ is the maximal value of $f'$ at points $u'$ s.t. $u' <_G u$. Thus $f'(u) = f'(u')$, for some $u' <_G u$, implying that $(u',v)$ is also an $f'$-violation, contradicting the minimality of $(u,v)$.
Before proving the bound on the distance, we state and prove the following simple combinatorial lemma that is used in the proof.

**Lemma 8.3.4** Given \( B \subseteq [n] \), define a set \( B' \) by the following process: first initialize \( B' = B \); then, for every \( i < j \), if at least half of the values between \( i \) and \( j \) are in \( B \) (i.e., \(| \{ k : k \in B \text{ and } i \leq k \leq j \} \| \geq \frac{j-i+1}{2} \)), set \( B' = B' \cup \{ k : i \leq k \leq j \} \). Then, \( |B'| \leq 3 \cdot |B| \).

**Proof:** Let \( B \) and \( B' \) be as above. We construct a set \( C \subseteq [n] \) with the following two properties: (a) \(|C| \leq 3 \cdot |B| \); and (b) \( B' \subseteq C \). The set \( C \) is constructed from \( B \) as follows. For every \( i \in B \) (in increasing order): denote by \( i_{\min} \) the minimal \( k \) such that \( i < k \) and \( k \) is not yet in \( C \) (if all the points \( k > i \) are in \( C \), set \( i_{\min} = i \)); denote by \( i_{\max} \) the maximal value \( k \) such that \( k < i \) and \( k \) is not in \( C \) (similarly, if all the points \( k < i \) are in \( C \), set \( i_{\max} = i \)); add \( i, i_{\min} \) and \( i_{\max} \) to \( C \).

Since for every point in \( B \) at most three points were added to \( C \), we have \(|C| \leq 3 \cdot |B| \). We show that \( B' \subseteq C \). By the definition of \( C \), we know that \( B \subseteq C \). Let \( i < j \) be a pair of points for which at least half of the points between \( i \) and \( j \) are in \( B \). All the points between \( i \) and \( j \) are in \( B' \) (by definition), and it is left to prove that they are also in \( C \). Assume there is a point \( i \leq k \leq j \) that is not in \( C \). Obviously \( k \) is not a point in \( B \). We examine the points between \( i \) and \( k \) (i.e., the points \( i \leq l < k \)). Since \( k \) is not in \( C \) and since, for each of these points that belongs to \( B \), we added to \( C \) the minimal point above it that was not yet in \( C \), we deduce that at most \( \frac{k-i}{2} \) of these points were in \( B \) (i.e., at most \( \frac{k-i}{2} \) points). Similarly, there are at most \( \frac{j-k}{2} \) points between \( i \) and \( j \) that belong to \( B \). Summing up, we have at most \( \frac{j-k}{2} \) points between \( i \) and \( j \) that belong to \( B \). Contradicting the choice of \( i \) and \( j \).

The next lemma shows that if a function \( f \) is monotone in the first dimension, then if \( x \) value modifications are needed to transform \( f \) into a monotone function in the second dimension, then it is possible to transform \( f \) into a monotone function using \( 3 \varepsilon \) modifications.

**Lemma 8.3.5** Let \( f : [n]^2 \rightarrow A \) be a function which is monotone in the first dimension, and \( \varepsilon \)-far from monotone in the second dimension (i.e., there exists a function \( g \) that is monotone in the second dimension such that \(|\{(a,b) : f(a,b) \neq g(a,b)\}| \leq \varepsilon \cdot n^2 \)). Then, the distance of \( f \) from being monotone is at most \( 3 \cdot \varepsilon \).

**Proof:** By Lemma 8.3.3, it is enough to show that there exists a set of points \( Y \) for which the following two conditions hold:

- \((Y_1)\) - For every \( f \)-violation \( ((a_1,b_1),(a_2,b_2)) \) at least one of \((a_1,b_1)\) and \((a_2,b_2)\) is in \( Y \).
- \((Y_2)\) - \( |Y| \leq 3 \cdot \varepsilon \cdot n^2 \).

