Monotonicity Testing for Boolean Functions over Graph Products

by

Zhengkun Chen

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

Waterloo, Ontario, Canada, 2020

Abstract

We establish a directed analogue of Chung and Tetali's isoperimetric inequality for graph products. We use this inequality to obtain new bounds on the query complexity for testing monotonicity of Boolean-valued functions over products of general posets.

Acknowledgements

I would first like to thank my supervisor, Eric Blais, for his patient guidance and encouragement. It is a pleasure to work with him; I would like to thank Amit Levi and Nathan Harms for their help throughout my masters; I would also like to my readers Gautam Kamath and Lap Chi Lau.

Special thanks to my friends: Chaser, Laetitia, Ming, Peter, Robert, TP, Xingyu, Yuki, and Zichuan for their support; my cat A-Mao for his unconditional companionship; bubble tea places in town; and lastly my parents for their support as well.

Dedication

For my grandmother Guoyin Gao.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Property Testing

The notion of property testing was first coined by Rubinfeld and Sudan [29]. It can be seen as a relaxation of the classic decision problem, where we want to determine if an object has a property or not. By introducing a gray area where mistakes are allowed, we want the property testing algorithm to be more efficient.

General property testing can be seen as the following problem: given query access to a function, the tester wants to determine if the function possess some certain property or far from having that property, where a small probability of failure is allowed.

There a few things we need to specify:

1. What types of queries are allowed?
2. How do we measure the distance from a property if the object does not possess the property?

For item 1, it can be either oracle access to a set of points chosen by the tester or a set of random points given by the oracle itself. In this thesis, we only focus on the setting where the points are chosen by the tester.

As for item 2, there can be different measures depending on the type of object we want to test. For boolean functions, it is common to use the number of mismatches between two functions (over the same domain) as the distance measure, and then the distance to having

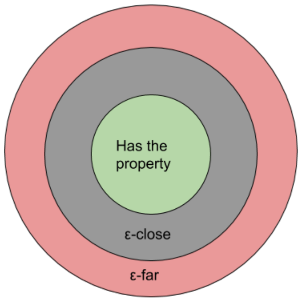

Figure 1.1: A property testing algorithm is allowed to make any mistakes in the gray area

some certain property is the smallest distance between the function being tested and any function that possesses the property. Therefore we often represent the property using the set of objects that have the property.

In the work of Goldreich, Goldwasser, and Ron [20], the authors provide a framework to extend the domain of testing to more complex combinatorial objects including graphs. They also make a connection between property testing and probably approximately correct (PAC) learning, where testing can offer new perspectives towards computational learning theory.

The formal definition of a testing algorithm is the following.

Definition 1 (Testing Algorithm)**.** Let P be a set of functions that map from an arbitrary domain X to $\{0, 1\}$. A randomized algorithm $\mathcal{A}(f)$ is an *ϵ-tester* for property P if and only if

- For all $f \in P$, $\Pr[\mathcal{A}(f) = 1] \geq 2/3$
- For all $f \notin P$ such that $\mathrm{dist}(f, P) > \epsilon$, $\Pr[\mathcal{A}(f) = 0] \geq 2/3$

Note that the definition of $\mathrm{dist}(\cdot, \cdot)$ and relevant ones are in Chapter 2.

1.2 Monotonicity Testing

For any directed acyclic graph G, a function $f : G \to \{0,1\}$ is *monotone* if $f(u) \leq f(v)$, for every edge $(u,v) \in E(G)$. A function f is *ϵ-far from monotone* for some parameter $\epsilon > 0$ if at least ϵ fraction of the points on G whose corresponding values in f need to be changed to obtain a monotone function.

The following definition is often used when measuring the efficiency of a testing algorithm for monotonicity.

Definition 2 (Query complexity of monotonicity)**.** The *query complexity of monotonicity* $Q_\epsilon(G)$ is the minimum number of queries to an unknown function $f : G \to \{0,1\}$ that a bounded-error randomized algorithm requires to distinguish monotone functions from functions that are ϵ-far from monotone

The *monotonicity testing* problem for Boolean-valued functions is the following: for a given directed acyclic graph G, what is $Q_\epsilon(G)$?

This is one of the most well-studied problems in the field of property testing. We want to determine if an object has a global property (monotonicity) by only utilizing local query (values of the function).

The most naive solution is to query every single value of the function, and we reject the function as being monotone if there is a non-monotone pair of points. This will result in a query complexity that equals to the size of the domain. As it turns out, we can do significantly better if we allow some margin of error and use the structural property of the domain.

For those who are familiar with randomized algorithm, one might attempt to just sample random edges by querying only adjacent points in the domain G. This is also known as the *edge test*.

Definition 3 (Edge test)**.** An *edge test* is the simple test that chooses an edge uniformly at random from the domain G and queries the two end points of the edge and rejects the function if these two end points form a violation of monotonicity.

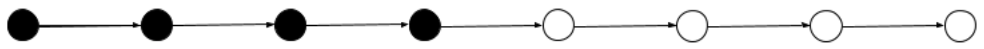

Figure 1.2: The anti-majority function on a total over of length 8. Black points have value 1 and white points have value 0 on the function.

Algorithm 1: Edge test algorithm

```
1 **Input:** a DAG $G$, a function $f : G \to \{0,1\}$, an distance parameter $\epsilon$
2 **Output:** **Accept** or **Reject**
3 Draw an edge $(u,v)$ uniformly at random from $G$.
4 Query $f(u)$ and $f(v)$.
5 **If** $f(u) \le f(v)$: **Accept** $f$
6 **Else**: **Reject** $f$
```

Based on this definition of the edge test, we also have the definition for an *edge tester*.

Definition 4 (Edge tester)**.** A monotonicity testing algorithm is an *edge tester* if it only uses edge tests.

In other words, the edge tester should have the following structure:

Algorithm 2: Edge tester algorithm

```
1 **Input:** a DAG $G$, a function $f : G \to \{0,1\}$, an distance parameter $\epsilon$
2 **Output:** **Accept** or **Reject**
3 **Repeat** $h(G,\epsilon)$ times the edge test. **Reject** if found violation
4 **Accept** $f$
```

The algorithm should only specify the number of repetitions.

This simple algorithm is inefficient for some graphs, including the line. Imagine the anti-majority function on the line where lower-half has the value of 1 and the upper-half has the value of 0 (see Fig 1.2). The edge tester will have to examine $\Theta(n)$ edges, where n is the length of the line. It is as bad as the naive algorithm that checks the entire domain.

However, for the boolean hypercube, as shown by Goldreich *et al.* [19], the edge tester can achieve a query complexity of $O(\frac{d}{\epsilon})$, where d is the number of dimensions in the hypercube. In this scenario, the edge tester is orders of magnitude better than the naive algorithm, as the latter will require 2^d queries.

That raises the question: when is the edge tester efficient? In this thesis, we answer this question with a general framework to describe how efficient the edge tester can be for different graph products.

Theorem 1 (Informal version)**.** *For every $d \geq 1$ and every product of directed acyclic graph $G = G_1 \times \cdots \times G_d$*[1]*, there is a monotonicity tester with query complexity*

$$Q_\epsilon(G) = O\left(\frac{d}{\epsilon} \cdot \max_{i \leq d}(\tau'(G_i))\right),$$

where $\tau'(G_i)$ is a parameter depending only on G_i for some $i \in \{1, \cdots, d\}$.

The best known algorithm to date is given by Halevy and Kushilevitz [22] with query complexity $Q_\epsilon(G) = O(\frac{d \cdot 2^d}{\epsilon} \cdot \tau)$, where τ is another graph dependent parameter of some base graph G_i. In Chapter 5, we show that this can be improved to $O(\frac{d^5}{\epsilon} \cdot \tau)$ by only using the theorems in their original paper.

1.3 Isoperimetry and Monotonicity Testing

The analysis of the above mentioned edge tester is non-trivial because we are dealing with two very different quantities of the underlying structure. One is the fraction of edges that violate monotonicity condition, while the other is the fraction of vertices that need to be changed to obtain a monotone function. In the proof of Goldreich *et al.* [19], they are the first to relate this problem to the study of discrete isoperimetry.