Since \( f \) is monotone in the first dimension, there are no \( f \)-violations of the form \( ((a_1,b_1),(a_2,b_2)) \). Hence, there are two kinds of \( f \)-violations: pairs of the form \( ((a_1,b),(a_2,b)) \) with \( a_1 < a_2 \), which we refer to as vertical \( f \)-violations, and pairs of the form \( ((a_1,b_1),(a_2,b_2)) \) with \( a_1 < a_2 \) and \( b_1 < b_2 \), which we refer to as diagonal \( f \)-violations.

Since \( f \) is \( \varepsilon \)-far from monotone in the second dimension, there is a set of points \( X \subseteq [n]^2 \) of size at most \( \varepsilon \cdot n^2 \) and a function \( g \) that is monotone in the second dimension such that, for every point \((a,b)\), if \( g(a,b) \neq f(a,b) \) then \((a,b) \in X \). Also, for every vertical \( f \)-violation \( ((a_1,b),(a_2,b)) \), either \((a_1,b)\) or \((a_2,b)\) is in \( X \). Before constructing the set \( Y \), based on \( X \) and the set of \( f \)-violations, we need the following observation: let \( ((a_1,b_1),(a_2,b_2)) \) be a diagonal \( f \)-violation, hence \( f(a_1,b_1) >_A f(a_2,b_2) \) and let \( b_1 \leq b \leq b_2 \). Since \( f \) is monotone...
in the first dimension we have \( f(a_1, b_1) \leq_A f(a_1, b) \) and \( f(a_2, b) \leq_A f(a_2, b_2) \). Therefore, \( f(a_2, b) <_A f(a_1, b) \), implying that \( ((a_1, b), (a_2, b)) \) is a vertical \( f \)-violation; i.e., either \((a_1, b)\) or \((a_2, b)\) is in \( X \). The reader may be tempted to think that by this \( X \) actually satisfies the requirements for \( Y \). However, although \( X \) contains at least one endpoint of every vertical violation, it is possible that for some diagonal violations neither of its endpoints belongs to \( X \) (see Figure 8.9, where the values that belong to \( X \) are surrounded by circles). The construction of \( Y \) is as follows:

1. Initialize \( Y = X \).
2. For every diagonal \( f \)-violation \( ((a_1, b_1), (a_2, b_2)) \), consider the set \( \{ b_1 \leq b \leq b_2 : (a_1, b) \in X \} \). If the size of this set is more than \( \frac{b_2 - b_1 + 1}{2} \) (i.e., for more than half of the values \( b \) between \( b_1 \) and \( b_2 \) the point \((a_1, b)\) is in \( X \)) then set \( Y = Y \cup \{ (a_1, b) : b_1 \leq b \leq b_2 \} \). Otherwise, set \( Y = Y \cup \{ (a_2, b) : b_1 \leq b \leq b_2 \} \).

We prove that \( Y \) satisfies the two conditions \( Y_1 \) and \( Y_2 \):

- \((Y_1)\): We need to show that every \( f \)-violation has (at least) one endpoint in \( Y \). As for the vertical violations – since \( X \subseteq Y \) and since every vertical \( f \)-violation has at least one end-point in \( X \), we are done. Therefore, let \( ((a_1, b_1), (a_2, b_2)) \) be a diagonal \( f \)-violation. By step 2 in \( Y \)'s construction, either \((a_1, b_1)\) or \((a_2, b_2)\) is in \( Y \).

- \((Y_2)\): We need to prove that \(|Y| \leq 3 \cdot |X| \). Applying Lemma 8.3.4 to each row, we conclude that the number of \( Y \)-points in each row is at most three times the number of \( X \)-points in that row.

Lemma 8.3.1 for \([n]^2\) can now be proved in a similar way to the proof of Lemma 8.2.1, where Lemma 8.3.5 replaces Lemma 8.2.2.

### 8.3.2 Monotonicity testing over products of the line

In this section we prove Lemma 8.3.1 for the general case, where the functions are defined over \([n] \times G_2\) (the same can be done for the domain \( G_1 \times [n] \)). The proof is similar to the proof given for the \([n]^2\) case. In addition to the obvious use of this lemma to construct testers for
functions defined over products of the line, it will also be used in the next section to provide a new analysis for the monotonicity tester for $[n]^d$.