The classic isoperimetry study concerns the relation between the surface area and the volume of an object, and tries to lower bound the surface area of the object by its volume. For example, on a two-dimensional plane, given a closed shape of volume (in this case area) V, its "surface area" (perimeter) S is at least $\sqrt{4\pi V}$. Furthermore, we know that this minimum is achieved only when the shape is a circle.

An analogous definition also exists for discrete domains. For an undirected graph $G(V, E)$, we want to partition the set of vertices V into A and $\overline{A}$. The volume of each set is its size (sometimes normalized), and the surface area is measured by the number of edges that go from one set to the other.

[1] Formal definition of a graph product is in Chapter 2.

The *edge isoperimetric number* of a graph G,

$$\mathrm{iso}(G) = \min_{0 \le |S| \le \frac{n}{2}} \frac{|\partial S|}{|S|},$$

is a fundamental parameter of interest in graphs. It describes the general relation between a subset of a graph and its expansion. Many fundamental properties of graphs can be derived from *isoperimetric inequalities* on this and other related isoperimetric numbers of graphs.

We can also consider directed isoperimetric inequalities over general graphs by introducing the *directed isoperimetric constant* of a directed acyclic graph.

Definition 5 (Isoperimetric constant)**.** The *directed isoperimetric constant* of a directed acyclic graph G is

$$\mathrm{iso}^{-}(G) = \min_{f} \frac{|I^{-}(f)|}{\mathrm{dist}_{\mathrm{mono}}(f)}$$[2]

where f is taken from all non-monotone function over G and

$$I^{-}(f) = |\partial^{-}(f)|/|V| = |\{(u,v) \in E : f(u) = 1 \wedge f(v) = 0\}|/|V|$$

$I^{-}(f)$ and $\partial^{-}(f)$ are the *negative influence* and the *negative boundary* of the function f respectively.

This definition of directed isoperimetric constant is an analogue to the classic isoperimetric constant. For the numerator, instead of considering all edges going from one set to the other, we only consider the edges that violate the monotonicity condition. As for the denominator, we replace the distance to constant functions with distance to monotone functions.

This definition of isoperimetric constant almost immediately gives a bound on the query complexity of the edge tester over that graph.

Proposition 1. *For every directed acyclic graph $G = (V, E)$,*

$$Q_{\epsilon}(G) = O\left(\frac{|E|}{|V|\mathrm{iso}^{-}(G)} \cdot \frac{1}{\epsilon}\right).$$

[2]The definition of $\mathrm{dist}_{\mathrm{mono}}$ is in Chapter 2.

Proof. We can get this bound by analyzing the query complexity of the edge tester over a graph G. The rejection probability of a single edge test on a function $f : G \to \{0,1\}$ that is at least ϵ-far from monotone is

$$\frac{|\partial^-(f)|}{|E|} = \frac{|V| \cdot I^-(f)}{|E|} \geq \frac{|V| \cdot \mathrm{dist}_{\mathrm{mono}}(f) \cdot \mathrm{iso}^-(G)}{|E|}.$$

Therefore we need to apply the edge test $O\left(\frac{|E|}{|V|\mathrm{iso}^-(G)} \cdot \frac{1}{\epsilon}\right)$ times to reject the function with high probability. □

Our main technical result is that directed isoperimetric inequalities does not get much smaller under graph product operations:

Theorem 2 (Directed isoperimetric inequality)**.** *For every $d \geq 1$ and every product of directed acyclic graphs $G = G_1 \times \cdots \times G_d$,*

$$\min_{i \leq d}(\mathrm{iso}^-(G_i)) \geq \mathrm{iso}^-(G) \geq \frac{\min_{i \leq d}(\mathrm{iso}^-(G_i))}{2}.$$

Theorem 2 can be viewed as a directed analogue of Chung and Tetali's classic isoperimetric inequality [15].

Chung–Tetali isoperimetric inequality. *For every $d \geq 1$ and every product of undirected graph $G = G_1 \times \cdots \times G_d$,*

$$\min_{i \leq d}(\mathrm{iso}(G_i)) \geq \mathrm{iso}(G) \geq \frac{\min_{i \leq d}(\mathrm{iso}(G_i))}{2}.$$

When combined with Proposition 1, Theorem 2 gives upper bounds on the query complexity for testing monotonicity of boolean functions using the edge tester over graph products, which is formally stated as the following:

Theorem 1 (Formal version)**.** *For every $d \geq 1$ and every product of directed acyclic graphs $G = G_1 \times \cdots \times G_d$,*

$$Q_\epsilon(G) = O\left(\frac{1}{\epsilon \cdot \min_{i \leq d}(\mathrm{iso}^-(G_i))} \cdot \sum_{i=1}^{d} \frac{|E_i|}{|V_i|}\right).$$

1.4 Organization of this Thesis

In Chapter 2, we will introduce notation and definitions we will be using. Along with a related work section providing some perspectives.

In Chapter 3, we will show the proof of the above mentioned directed isoperimetric inequality, *i.e.* Theorem 2.

In Chapter 4, we show how we can use the directed isoperimetric inequality to obtain the monotonicity tester result, *i.e.* the proof of Theorem 1. We also explore some other interesting graphs that exhibit nice isoperimetric constant, and therefore can be tested effective for monotonicity under our framework.

In Chapter 5, we show how we can improve the query complexity of the algorithm given by Halevy and Kushilevitz [22] by applying their theorem in a slightly different way.

Chapter 2

Preliminaries

In this chapter, we will introduce the notation and the definitions we use throughout the thesis in Section 2.1, and also review related work in Section 2.2.

2.1 Definitions and Notation

The most important measure we need when it comes to property testing is the notion of distance. Here we are mainly concerned about discrete domain with discrete range, and therefore we will use the following definition for measuring distance between functions.

Definition 6 (Distance between functions)**.** The *distance* between two function $f : G \to \{0, 1\}$ and $g : G \to \{0, 1\}$ over $G = (V, E)$ is

$$\text{dist}(f, g) = |\{x : f(x) \neq g(x)\}|/|V| \,.$$

Now we can also measure a distance between a function to a property using the following definition:

Definition 7 (Distance to monotonicity)**.** The *distance to monotonicity* of $f : G \to \{0, 1\}$ is

$$\text{dist}_{\text{mono}}(f) = \min_{g} \text{dist}(f, g) \,.$$

where g is taken from all monotone functions.

In this thesis, we are interested in the query complexity of testing monotonicity of function over graph products, and a graph product is defined as follows:

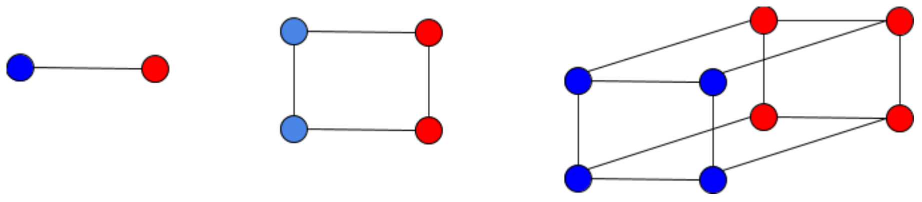

Figure 2.1: Example graph power of G, G^2, G^3. We can view this process as replacing every node in the current graph with a copy of the base graph.

Definition 8 (Product graphs)**.** Given directed acyclic graphs G and H, the product of G and H, $G \times H$ is given by $V(G \times H) = \{(u, v) : u \in G, v \in H\}$, and $E(G \times H) = \{((u, v), (u', v')) : (u = u' \wedge (v, v') \in E(H)) \vee (v = v' \wedge (u, u') \in E(G))\}$

We can also apply this definition repeatedly to get higher order graph products.

In a graph product, we will often use the notion of *dimension*, where a "dimension" is formed by considering all the copies of G_i in the product graph. Restricting the product graph down to a single dimension is a commonly used technique throughout this thesis.

The following proposition is a basic property of graph products that describes the number of edges and vertices in the product graphs in terms of that of the base graphs. In the proof of this proposition, we will see how this concept of dimension can be used.