The following lemma is a generalization of Lemma 8.3.5. The proof of Lemma 8.3.1 follows easily from it (just as was the case with Lemma 8.3.5). We say that a function $f$ is monotone in the first dimension if it is $G_1$-monotone.

**Lemma 8.3.6** Let $f$ be a function defined over $[n] \times G_2$, for an arbitrary graph $G_2 = (V_2, E_2)$. If $f$ is $[n]$-monotone then $\epsilon(f) \leq 3\epsilon_2(f)$.

**Proof:** As in the proof of Lemma 8.3.5, based on Lemma 8.3.3, we construct a set $Y \subseteq [n] \times V_2$ of size at most $3 \cdot \epsilon_2(f) \cdot n \cdot |V_2|$, that contains at least one end-point of every $f$-violation. We again distinguish between two kinds of $f$-violation: vertical violations (of the form $((i, u), (i, v))$ for $u <_{G_2} v$) and diagonal ones (of the form $((i, u), (j, v))$ for $i < j$ and $u <_{G_2} v$).

Since $f$ is $\epsilon_2(f)$ far from $G_2$-monotone, there is a set of points $X$ of size at most $\epsilon_2(f) \cdot n \cdot |V_2|$, that contains at least one end-point of every vertical $f$-violation. We observe the following facts:

- Let $((i, u), (j, v))$ be a diagonal $f$-violation. Hence, $f((i, u)) >_A f((j, v))$. Based on the $[n]$-monotonicity of $f$, for every $i \leq l \leq j$, $f((i, u)) \leq_A f((l, u))$ and $f((l, v)) \leq_A f((l, v))$, implying that $((l, u), (l, v))$ is a vertical $f$-violation. This leads to the next fact.

- For every diagonal $f$-violation $((i, u), (j, v))$ and $i \leq l \leq j$, either $(l, u)$ or $(l, v)$ is in $X$.

We now build the set $Y$ based on the set $X$:

1. initialize $Y = X$.

2. for every diagonal $f$-violation $((i, u), (j, v))$:
   - if for more than half of the values $i \leq l \leq j$, the point $(l, u)$ is in $X$, add to $Y$ the set $\{(l, u) : i \leq l \leq j\}$.
   - otherwise, add to $Y$ the set $\{(l, v) : i \leq l \leq j\}$.

By the construction of $Y$ from $X$, every $f$-violation has at least one end-point in $Y$ (the vertical violations due to the first step of the construction and the diagonal ones due to the second step). As for the size of $Y$, the extension of $Y$ into $X$ can be seen as applying Lemma 8.3.4 to each one-dimensional function induced by $f$, and hence the number of $Y$ points can be at most three times the size of $X$. We have shown the existence of a set $Y$ of size at most $3\epsilon_2(f) \cdot n \cdot |V_2|$ that contains at least one end-point of every $f$-violation. By Lemma 8.3.3, the claim follows.

**8.3.3 Testing monotonicity over $[n]^d$**

In this section we use the result shown in the previous section to provide a new analysis of the monotonicity tester presented by [19], based on dimension reduction, and by this improve the known upper bound on the query complexity of the algorithm. The main difference between the dimension reduction introduced here and the one introduced in [19] is that our reduction deals directly with functions with general range, while the approach in [19] deals with the
boolean case first and then generalizes to an arbitrary range using a general transformation presented in that work.

First, we consider the tester that follows directly from the general framework presented in Section 8.1. Then, using specific knowledge of the monotonicity tester for the line, we significantly improve the query complexity of the tester.