Proposition 2. *Let $G = G_1 \times \cdots \times G_d$. Then $|V(G)| = \prod_{i=1}^{d} |V(G_i)|$ and*

$$|E(G)| = |V(G)| \cdot \sum_{i=1}^{d} \frac{|E(G_i)|}{|V(G_i)|}$$

so that $\frac{|E(G)|}{|V(G)|} = \sum_{i=1}^{d} \frac{|E(G_i)|}{|V(G_i)|}$.

Proof. The number of edges along the ith dimension is $\frac{|V(G)|}{|V(G_i)|} \cdot |E(G_i)|$, as there are $\frac{|V(G)|}{|V(G_i)|}$ copies of the base graph G_i. And therefore

$$|E(G)| = \sum_{i=1}^{d} \frac{|V(G)| \cdot |E(G_i)|}{|V(G_i)|} = |V(G)| \cdot \sum_{i=1}^{d} \frac{|E(G_i)|}{|V(G_i)|}. \qquad \square$$

In the graph products setting, it is often useful to consider the distance to monotonicity for a single dimension of the graph product. To formally define this, we also need the notation for functional restriction.

Definition 9 (Functional restriction)**.** The function $f|^i_{\alpha,\beta} : G_i \to \{0,1\}$ is a *functional restriction* of $f : G_1 \times \cdots \times G_d \to \{0,1\}$ onto G_i given by $f|^i_{\alpha,\beta}(x) = f(\alpha x \beta)$, where $\alpha \in G_1 \times \cdots \times G_{i-1}, \beta \in G_{i+1} \times \cdots \times G_d$.

Hence the function $f|^i_{\alpha,\beta}$ is f over a slice of G_i specified by α and β.

The distance to monotonicity in dimension i is then defined as:

Definition 10.

$$\mathrm{dist}^i_{\mathrm{mono}}(f) = \frac{1}{\Pi_{j \neq i}|V(G_j)|} \sum_{\alpha,\beta} \mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta}),$$ [1]

It is also useful to talk about the negative influence in one direction. That is, the average negative influence of all copies of the base graph in that dimension. Or equivalently, the average number of non-monotone edges per node in that dimension.

Definition 11 (Dimensional negative influence)**.** The *dimensional negative influence* of a function $f : G = G_1 \times \cdots \times G_d \to \{0,1\}$ is

$$I_i^-(f) = \frac{1}{\Pi_{j \neq i}|V(G_j)|} \sum_{\alpha,\beta} I^-(f|^i_{\alpha,\beta}).$$

Similar to the definition of total influence, the sum of dimensional negative influences equals to the total negative influence.

Lemma 1. *For any function $f : G = G_1 \times \cdots \times G_d \to \{0,1\}$,*

$$I^-(f) = \sum_{i=1}^{d} I_i^-(f)$$

Proof. We can establish a bijection between the violation pairs in the every functional restriction and the original function. For every violation pair in the original function, it must be in the form of $(\alpha x \beta, \alpha y \beta)$ where x, y has an edge between them in G_i. Then this pair can be mapped to the violation pair (x, y) in $f|^i_{\alpha,\beta}$, and vice versa. Therefore we have $\partial^-(f) = \sum_{i=1}^d \sum_{\alpha,\beta} \partial^-(f|^i_{\alpha,\beta})$. After re-normalizing, we have

[1] We use $\sum_{\alpha,\beta}$ as a short-hand for $\sum_{\alpha \in G_1 \times \cdots \times G_{i-1}, \beta \in G_{i+1} \times \cdots \times G_d}$, where i should always be clearly indicated in an outter scope.

$$
\begin{aligned}
\sum_{i=1}^{d} I_i^-(f) &= \sum_{i=1}^{d} \frac{1}{\Pi_{j\neq i}|V(G_j)|} \sum_{\alpha,\beta} I^-(f|^i_{\alpha,\beta}) \\
&= \sum_{i=1}^{d} \frac{1}{\Pi_{j\neq i}|V(G_j)|} \sum_{\alpha,\beta} \frac{\partial^-(f|^i_{\alpha,\beta})}{|V(G_i)|} \\
&= \frac{1}{|V(G)|} \sum_{i=1}^{d} \sum_{\alpha,\beta} \partial^-(f|^i_{\alpha,\beta}) \\
&= \frac{1}{|V(G)|} \cdot \partial^-(f) \\
&= I^-(f). \qquad \square
\end{aligned}
$$

Another fact that we will use in later sections is that $\mathrm{iso}^-(G_i)$ still serves as a lower bound on the ratio between dimensional negative influence I_i^- and dimensional distance to monotonicity $\mathrm{dist}^i_{\mathrm{mono}}$.

Lemma 2.

$$\mathrm{iso}^-(G_i) \leq \frac{I_i^-(f)}{\mathrm{dist}^i_{\mathrm{mono}}(f)}.$$

Proof. This is true because if we apply the definition of I_i^- and $\mathrm{dist}^i_{\mathrm{mono}}$ with the normalization factors canceled, we have

$$\frac{I_i^-(f)}{\mathrm{dist}^i_{\mathrm{mono}}(f)} = \frac{\sum_{\alpha,\beta} I^-(f|^i_{\alpha,\beta})}{\sum_{\alpha,\beta} \mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta})}.$$

Note that for any (α,β), $\mathrm{iso}^-(G_i) \leq \frac{I^-(f|^i_{\alpha,\beta})}{\mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta})}$, so $\mathrm{iso}^-(G_i) \cdot \mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta}) \leq I^-(f|^i_{\alpha,\beta})$. Therefore,

$$\mathrm{iso}^-(G_i) \cdot \sum_{\alpha,\beta} \mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta}) \leq \sum_{\alpha,\beta} I^-(f|^i_{\alpha,\beta}),$$

and so

$$\mathrm{iso}^-(G_i) \leq \frac{I_i^-(f)}{\mathrm{dist}^i_{\mathrm{mono}}(f)}. \qquad \square$$

2.2 Related Work

Monotonicity testing has been extensively studied over the past two decades with the main focus on the hypercube and the hypergrid domain with different ranges. The majority of testing algorithms can be categorized into the edge tester or the pair tester, with one notable exception that we will see in later sections. The pair tester is a natural extension of the edge tester, where instead of just picking adjacent vertices, a pair test consists of queries to any two comparable vertices.

Definition 12 (Pair tester)**.** A monotonicity testing algorithm is a *pair tester* if it only uses the pair test. A *pair test* is the simple test that queries two comparable vertices drawn from some distribution and rejects the function if these two vertices form a violation of monotonicity.

In the following sections, we will review some of the previous result of monotonicity testing over the hypercube, the hypergrid, general domains, and graph products.

2.2.1 Boolean Hypercube

Goldreich *et al.* [21] present the first analysis of edge tester for the boolean hypercube with a query complexity of $O(d^2/\epsilon)$. In the journal version of this paper [19], they improved the upper bound to $O(d/\epsilon)$, which matches the lower bound for edge tester shown in the same paper. In their proof of the query complexity of the monotonicity tester, they implicitly prove a directed version of the well-known inequality $I(f) \geq \Omega(\text{var}(f))$ for boolean hypercube using the *shifting argument*, which inspire the work in this thesis.

Goldreich *et al.* [19] was the best known monotoncity tester for the boolean hypercube for over 15 years until Chakrabarty and Seshadhri [11] improved the result of Goldreich *et al.* [19] to a query complexity of $\tilde{O}(d^{7/8}\epsilon^{-3/2})$ by using a *pair tester* that samples pairs that are at most $O(\sqrt{d})$ away from each other instead of an edge tester. This is the first sublinear tester in terms of the number of dimensions d. They are also the first to explicitly identify the link between directed isoperimetric inequality. In the proof, they show a directed version of Margulis' isoperimetric inequality, which is stronger than the inequality used by Goldreich *et al.* [19], and immediately get a better upper bound on the corresponding tester.

Khot, Minzer, and Safra [25] extend this idea further and later establish a directed analogue of Talagrand's isoperimetric inequality on the hypercube to obtain the optimal monotonicity tester, improved the upper bound to $\tilde{O}(\sqrt{d}/\epsilon^2)$.

On the lower bound side, Fischer *et al.* [18] develop the first lower bound result of $\Omega(\log d)$ for non-adaptive two-sided algorithm and $\Omega(\sqrt{d})$ for non-adaptive one-sided algorithm. In the same paper where Chakrabarty and Seshadhri [11] show the first sublinear tester, they also improve the lower bound for non-adaptive two-sided algorithm to $\tilde{\Omega}(d^{1/5}))$. This is later improved to $\Omega(d^{1/2-o(1)})$ by Chen *et al.* [12]. Therefore the result of Khot, Minzer, and Safra [25] is indeed near optimal.

For adaptive algorithms, every lower bound for non-adaptive lower bound implies a corresponding lower bound for adaptive algorithm with a loss of a log-factor. Therefore, the result of Fischer *et al.* [18] implies a lower bound of $\Omega(\log\log d)$ for the more general adaptive algorithms. And the result of Chen, Servedio, and Tan [13] implies a lower bound of $\Omega(\log d)$. The first polynomial bound of $\tilde{\Omega}(d^{1/4}))$ is given by Belovs and Blais [3], and later improved to $\Omega(d^{1/3})$ by Chen, Waingarten, and Xie [14].

2.2.2 Boolean Hypergrid

The other domain that is most well-studied is the hypergrid, where the base graph is a total order of length n. Therefore the hypercube is a special instance where $n = 2$.

Testing over the line (which is another special case of hypergrid with $d = 1$) was first brought to attention by Ergün *et al.* [17], where they proposed a $O(\log n)$ pair tester for the line. Note that their method is originally proposed for testing if an array is sorted, and therefore works on general range as well.

In the work of Goldreich *et al.* [20], they propose a framework that can be used to construct an $O(1/\epsilon)$ tester for boolean line (*i.e.* testing threshold function) that is neither an edge tester nor a pair tester. The tester works by trying to learn where the threshold for the function should be, and rejecting the function if the subsequent samples conflict with that belief.

Dodis *et al.* [16] first studies the higher dimensional hypergrid. Their construction gives an $O(\frac{d}{\epsilon}\log(n)\log(|\mathrm{R}|))$ tester where R is the range of the function. When the range is boolean, the above tester gives a bound of $O(\frac{d}{\epsilon}\log(n))$. In their proof, they build a shifting operator that can reduce the analysis of the hypergrid to the line. And then they construct a 2-spanner graph to work with the shifting argument. We will be discussing more about this in Section 4.2.

In the latest paper by Black, Chakrabarty, and Seshadhri [6, 7], they use a Margulis-style isoperimetric inequality to achieve a query complexity of $\tilde{O}(d^{5/6}/\epsilon)$, which is the first sub-linear tester in terms of the number of dimensions in the domain for the hypergrid.

It is worth noting that there has been no lower bound specifically targeting the hypergrid. It is mostly believed that testing on hypergrid should not be harder than testing over hypercube because of various domain reduction techniques.

2.2.3 Testing over General Domain

Ficher *et al.* [18] give the first non-trivial result for testing functions over general graphs using the pair tester. The tester requires $O(\sqrt{|V|/\epsilon})$ queries. They convert the original graph into a complete bipartite graph and draw a set of points and then check all comparable points for non-monotonicity. The function is rejected if there is any violating pair. The correctness follows from the matching lemma, which states that if the function is ϵ-far from monotone, then there exists a matching of violated edges of size at least $\epsilon|V|/2$.

For the special class of graphs with a sparse 2-spanner, Dodis *et al.* and Bhattacharyya *et al.* [16, 5] show that there is an $O(\frac{|E(2\text{-TC}(G))|}{\epsilon|V(G)|})$-tester for such graphs. In particular, Bhattacharyya *et al.* [5] give a method for constructing a sparse 2-spanner for any H-minor-free graph, which gives query complexity of $O(\epsilon \log^2 |V(G)|)$ for these graphs. We will give a in-depth look into these graphs in Section 4.3 and Section 4.4.

In the same paper of Ficher *et al.* [18], the authors also give a lower bound of $|V|^{\Omega(\frac{1}{\log\log|V|})}$ for non-adapt testers. This is the only lower bound result for testing over general domain.

2.2.4 Testing over Graph Products

As far as we know, the problem of testing over general graph products is studied only by Halevy and Kushilevitz [22], where they give query complexity of $O(d2^d) \cdot \max_i(Q_\epsilon(G_i))$.

However, they implicitly show a query complexity of $O(d^3/\epsilon) \cdot Q_{\epsilon/2d^2}(G)$ for boolean functions over general graph products. This result does not appear in their original paper probably due to the fact that boolean range is not their main concern. For completeness, we include the proof of this bound in Chapter 5

It is also worth mentioning that there is no lower bound specifically for product graphs, only the general ones that apply to product graphs.

Chapter 3

Proof of the Directed Isoperimetric Inequality

In this chapter, we show that the technique of Goldreich *et al.* [19] can be extended to general graph products. In particular, we construct a shifting argument that works on any graph products beyond the hypercube. This immediately gives better query complexity for many types of graph products as shown in Chapter 4.

The main goal of this chapter is to prove the directed isoperimetric inequality, which is restated below.

Theorem 2 (Restated)**.** *For any directed acyclic graph $G_1 \times \cdots \times G_d$,*

$$\min_{i \le d}(\text{iso}^-(G_i)) \ge \text{iso}^-(G_1 \times \cdots \times G_d) \ge \frac{\min_{i \le d}(\text{iso}^-(G_i))}{2}.$$

3.1 Proof Overview

In the proof of Goldreich *et al.* [19], they show that there is a shifting operator that can transform every function $f : \{0,1\}^d \to \{0,1\}$ into a monotone function by modifying up to $2 \cdot \text{dist}_{\text{mono}}(f)$ fraction of the function. The number of points being changed in this shifting operator is dependent only on $I^-(f)$, therefore it is possible to lower bound I^- in terms of $\text{dist}_{\text{mono}}$.

We take this technique one step further, and construct shifting operator that works in the same fashion but over any general graph product instead of just the boolean hypercube.

In our proof, we show that this new shifting operator has the same properties that we can use to lower bound I_i^- in terms of $\mathrm{dist}_{\mathrm{mono}}$ plus a graph dependent constant iso^-. In fact, we can also show that Goldreich *et al.* [19] is a special case of this shifting operator with iso^- being 1 for the boolean hypercube.

3.2 Shifting Operator

The core of our proof is to construct the proper shifting operator with desired properties. In this section, we describe the shifting operator as an algorithm and summarized the properties it possesses in Lemma 3.

Algorithm 3: Shifting operator

```
Input: a DAG $G_1 \times \cdots \times G_d$, a function $f : G \to \{0,1\}$, an integer $i$
Output: a function $f : G \to \{0,1\}$
Let $L$ be a linear extension of $G_i$
Let $h = f$
foreach $p \in L$ from lowest to highest do
    foreach $\alpha \in G_1 \times \cdots \times G_{i-1}, \beta \in G_{i+1} \times \cdots \times G_d$ do
        if there exists $q >_G p$ such that $f|^i_{\alpha,\beta}(p) > f|^i_{\alpha,\beta}(q)$ then
            Select the highest such $q$ (in $L$)
            swap the values of $h(\alpha p \beta)$ and $h(\alpha q \beta)$
return $h$
```

We claim that this operator satisfies three conditions:

Lemma 3. *For every graph $G = G_1 \times \cdots \times G_d$, every function $f : G \to \{0,1\}$ and every $i \in [d]$, the output h of the shifting operator (Algorithm 3) satisfies the following three properties.*

P1 *h is monotone in dimension i, i.e. $I_i^-(h) = 0$;*

P2 *for every $j \in [d]$, $I_j^-(h) \le I_j^-(f)$;*

P3 $\mathrm{dist}(f,h) \le 2 \cdot I_i^-(f) \cdot \frac{1}{\mathrm{iso}^-(G_i)}$.

We will be using these three properties in our proof of Theorem 2.

3.3 Proof of Lemma 3

In this section, we will prove the above mentioned three properties of the shifting operator (Algorithm 3).

Property P1

Proof. We will show by strong induction that every time after $p \in L$ being checked and potentially swapped, there will be no violating like (r, p), where $r <_G p$.

Base case: For nodes with in-degree of 0, this is trivially true.