By successive applications of our general scheme (presented in Section 8.1), one can obtain a monotonicity tester for $[n]^d$ with query complexity exponential in $\frac{1}{\epsilon}$ (which is undesirable). A different possible approach may be to successively use Lemma 8.3.1 to reduce the testing problem to the one-dimensional case, as was done in [19]; it is possible to show that if a function $f$ defined over $[n]^d$ is $\epsilon$-far from monotone, then the expected distance of a one-dimensional function $f'$ induced by $f$ (by fixing $d - 1$ coordinates and allowing the remaining coordinate to range from 1 to $n$) from being monotone is at least $\epsilon' = \frac{\epsilon}{d\epsilon^{1-d}}$. Based on this, a possible testing strategy is the following: randomly choose a one-dimensional function $f'$ induced by $f$ (i.e., randomly set $d - 1$ coordinates), and test the function $f'$ using the monotonicity tester that was presented in [20] (with $f', \epsilon'$). Due to the expected distance of such a one-dimensional function $f'$ from being monotone, at least an $\frac{d\epsilon^{1-d}}{\epsilon}$ fraction of the one-dimensional functions induced by $f$ are $\frac{d\epsilon^{1-d}}{\epsilon}$-far from monotone; therefore, randomly choosing $O(d\epsilon^{1-d})$ such lines and testing monotonicity on each of them, yields a monotonicity tester for $[n]^d$.

However, the query complexity of this tester is $O(d^2 \frac{16d \log n}{\epsilon^2})$ (in particular, it has a quadratic dependence on $\frac{1}{\epsilon}$). Our goal is to get linear dependence on $\frac{1}{\epsilon}$ (notice that for the one-dimensional case the algorithm presented in [20] already achieves this goal). As stated before, the general approach assumes no knowledge of the testers, and uses them as black boxes. However, in this case we use specific properties of the tester to get a significant improvement.

The one-dimensional monotonicity tester of [20] can be viewed as based on picking pairs of points (according to some distribution, denoted by $\mathcal{P}$) and looking for a violation of monotonicity. This observation leads to the following different approach: in each phase of the algorithm pick a line, but rather than applying to it the full one-dimensional tester, choose only one pair of points on this line (according to the distribution $\mathcal{P}$). It turns out that this approach enables us to use less queries; specifically, $O(d \frac{d\log n}{\epsilon})$ queries suffice. We show that this can be used to reduce the query complexity in many other cases as well.

Below we focus on a special kind of testers: “edge tests” with query complexity linear in $\frac{1}{\epsilon}$. We first define the notion of a \textit{linear edge test} more formally.

\textbf{Definition 8.3.7} A monotonicity tester $T$, for $G = (V, E)$ and range $A$, is said to be an edge test if $T$ works by repeatedly picking an edge $e \in E$ according to some distribution $\mathcal{P}$, and looking for a violation of monotonicity between the two endpoints of $e$. We say that $T$ is linear if whenever $f$ is $\epsilon$-far from monotone, then the probability of picking an $f$-violation of monotonicity using $\mathcal{P}$ is a linear function of $\epsilon$. We refer to this probability as the detection probability of $T$ and denote it by $P_T(e)$.

A linear monotonicity edge test for $[n]$ can be found at [19]. For the sake of completeness, we present a slightly different version of the tester of [20] that is a linear edge tester (See Figure 8.10).

Denote by $\mathcal{P}$ the probability measure induced by the tester on edges. The same arguments used to analyze the original algorithm [20] imply:

\textbf{Proposition 8.3.8} Let $f : [n] \to A$ be a function $\epsilon$-far from monotone. Then,

$$\Pr\{f(m) >_A f(l) | (m, l) \sim \mathcal{P}\} \geq \frac{\epsilon}{8 \cdot \log n}.$$
Algorithm monotone\((f, \epsilon)\):
\[
\text{repeat } \frac{1}{16} \cdot \log n \text{ times}
\]
\[
\text{choose } i \in R[n]
\]
\[
\text{choose } b \in R \{0, 1\}
\]
\[
\text{if } b = 1
\]
\[
\text{choose } k \in R[\lceil \log i \rceil]
\]
\[
\text{choose } j \in R[2^k]
\]
\[
\text{if } f(i - j) > A f(i) \text{ then return FAIL}
\]
\[
\text{else}
\]
\[
\text{choose } k \in R[\lceil \log(n - i) \rceil]
\]
\[
\text{choose } j \in R[2^k]
\]
\[
\text{if } f(i) > A f(i + j) \text{ then return FAIL}
\]
return PASS

Figure 8.10: Linear monotonicity edge test for the line

We show that, for the special case of linear edge tests, lower query complexity can be achieved. Then, we show that for the special case of powers of graphs, this scheme can be further improved. The result for \([n]^d\) will be obtained as a special case of graph powers.