Inductive step: For the case where p is swapped with some other node, assume towards the contradiction that a violating pair (r, p), where $r <_G p$, is created as result of the swap. By definition, we have $f|^i_{\alpha,\beta}(r) = 1$, $f|^i_{\alpha,\beta}(p) = 1$, $f|^i_{\alpha,\beta}(q) = 0$, $h|^i_{\alpha,\beta}(p) = 0$, and $h|^i_{\alpha,\beta}(q) = 1$. Then $\alpha r \beta$ would have swapped with $\alpha q \beta$, since they are comparable in G through p, which is a contradiction to the fact that the lower node will always be swapped with the highest node available.

For the case where p is not swapped with any higher node, again assume towards the contradiction that there is a violating pair (r, p), where $r <_G p$. By definition, we have $f|^i_{\alpha,\beta}(r) = 1$, $f|^i_{\alpha,\beta}(p) = 0$. Since p is not swapped with any higher node, then we also have for every $q >_G p$, $f|^i_{\alpha,\beta}(q) = 0$. Then r would have swapped with q when r is visited by the algorithm since $r <_G p <_G q$. Then $f|^i_{\alpha,\beta}(r)$ should be 0, which contradicts our assumption.

Thus at the end of this shifting operator, there will be no violation in dimension i. □

Property P2

Proof. We will prove P2 by showing that during the shifting, for any two slices of G_i, the number of non-monotone edge between them does not increase. We will explain in the next paragraph why this construction is enough to prove P2.

Let g and g' be the function h before and after each iteration of Line 5, *i.e.* every time a pair of values being swapped. Formally, for any $\alpha \in G_1 \times \cdots \times G_{i-1}, \beta \in G_{i+1} \times \cdots \times G_d, \gamma \in G_1 \times \cdots \times G_{i-1}, \delta \in G_{i+1} \times \cdots \times G_d$, such that $\alpha \leq \gamma$ and $\beta \leq \delta$ with $\alpha \neq \gamma$ or $\beta \neq \delta$, let $S = \{p \in G_i : g|^i_{\alpha,\beta}(p) > g|^i_{\gamma,\delta}(p)\}$ and $S' = \{p \in G_i : g'|^i_{\alpha,\beta}(p) > g'|^i_{\gamma,\delta}(p)\}$. We want to show that $|S| \geq |S'|$. Here each pair of (α, β) and (γ, δ) will specify two slices of G_i in ith dimension. By considering the edges between these two copies of G_i for all possible pairs of (α, β) and (γ, δ), it allows us to draw conclusion about I_j for any $j \neq i$.

If $(g|^i_{\alpha,\beta}(p), g|^i_{\gamma,\delta}(p))$ is $(0, 0)$, then no values will be swapped by the shifting operator therefore $S = S'$.

If $(g|^i_{\alpha,\beta}(p), g|^i_{\gamma,\delta}(p))$ is $(0, 1)$, let q be the point that gets swapped with $\gamma p\delta$ if it exists (otherwise we have $S = S'$). Then after the swap, we have $(g'|^i_{\alpha,\beta}(p), g'|^i_{\gamma,\delta}(p)) = (0, 0)$ and $(g'|^i_{\alpha,\beta}(q), g'|^i_{\gamma,\delta}(q)) = (0, 1)$ or $(1, 1)$. There is no increase in the number of violating pairs.

If $(g|^i_{\alpha,\beta}(p), g|^i_{\gamma,\delta}(p))$ is $(1, 0)$, let q be the point that gets swapped with $\alpha p\beta$ if it exists (otherwise we have $S = S'$ again). Then after the swap, we have $(g'|^i_{\alpha,\beta}(p), g'|^i_{\gamma,\delta}(p)) = (0, 0)$ and $(g'|^i_{\alpha,\beta}(q), g'|^i_{\gamma,\delta}(q)) = (1, 0)$ or $(1, 1)$. Therefore we always remove the violating pair at $(\alpha p\beta, \gamma p\delta)$ and add at most one violating pair at $(\alpha q\beta, \gamma q\delta)$. Hence there is no increase in the number of violating pairs.

If $(g|^i_{\alpha,\beta}(p), g|^i_{\gamma,\delta}(p))$ is $(1, 1)$, let q be the point that gets swapped with $\alpha p\beta$ if it exists, and let q' be the point that gets swapped with $\gamma p\delta$ if it exists. We can divide this into three cases:

1. If q does not exist, it means that all comparable points above p in $g|^i_{\alpha,\beta}$ are all 1's. Therefore $(g'|^i_{\alpha,\beta}(p), g'|^i_{\gamma,\delta}(p)) = (1, 0)$ and $(g'|^i_{\alpha,\beta}(q'), g'|^i_{\gamma,\delta}(q')) = (1, 1)$. We add one violating pair at $(\alpha p\beta, \gamma p\delta)$ and remove one violating pair at $(\alpha q'\beta, \gamma q'\delta)$. Therefore there is no increase.

2. If q' does not exists it means that all comparable points above p in $g|^i_{\gamma,\delta}$ are all 1's. Therefore $(g'|^i_{\alpha,\beta}(p), g'|^i_{\gamma,\delta}(p)) = (0, 1)$ and $(g'|^i_{\alpha,\beta}(q'), g'|^i_{\gamma,\delta}(q')) = (1, 1)$. There is no increase.

3. If both q, q' exist and $(g'|^i_{\alpha,\beta}(q), g'|^i_{\gamma,\delta}(q)) = (1, 0)$, then it implies that $q <_L q'$ and therefore $g|^i_{\alpha,\beta}(q') = 1$, and hence $(g|^i_{\alpha,\beta}(q'), g|^i_{\gamma,\delta}(q')) = (1, 0)$ and $(g'|^i_{\alpha,\beta}(q'), g'|^i_{\gamma,\delta}(q')) = (1, 1)$. We add one violating pair at $(\alpha q\beta, \gamma q\delta)$ and remove one violating pair at $(\alpha q'\beta, \gamma q'\delta)$. Therefore there is no increase. Note that it is impossible to create a violation at $g'|^i_{\gamma,\delta}(q')$, since it is always 1. □

Property P3

Proof. For any violation identified in $f|^i_{\alpha,\beta}$, at least one of the end points need to be changed to obtain a monotone function. Notice that the shifting operator never swaps value on a node twice because after a swap, the lower node p is never touched again and the higher node q will have every node above being 1 which cannot form any violating pair.

Combining these two facts, we have $\mathrm{dist}(f,h)$ is at most twice as large as $\mathrm{dist}^i_{\mathrm{mono}}(f)$. Therefore we have

$$\mathrm{dist}(f,h) \le 2 \cdot \frac{V(G_i) \cdot \sum_{\alpha,\beta} \mathrm{dist}_{\mathrm{mono}}(f|^i_{\alpha,\beta})}{V(G)} = 2 \cdot \mathrm{dist}^i_{\mathrm{mono}}(f)$$

From Lemma 2, we have

$$\mathrm{iso}^-(G_i) \le \frac{I_i^-(f)}{\mathrm{dist}^i_{\mathrm{mono}}(f)}.$$

Combining these two inequalities, we finally arrive at

$$\mathrm{dist}(f,h) \cdot \mathrm{iso}^-(G_i) \le 2 \cdot \mathrm{dist}^i_{\mathrm{mono}}(f) \cdot \mathrm{iso}^-(G_i) \le 2 \cdot I_i^-(f). \qquad \square$$

3.4 Proof of Theorem 2

Now equipped with Lemma 3, we are ready to prove our main theorem. We restate the theorem again for reference:

Theorem 2 (Restated)**.** *For any directed acyclic graph $G_1 \times \cdots \times G_d$,*

$$\min_{i \le d}(\mathrm{iso}^-(G_i)) \ge \mathrm{iso}^-(G_1 \times \cdots \times G_d) \ge \frac{\min_{i\le d}(\mathrm{iso}^-(G_i))}{2}.$$

Proof. For any product of directed acyclic graphs $G = G_1 \times \cdots \times G_d$ and any function $f : G \to \{0,1\}$ that satisfies $\mathrm{dist}_{\mathrm{mono}}(f) = \epsilon$, let $f_1, \cdots, f_{d+1}$ be the sequence of functions we get from applying above shifting operation, where f_1 is the original function and f_{d+1} is the final resulting function. From P1 and P2, we know that the f_{d+1} is a monotone function over G. The distance between the initial function and the resulting function is

$$\begin{aligned}
\mathrm{dist}(f_1, f_{d+1}) &\le \sum_{i=1}^{d} \mathrm{dist}(f_i, f_{i+1}) \\
&\le \sum_{i=1}^{d} 2 \cdot I_i^-(f_i) \cdot \frac{1}{\mathrm{iso}^-(G_i)} && \text{(from P3)} \\
&\le \sum_{i=1}^{d} 2 \cdot I_i^-(f) \cdot \frac{1}{\mathrm{iso}^-(G_i)} && \text{(from P2)} \\
&\le \frac{2}{\min_{i\le d}(\mathrm{iso}^-(G_i))} \sum_{i=1}^{d} I_i^-(f) && \text{(taking the largest factor)} \\
&= \frac{2}{\min_{i\le d}(\mathrm{iso}^-(G_i))} \cdot I^-(f)\,. && \text{(from Lemma 1)}
\end{aligned}$$

Therefore we have

$$\frac{I^-(f)}{\epsilon} \ge \frac{\min_{i\le d}(\mathrm{iso}^-(G_i))}{2}$$

for any function, which by the definition of isoperimetric constant implies

$$\mathrm{iso}^-(G_1 \times \cdots \times G_d) \ge \frac{\min_{i\le d}(\mathrm{iso}^-(G_i))}{2}.$$

For the left half of the inequality, we can show that by constructing a function over $G = G_1 \times \cdots \times G_d$ that achieves the same I^- to $\mathrm{dist}_{\mathrm{mono}}$ ratio.

Without loss of generality, we can assume the minimum is taken from G_1, then let f be the function that achieves this minimum I^- to $\mathrm{dist}_{\mathrm{mono}}$ ratio. We can construct $f' : G = G_1 \times \cdots \times G_d \to \{0, 1\}$ in the following way: $f'(x_1, \cdots, x_d) = f(x_1)$, where $x_i \in G_i$. *i.e.*, the function whose every functional restriction onto G_1 is always f. Then we have

$$I^-(f') = \frac{|\partial^-(f)| \cdot |V(G_2 \times \cdots \times G_d)|}{|V(G)|} = \frac{|\partial^-(f)|}{|V(G_1)|} = I^-(f).$$

As for distance to monotonicity, all functional restriction of f' onto G_1 will have to be transformed into monotone function in order for the transformed f' to be monotone, therefore,

$$\mathrm{dist}_{\mathrm{mono}}(f') = \frac{|\mathrm{dist}_{\mathrm{mono}}(f)| \cdot |V(G_1)| \cdot |V(G_2 \times \cdots \times G_d)|}{|V(G)|} = \mathrm{dist}_{\mathrm{mono}}(f).$$

Combining those two equations, we have

$$\frac{I^-(f')}{\mathrm{dist}_{\mathrm{mono}}(f')} = \frac{I^-(f)}{\mathrm{dist}_{\mathrm{mono}}(f)}.$$

Thus we have,

$$\min_{i \le d}(\mathrm{iso}^-(G_i)) \ge \mathrm{iso}^-(G_1 \times \cdots \times G_d). \qquad \square$$

This intuition behind the left half of inequality is that by granting the ability to manipulate the function on a finer granularity, we should be able to achieve lower isoperimetric constant. However, the other half of inequality tells that one can only hope to reduce isoperimetric constant by a half. It remains open whether the bound is tight.

Chapter 4

Monotonicity Testing Result

In this chapter we will show the monotonicity testing results we can get from applying the directed isoperimetric inequality on many types of graph products. We do this by studying the isoperimetric constant of different base graphs, and then applying the directed isoperimetric inequality we get from Chapter 3 to extend the result to their corresponding power/products.

In particular, we will show the improved bound of testing general graph products in Section 4.1; a simplified version of a $O(d \log n)$ bound for the hypergrid in Section 4.2; the bound of products of transitive closures and spanner graphs in Section 4.3; the bound of product of $\mathcal{H}$-minor free graphs in Section 4.4; and lastly the bound on constant comparable graphs in Section 4.5.

4.1 General Graph Products

The improve bound for testing monotonicity over general graph products follows almost immediately from the directed isoperimetric inequality.

Theorem 1 (Restated)**.** *For every directed acyclic graph product $G = G_1 \times \cdots \times G_d$,*

$$Q_\epsilon(G) = O\left(\frac{1}{\epsilon \cdot \min_{i \le d}(\text{iso}^-(G_i))} \cdot \sum_{i=1}^{d} \frac{|E_i|}{|V_i|}\right).$$

Proof. For any function $f : G = G_1 \times \cdots \times G_d \to \{0, 1\}$, the rejection probability of the edge tester is

$$\begin{aligned}
\frac{I^-(f) \cdot |V(G)|}{|E(G)|} &= \frac{I^-(f)}{\sum_{i=1}^d \frac{|E_i|}{|V_i|}} && \text{(from Proposition 2)} \\
&\geq \frac{\text{iso}^-(G) \cdot \text{dist}_{\text{mono}}(f)}{\sum_{i=1}^d \frac{|E_i|}{|V_i|}} && \text{(from Definition 5)} \\
&\geq \frac{\min_{i \leq d}(\text{iso}^-(G_i)) \cdot \text{dist}_{\text{mono}}(f)}{2 \cdot \sum_{i=1}^d \frac{|E_i|}{|V_i|}} && \text{(from Theorem 2)}
\end{aligned}$$

Therefore, from Proposition 1, by repeating the edge test $O\left(\frac{1}{\epsilon \cdot \min_{i \leq d}(\text{iso}^-(G_i))} \cdot \sum_{i=1}^d \frac{|E_i|}{|V_i|}\right)$ times, the function will be rejected with high probability. □

This is the most general bound we can get from directly applying the inequality we have from Chapter 3. In the following chapters, we will show results on more specific graphs by studying their isoperimetric constants.

4.2 Hypergrid

In this section, we use our inequality and an augmented hypergrid to show that testing monotonicity of boolean functions over hypergrid can be done with query complexity $O(\frac{d}{\epsilon} \log n)$, matching the bound given by Dodis *et al.* [16]. We will use the same augmented line construction used by Dodis *et al.* [16], but we also demonstrate how this can fit into our framework of isoperimetric constant.

Theorem 3. *There is a monotonicity testing algorithm for the hypergrid $[n]^d$ with query complexity $O(d \log n/\epsilon)$*

Proof. To prove the theorem, we only need to construct a augmented line L which satisfies that

1. $E(L)/V(L) = \log(n)$,
2. and $\text{iso}^-(L) = 1$.

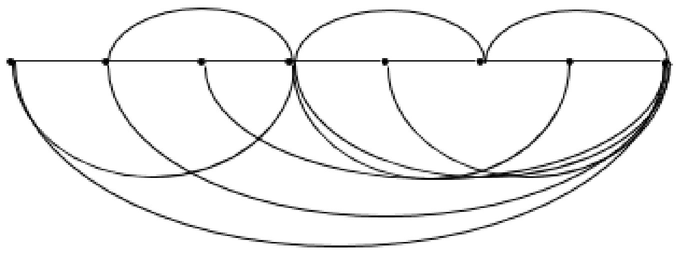

Figure 4.1: An example of the above algorithm executed on a line of length 8. The arcs above are from the first round of iteration, and the arcs underneath are from the second round. This construction was first given by Dodis *et al.* [16] in their proof of Proposition 9.

Once we have these conditions, the theorem itself follows from Theorem 1.

Without loss of generality, we assume n is a power of 2. We construct the following augmented line.

Algorithm 4: Augmented Line

```
1 Let $L$ be a linear order
2 foreach $i$ in $1, 2, 4, \cdots, \log(n) - 1$ do
3     foreach $j$ in $1, 2, 3, \cdots \lceil n/2^i \rceil$ do
4         color the node $i \cdot j$ with color $i$
5     Connect every node to two closest colored nodes with color $i$
6 return $L$
```

We call these colored nodes hubs. We use different colors mainly to distinguish between different rounds of execution, a node can have multiple color. At the end of the algorithm, we have $E(L) = n\log(n)$ and therefore the first condition is satisfied.