General testing scheme for linear edge tests:

Given \(G_1 = (V_1, E_1)\) and \(G_2 = (V_2, E_2)\), let \(T_1\) be a linear monotonicity edge test for \(G_1\) with detection probability \(P_{T_1}(\epsilon)\) and query complexity \(Q_{T_1}(\epsilon)\), and let \(P_1\) be the distribution on \(E_1\) according to which \(T_1\) picks the edges. Similarly, let \(T_2\) be a linear monotonicity edge test for \(G_2\) with detection probability \(P_{T_2}(\epsilon)\) and query complexity \(Q_{T_2}(\epsilon)\) that picks the edges in \(E_2\) according to \(P_2\). The testing scheme for functions defined over \(G_1 \times G_2\) appears in Figure 8.11.

Algorithm Linear_Tester\((f, \epsilon)\):
\[
\text{repeat } 2c \max\{Q_1(\epsilon), Q_2(\epsilon)\} \text{ times:}
\]
\[
\text{choose } i \in \{1, 2\},
\]
\[
\text{choose } v \in V_i \text{ uniformly.}
\]
\[
\text{choose } e \in E_i \text{ according to } P_i.
\]
\[
\text{check both endpoints of the edge corresponding to } e \text{ in } f_v.
\]
\[
\text{if a violation is found then return FAIL.}
\]
return PASS

Figure 8.11: Testing scheme for linear edge tests.

**Theorem 8.3.9** Let \(G_1, G_2, T_1\) and \(T_2\) be as above and let \(c\) be a constant. Assume that \(\epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f))\) for every function \(f : V_1 \times V_2 \to A\) defined over \(G_1 \times G_2\). Then, Linear_Tester\((f, \epsilon)\) is a monotonicity tester for functions defined over \(G_1 \times G_2\) with query complexity \(O(c(Q_1(\epsilon) + Q_2(\epsilon)))\).

**Proof:** Denote Linear_Tester by \(T\), and note that \(T\) is an edge test. The query complexity \(Q_T(\epsilon)\) of \(T\) follows immediately the definition. In addition, if \(f\) is indeed monotone, then
clearly it is accepted by the tester $T$ with probability 1. Hence, it remains to show that if $f$ is $\epsilon$-far from monotone, then it is rejected by $T$ with high probability. To do so, we first show that $P_T(\epsilon) \geq \frac{1}{2} \min \{P_{T_1}(\epsilon), P_{T_2}(\epsilon)\}$. Based on the definition of $T$, the assumption, and the fact that both $P_{T_1}$ and $P_{T_2}$ are linear functions of $\epsilon$, we have,

$$P_T(\epsilon) = \frac{1}{2} P_{T_1}(\epsilon_1) + \frac{1}{2} P_{T_2}(\epsilon_2) \geq \frac{1}{2} \min \{P_{T_1}(\epsilon_1), P_{T_2}(\epsilon_1)\} + \frac{1}{2} \min \{P_{T_1}(\epsilon_2), P_{T_2}(\epsilon_2)\} = \frac{1}{2} \min \{P_{T_1}(\epsilon + \epsilon), P_{T_2}(\epsilon + \epsilon)\} \geq \frac{1}{2} \min \{P_{T_1}(\epsilon'), P_{T_2}(\epsilon')\} = \frac{1}{2C} \min \{P_{T_1}(\epsilon), P_{T_2}(\epsilon)\}.$$

The probability that a function $f$ that is $\epsilon$-far from monotone is accepted by $T$, is at most $(1 - P_T(\epsilon))^{Q_T(\epsilon)} \leq (1 - \frac{1}{2C} \min \{P_{T_1}(\epsilon), P_{T_2}(\epsilon)\})^{2c \max \{Q_{T_1}(\epsilon), Q_{T_2}(\epsilon)\}}$. Assuming w.l.o.g. that $P_{T_1}(\epsilon) \leq P_{T_2}(\epsilon)$, we have $(1 - \frac{1}{2C} \min \{P_{T_1}(\epsilon), P_{T_2}(\epsilon)\})^{2c \max \{Q_{T_1}(\epsilon), Q_{T_2}(\epsilon)\}} = (1 - \frac{1}{2C} P_{T_1}(\epsilon))^{2cQ_{T_1}(\epsilon)}$. For some constant $C \leq 1$. Based on the fact that $T_1$ is an edge test for functions defined over $G_1$, we deduce that the above probability is bounded by $\frac{1}{3} \cdot C \leq \frac{1}{3}$.