For the second condition, we argue that between any two nodes, there must be a node they both connect to, if they are not already connected. As for any pair that are k away from each other, let m be the smallest power of 2 that are greater than k. These two nodes must be contain by two other nodes of color $\log(m)$ that are m away. They must both be connected the middle point this range.

Therefore if any pair forms a violation, there is also a violation between either of the pair and the node they both connect to. This will mean the ratio between the I^- and $\text{dist}_{\text{mono}}$ is 1 for any function, *i.e.*, $iso^-(L) = 1$.

By apply Theorem 1, we have the query complexity of edge tester over boolean hypergrid is $O(\frac{d}{\epsilon}\log n)$ $\square$

4.3 Transitive-Closures Spanners

In this section, we will explore a special type of supergraph called spanner graphs. We show that both the transitive closure and 2-spanner graphs of any base graph possess an isoperimetric constant of $\Theta(1)$.

The study of such graphs is also pioneered by Dodis *et al.* [16] in an application to the augmented line in their work. And later formally proven by Bhattacharyya *et al.* [5] where it is also used in showing the query complexity of $\mathcal{H}$-minor free graph.

Transitive closure of a graph is constructed by adding edges between all comparable nodes, *i.e.*,

Definition 13 (Transitive Closure)**.** the *transitive closure* $\mathrm{TC}(G)$ of a directed acyclic graph G is a graph that satisfies:

1. $V(\mathrm{TC}(G)) = V(G)$;
2. $E(\mathrm{TC}(G)) = \{(u, v) : \text{there exists a path from } u \text{ to } v \text{ in } G\}$.

And then the definition of k-TC-spanner is obtained by relaxing the above condition by allowing two vertices to be not directly connected.

Definition 14 (k-TC-spanner)**.** A graph G' is a *k-TC-spanner* $\text{k-TC}(G)$ of a directed acyclic graph G if G' satisfies that:

1. $V(\text{k-TC}(G)) = V(G)$;
2. $E(\text{k-TC}(G))$ is a subset of $E(\mathrm{TC}(G))$ and for all comparable vertices u, v there exists a path from u to v of length less than or equal to k in $\text{k-TC}(G)$.

Transitive closures and 2-TC-spanners behave very nicely under our isoperimetric constant framework.

Lemma 4. *For $k = 1$ or $k = 2$ and every directed acyclic graph G,* $\mathrm{iso}^{-}(\text{k-TC}(G)) = \Omega(1)$

We will use a lemma by Dodis *et al.* [16], which is stated as following:

Lemma (Lemma 7 of Dodis *et al.* [16]). *For every directed acyclic graph G and every function $f : G \to \{0, 1\}$, the violation graph of f on G has a matching of size* $\mathrm{dist}_{\mathrm{mono}}(f) \cdot |V(G)|/2$*. The violation graph is formed all pairs (u, v) such that $u < v$ and $f(u) > f(v)$.*

Then our lemma follows directly from this matching lemma.

Proof of Lemma 4. For $k = 1$, consider any maximum matching in the violation graph. Since $E(\mathrm{TC}(G))$ contains all comparable pairs, then the edges in the maximum matching must be a subset of $E(\mathrm{TC}(G))$. Therefore $I^{-}(f)/\mathrm{dist}_{\mathrm{mono}}(f) \geq 1/2$ for all f.

For $k = 2$, consider any maximum matching in the violation graph. For any pair (u, v) in the matching, if (u, v) is an edge in the 2-TC(G), then we have an violated edge. Otherwise, we know that there is a w such that (u, w) and (w, v) are both edges of 2-TC(G). And exactly one of them is a violation edge. Since this is a matching, w can serve as an endpoint for at most another edge. Therefore for every four vertices in the matching, there has to be at least one violation edge in 2-TC(G). Thus $I^{-}(f)/\mathrm{dist}_{\mathrm{mono}}(f) \geq 1/4$ for all f.[1] □

Therefore we can test monotonicity over products of transitive closures and 2-TC-spanners effectively.

Theorem 4. *Let $G = G_1 \times \cdots \times G_d$ where each of G_i is a transitive closure or a 2-TC-spanner graph. Then there exists a monotonicity tester for G that satisfies that*

$$Q_{\epsilon}(G) = O\left(\frac{1}{\epsilon} \cdot \sum_{i=1}^{d} \frac{|E_i|}{|V_i|}\right).$$

Proof. By replacing the $\min_{i \leq d}(\mathrm{iso}^{-}(G_i))$ in Theorem 1 with $\mathrm{iso}^{-}(\mathrm{TC}(G)) = \mathrm{iso}^{-}(2-\mathrm{TC}(G)) = \Omega(1)$ from Lemma 4 □

In fact, we can also give a upper bound of 1 for iso^{-} for all graphs.

Lemma 5. *For any directed acyclic graph G,* $\mathrm{iso}^{-}(G) \leq 1$

Proof. Given a graph G, choose any maximal node s, and one of the maximal node that s covers, call it t. We set $f(s) = 0$ and $f(t) = 1$. For the rest of the graph, set $f(v) = 1$ if $t < v$, and $f(v) = 0$ otherwise. For this function, there is exactly one pair of violation and one point needs to be changed to obtain a monotone function. Therefore, $I^{-}(f) = 1/|V(G)|$ and $\mathrm{dist}_{\mathrm{mono}}(f) = 1/|V(G)|$, which gives $\mathrm{iso}^{-}(G) \leq 1$. □

[1] Note that this technique used in $k = 2$ case was first introduced by Bhattacharyya *et al.* [5].

4.4 $\mathcal{H}$-minor free Graphs

In this section, we will show how to extend the result of Bhattacharyya *et al.* [5] on $\mathcal{H}$-minor free graphs to products of such graphs.

First we will need to define graph minors.

Definition 15 (Graph minors)**.** An undirected graph H is a *minor* of G if H can be obtained by a sequence of edge contractions in G.

Based on this definition, we can also define a minor-closed family of graphs.

Definition 16 (Minor-closed families)**.** A family $\mathcal{F}$ is *minor-closed* if it contains every minor of every graph in $\mathcal{F}$.

Finally, $\mathcal{H}$-minor free families are special minor-closed families.

Definition 17 ($\mathcal{H}$-minor free families)**.** For a finitely large graph set $\mathcal{H}$ and a minor-closed family of graphs $\mathcal{F}$, $\mathcal{F}$ is *$\mathcal{H}$-minor-free* if for all $H \in \mathcal{H}$, $H \notin \mathcal{F}$.

The notion of $\mathcal{H}$-minor free graphs have lots of applications. For example, the family of all planar graphs are $\mathcal{H}$-minor free for $\mathcal{H} = \{K_5, K_{3,3}\}$.

In the paper of Bhattacharyya *et al.* [5], they show a construction of a 2-TC-spanner for any graph of a family of an $\mathcal{H}$-minor free graph, which gives the following theorem:

Theorem 4.3 of Bhattacharyya *et al.* [5]. *If G' belongs to an $\mathcal{H}$-minor-free graph family, and if G is a directed graph whose underlying undirected graph is G', then G has a 2-*TC*-spanner of size $O(|V(G)| \log^2 |V(G)|)$.*

Therefore, by applying our Theorem 1, we can have the following theorem for testing products of such graphs.

Theorem 5. *Let $G = G_1 \times \cdots \times G_d$, where G_i's all have an undirected underlying graph that belongs to some $\mathcal{H}_i$-minor-free family for some $\mathcal{H}_i$. Then we have*

$$Q_\epsilon(G) = O\left(\frac{1}{\epsilon} \cdot \sum_{i=1}^{d} \log^2 |V(G_i)|\right) = O\left(\frac{d}{\epsilon} \max_i \log^2 |V(G_i)|\right).$$

Proof. By the construction of Bhattacharyya *et al.* [5], we know that there is a 2-TC-spanner for each of G_i with $O(|V(G)|\log^2|V(G)|)$ edges. Therefore the ratios between the number of edges and the number of vertices in these 2-TC-spanners are $\log^2|V(G_i)|$ for all G_i's.