Successive application of this testing scheme for $[n]^d$ yields a monotonicity tester with query complexity of $O(\frac{\epsilon^d \log n}{\epsilon})$ and running time of $O(\frac{\epsilon^d \log^2 n}{\epsilon})$. This query complexity is indeed linear in $\frac{1}{\epsilon}$; however, as stated above, in the case of graph powers, $G^d$, a better bound (as a function of $d$) on the query complexity can be achieved.

**General testing scheme for linear edge tests over graph powers:**

Let $G = (V, E)$ be a graph. We wish to reduce the monotonicity testing problem of functions defined over $G^d$ to the problem of monotonicity testing over $G$. We first define the functions induced by $f$ on copies of $G$.

**Definition 8.3.10** Given a function $f : V^d \to A$, define the set of one-dimensional functions induced by $f$. For every coordinate $i \in [d]$ and $d - 1$ vertices $v_1, \ldots, v_{d-1} \in V$, the function $f_{v_1,\ldots,v_{d-1}}^i : V \to A$ is defined in the following manner: given a point $u \in V$, set $f_{v_1,\ldots,v_{d-1}}^i(u) = f(v_1, \ldots, v_{i-1}, u, v_i, \ldots, v_{d-1})$.

Let $T$ be a monotonicity linear edge test for $G$, and let $P$ be the distribution according to which $T$ picks the edges in $E$. Denote by $Q_T(\epsilon)$ the query complexity of $T$ and by $P_T(\epsilon)$ its detection probability. The tester is presented in Figure 8.12.

```
Power(f, \epsilon)
repeat \epsilon^{-d} dQ_T(\epsilon) times:
    choose v_1, \ldots, v_{d-1} \in_R V, i \in_R [d]
    choose e = (u, u') \in E according to P.
    if f_{v_1,\ldots,v_{d-1}}^i(u) > \epsilon f_{v_1,\ldots,v_{d-1}}^i(u') then return FAIL
return PASS
```

Figure 8.12: Testing scheme over graph powers
The proof of the above tester is based on the fact that if \( f \) is indeed \( \epsilon \)-far from monotone, then the functions induced by \( f \) on copies of \( G \) are not likely to be too close to monotone. Before formalizing this claim, we need the following notation. For every function \( f_{v_1,\ldots,v_{d-1}} \) induced by \( f \), denote by \( \epsilon'_{v_1,\ldots,v_{d-1}}(f) \) its distance from monotone. Denote by \( \epsilon_{1D}(f) \) the expected value of \( \epsilon'_{v_1,\ldots,v_{d-1}}(f) \); i.e., \( \epsilon_{1D}(f) = E_{v_1,\ldots,v_{d-1}\in [n]}[\epsilon'_{v_1,\ldots,v_{d-1}}(f)] \). The correctness proof is based on the following lemma.

**Lemma 8.3.11** Let \( c \) be a constant such that for every graph \( G_2 = (V_2, E_2) \) and for every function \( f : V \times V_2 \rightarrow A \) defined over \( G \times G_2 \), it holds that \( \epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f)) \). Given a function \( f : V^d \rightarrow A \) defined over \( G^d \) that is \( \epsilon \)-far from monotone, then \( \epsilon_{1D}(f) \geq \frac{\epsilon}{c^d - 1} \).

**Proof:** For the sake of the proof, we generalize the notion of functions induced by \( f \) to an arbitrary dimension. Given a function \( f : V^d \rightarrow A \), for every set \( B \subseteq [d] \) and values \( v_1, \ldots, v_{|B|} \in V \) we derive from \( f \) the function \( f_{v_1,\ldots,v_{|B|}}^B : V^{d-|B|} \rightarrow A \), by fixing the values of the coordinates in \( B \) to the values \( v_1, \ldots, v_{|B|} \), and allowing the other coordinates to range over the possible values in \( V \). We say that \( f \) is \( B \)-monotone if for every choice of \( v_1, \ldots, v_{|B|} \), the function \( f_{v_1,\ldots,v_{|B|}}^B \) is monotone. Note that the notion of \( B \)-monotonicity is different than the notion of \( G_1 \)-monotonicity presented before, since \( B \subseteq [d] \). However, \( G_1 \)-monotonicity for the case \( G^d = G \times G^{d-1} \) can be described as \( \{2, \ldots, d\} \)-monotonicity, and \( G_2 \)-monotonicity as \( \{1\} \)-monotonicity (in some sense, these notations are complementing, since the indexes that appear in the set are the ones that we choose their values, and not the free ones).

Denote by \( \epsilon(f) \) the distance of \( f \) from being monotone as before, and define \( \epsilon_B(f) = E_{v_1,\ldots,v_{|B|}\in [n]}[\epsilon(f_{v_1,\ldots,v_{|B|}}^B)] \). Equivalently, \( \epsilon_B(f) \) is the distance of \( f \) from being \( B \)-monotone. By definition, for every set \( B \), there exists a \( B \)-monotone function \( g_B \) whose distance from \( f \) is \( \epsilon_B(f) \).

Since \( G^d = G \times G^{d-1} \), then by the assumption in the lemma \( \epsilon(f) \leq c \cdot (\epsilon_{\{2,\ldots,d\}}(f) + \epsilon_{\{1\}}(f)) \). Since \( \epsilon_{\{1\}} \) is the expected value of \( \epsilon(f^{\{1\}}) \), by applying the assumption again, we have \( \epsilon_{\{1\}}(f) \leq c \cdot (\epsilon_{\{1,2\}}(f) + \epsilon_{\{1,3,\ldots,d\}}(f)) \). Successive application of this argument yields the following inequalities:

\[
\epsilon(f) \leq c \cdot (\epsilon_{\{2,\ldots,d\}}(f) + \epsilon_{\{1\}}(f)) \\
\leq c \cdot (\epsilon_{\{2,\ldots,d\}}(f) + c \cdot (\epsilon_{\{1,3,\ldots,d\}}(f) + \epsilon_{\{1,2\}}(f))) \\
\vdots \\
\leq c^{d-1} (\sum_{i=1}^{d} \epsilon_{\{d\setminus i\}}(f)) \\
= c^{d-1} \cdot d \cdot E_{i}[\epsilon_{\{d\setminus i\}}(f)] \\
= c^{d-1} \cdot d \cdot \epsilon_{1D}(f).
\]

We can now state and prove the correctness of the tester \( \text{Power} \).

**Theorem 8.3.12** Let \( T \) be a linear edge test for \( G \), and let \( c \) be a constant such that for every graph \( G_2 = (V_2, E_2) \) and for every function \( f : V \times V_2 \rightarrow A \) defined over \( G \times G_2 \), it holds that \( \epsilon(f) \leq c(\epsilon_1(f) + \epsilon_2(f)) \). Then, \( \text{Power}(f, \epsilon) \) is a monotonicity tester for functions defined over \( G^d \) with query complexity \( O(dc^{d-1}Q_T(\epsilon)) \).
Applying the above theorem for \(|n|^d\) implies an improved upper bound on the query complexity of the monotonicity tester presented in [19].

**Proof:** By the definition of the algorithm, its query complexity is \(O(d \cdot c^{d-1}Q_T(\epsilon))\) as required. It is left to prove that it is indeed a monotonicity tester. If \(f\) is monotone then clearly, by the definition of the tester, it is accepted by the tester with probability 1. Assume from now on that \(f\) is \(\epsilon\)-far from monotone; we show that it is rejected by the tester with probability of at least \(\frac{2}{3}\). For this, we prove that the probability of each iteration of the algorithm to pick an \(f\)-violation is at least \(\frac{P_T(\epsilon)}{dc^{d-1}}\); therefore, the probability of the algorithm to fail to detect an \(f\)-violation during the \(c^{d-1}dQ_T(\epsilon)\) iterations it performs, is at most \((1 - \frac{P_T(\epsilon)}{dc^{d-1}})^{c^{d-1}dQ_T(\epsilon)} \sim (1 - P_T(\epsilon))^{Q_T(\epsilon)} \leq \frac{1}{3}\), where the equivalence follows \(T\)’s linearity and the inequality follows the fact that \(T\) is a tester.

It remains to bound the probability of the algorithm to select an \(f\)-violation.

\[
\Pr_{i, v_1, \ldots, v_{d-1}} \{f^i_{v_1, \ldots, v_{d-1}}(u) > A f^i_{v_1, \ldots, v_{d-1}}(u')\} = E_{i, v_1, \ldots, v_{d-1}} [\Pr_{(u, u') \sim P} \{f^i_{v_1, \ldots, v_{d-1}}(u) > A f^i_{v_1, \ldots, v_{d-1}}(u')\}]
\]

\[
= E_{i, v_1, \ldots, v_{d-1}} [P_T(\epsilon_{v_1, \ldots, v_{d-1}}(f))]
\]

\[
= P_T(\epsilon_{ID}(f)) \geq \frac{P_T(\epsilon)}{dc^{d-1}}.
\]

The last equality follows \(P_T\)’s linearity and the inequality is by Lemma 8.3.11. \(\Box\)
Chapter 9

Open problems

In this work we introduce the first distribution-free testers for some of the central problems studied in the property testing literature: low-degree multivariate polynomials testing, monotonicity testing in the low dimensional case and connectivity of sparse graphs. We then generalize the testing scheme for the low-degree multivariate polynomials to give sufficient conditions for the existence of a distribution-free testers. By this, we answer a natural question that has already been raised explicitly by Fischer [22, Subsection 9.3] and is implicit in [28]. In addition, by showing a lower bound on the query complexity required for distribution-free monotonicity testing in the high dimensional case, we show that distribution-free testing, even if possible with non-trivial query complexity, cannot always be done using similar query complexity to the one used in the uniform setting.

However, there are still many open questions with respect to distribution-free testing. The first problem that remains open is trying to narrow the gap between the query complexity of the known distribution-free monotonicity tester for the high dimensional case (which is exponential in $\log n$) and our lower bound (which is only exponential in $d$). In addition, now that we already know that it is possible to construct distribution-free testers for non-trivial problems, we wish to further study existence of such testers for different problems that we know how to test with respect to the uniform distribution. Eventually, our goal will be to try and find characterizations (and not only sufficient conditions) for problems that can be efficiently tested in a distribution-free manner, given that they can be efficiently tested with respect to the uniform distribution.

Another interesting possible direction is to try and relax the requirements for distribution-free testing. There are certain problems, for which testers exist for the uniform distribution case, and it has been proven that they cannot be efficiently tested in the distribution-free setting. Among these problems are the partition problems in the dense graph model that were studied in [28]. In these cases, it is interesting to try and relax the distribution-free testing requirement, by allowing a stronger oracle to the input function $f$. One possibility is to enable the algorithm to ask queries of the form: is there a value $x$ in a sub-domain $X' \subseteq X$ of the function $f$, for which $f(x) = y$ (for a specific value of $y$)? This kind of oracle access seems relevant, for example, in monotonicity testing, when given a point $z \in X$ such that $f(z) = 1$, we wish to find a point $x \in \{v \in X : z \leq v\}$ such that $f(x) = 0$, and it was already introduced in the context of learning theory. Notice that, the strength of the oracle is determined by the kind of specification we allow the algorithm to give on the sub-domains. Another possibility, is to allow the algorithm to ask the actual probability of the points it sampled according to $D$ (in the distribution-free model the tester has no knowledge of the probability of the sampled points), and to use it in its decision process. Finally, there is
always the possibility of distribution-known testing, in which the distribution $D$ is known to the tester in advance. This is very different from the distribution-free case, where we are only allowed to sample according to the distribution $D$, but have no actual knowledge of $D$. 
Bibliography


[21] E. Fischer, On the strength of comparisons in property testing, manuscript (available at ECCC 8(8): (2001)).