By Lemma 4, iso^- is 1 for the 2-TC-spanners. Thus, the result follows once we apply these numbers to Theorem 1 □

Note that the product graph G is not necessarily $\mathcal{H}$-minor free for some $\mathcal{H}_i$. Therefore our result is an improvement for such graphs as there is no specialized bound for such graphs.

4.5 Constant Comparable Graphs

If in graphs have limited comparability, then it is easy to test monotonicity over this type of graphs, and even their products

This type of graphs retains the property of a transitive closure without the extra edges, hence can be tested effectively under our framework. In fact, it has a query complexity of $O(dk^2/\epsilon)$ where d is the number of dimensions and k is the comparable constant mentioned above.

We will first define comparability formally.

Definition 18. A direct acyclic graph G is a *k-comparable graph* if for any $v \in V(G)$, v is comparable to at most k other vertices in G.

The two point line is a very special case of this type of graph with $k = 1$.

Theorem 6. *For any direct acyclic $G = G_1 \times \cdots \times G_d$, where in each of G_i, every node is only comparable to at most k other nodes, we have*

$$Q_\epsilon(G) = O(dk^2/\epsilon).$$

Proof. Notice that $E(G_i)/V(G_i) \leq k$ because every node can be adjacent to at most k other nodes.

And $\text{iso}^-(G_i) = \Omega(1/k)$ in such graphs, since for every pair of violation, at most $2k$ other nodes need to be changed to obtain a monotone function. And this does not propagate to more than constant number of nodes at second level, otherwise the graph will not be constant comparable. Therefore, by applying Theorem 1, we have $Q_\epsilon(G) = O(dk^2/\epsilon)$. □

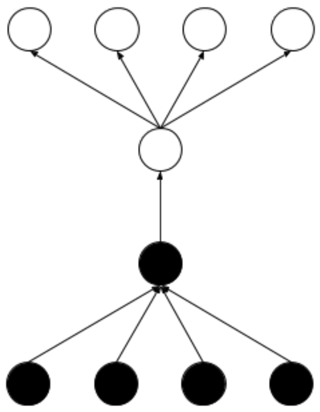

Figure 4.2: The top and bottom layer both have k nodes. In this function, we can see only 1 violating edge but have to change half of the graph to obtain a monotone function.

Note that the bound of the isoperimetric constant in this proof is reachable in the graph given in the following Figure 4.2:

This result is interesting because in the product graph, the comparability grows exponentially, and yet with our theorem we can still have a tester with linear query complexity.

Chapter 5

Improved Bound of Halevy and Kushilevitz

Here we will show how we can result from their original paper to construct a monotonicity tester that has a polynomial query complexity instead of exponential.

Theorem 7. *There is an ϵ monotonicity tester for boolean functions the graph product $G = G_1 \times \cdots \times G_d$ with query complexity $O(d^3/\epsilon) \cdot \max_i(Q_{\epsilon/d^2}(G_i))$.*

Our proof is built upon the same fundamental lemma, where Halevy and Kushilevitz [22] give a way to bound the distance to monotonicity in each dimension in terms of the distance to monotonicity of the entire product domain.

Lemma (Lemma 3.1 of Halevy and Kushilevitz [22])**.** *For any directed acyclic graph G, and function $f : G_1 \times G_2 \to \{0, 1\}$, we have*

$$\mathrm{dist}_{\mathrm{mono}}(f) \leq 2 \cdot (\mathrm{dist}^1_{\mathrm{mono}}(f) + \mathrm{dist}^2_{\mathrm{mono}}(f)).$$

The main technique novelty is to apply the lemma of Halevy and Kushilevitz recursively instead of iteratively to get the following corollary:

Corollary 1. *For any directed acyclic graph $G = G_1 \times \cdots \times G_d$, and function $f : G \to \{0, 1\}$, we have*

$$\mathrm{dist}_{\mathrm{mono}}(f) \leq d \cdot \sum_i \mathrm{dist}^i_{\mathrm{mono}}(f).$$

Proof. Without loss of generality, we can assume d is a power of 2. Since if otherwise, we can append graphs with a single vertex to the product without changing the resulting G. Now let $G' = (G_1 \times \cdots \times G_{d/2})$ and $G'' = \times(G_{d/2+1} \times \cdots \times G_d)$, so that we can also view $G = G_1 \times \cdots \times G_d$ as $G = G' \times G''$. We will use $\text{dist}'_{\text{mono}}$ and $\text{dist}''_{\text{mono}}$ to denote the distance to monotonicity by considering G' and G'' as the first and second dimension of G. Therefore we have

$$\text{dist}_{\text{mono}}(f) \le 2 \cdot (\text{dist}'_{\text{mono}}(f) + \text{dist}''_{\text{mono}}(f)).$$

We can repeat this process recursively on G' and G'' until we reach the base graphs. We will have

$$\text{dist}_{\text{mono}}(f) \le 2^{\log d} \cdot \sum_i \text{dist}^i_{\text{mono}}(f) = d \cdot \sum_i \text{dist}^i_{\text{mono}}(f).$$ □

We will also need the following lemma for the proof:

Lemma 6. *Let $\mathcal{D}$ be any distribution with range in $[0,1]$ that has $\mathbf{E}_{X\sim\mathcal{D}}[X] = \mu$, then*

$$\Pr_{X\sim\mathcal{D}}[X \ge \mu/2] \ge \frac{\mu}{2-\mu}.$$

Proof. This lemma is actually a special case of Markov's inequality. We will prove this for the sake of completeness for this thesis.

By the way of contradiction, assume that less than $\frac{\mu}{2-\mu}$ of the sample space has value less than $\mu/2$, then the expected value of a random sample from the distribution is less than

$$\frac{\mu}{2-\mu} \cdot 1 + (1 - \frac{\mu}{2-\mu}) \cdot \frac{\mu}{2} = \frac{\mu}{2-\mu} + \frac{\mu}{2} - \frac{\mu^2}{2(2-\mu)} = \mu$$

which is a contradiction. □

Now we are ready to prove Theorem 7, which is restated here:

Theorem 7 (Restated)**.** *There is an ϵ monotonicity tester for boolean functions the graph product $G = G_1 \times \cdots \times G_d$ with query complexity $O(d^3/\epsilon) \cdot \max_i(Q_{\epsilon/d^2}(G_i))$*

Proof. We claim that the following tester satisfies the query complexity condition.

From Corollary 1, we know that at least one of the dimensions has $\text{dist}^i_{\text{mono}}(f) \ge \epsilon/d^2$. By Lemma 6, we know at least $\frac{\epsilon}{2d^2-\epsilon}$ fraction of the slices are at least $\frac{\epsilon}{2d^2}$-far from monotone. Hence the rejection probability of applying the base tester T to a random slice in a random dimension is $\Omega(\frac{1}{d}) \cdot \Omega(\frac{\epsilon}{2d^2-\epsilon}) \cdot \Theta(1)$. If we repeat the base tester $\Theta(d^3/\epsilon)$ times with distance

```
1 **Repeat** $\Theta(d^3/\epsilon)$ times:
2 Choose $i \in [d]$ uniformly at random
3 Choose $\alpha \in G_1 \times \cdots \times G_{i-1}$ and $\beta \in G_{i+1} \times \cdots \times G_d$ uniformly at random.
4 Apply the base tester $T$ on $f|^i_{\alpha,\beta}$ with distance parameter set to $\frac{\epsilon}{2d^2}$
```

parameter set to ϵ/d^2, the base tester will catch a violation with high probability. Therefore the query complexity of the resulting tester is $O(d^3/\epsilon) \cdot Q_{\epsilon/d^2}(G)) = O(d^5/\epsilon) \cdot Q_\epsilon(G))$ when the base tester is a linear tester.[1] □

[1]A tester is linear if $Q_\epsilon(G)$ is linearly dependent on $1/\epsilon$

References

[1] Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Information and Computation*, 204(11):1704–1717, 2006.

[2] Tuğkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.

[3] Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016.

[4] Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lp-testing. *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014.

[5] Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

[6] Hadley Black, Deeparnab Chakrabarty, and Comandur Seshadhri. A $o(d) \cdot \text{polylog}(n)$ monotonicity tester for boolean functions over the hypergrid $[n]^d$. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2133–2151, 2018.

[7] Hadley Black, Deeparnab Chakrabarty, and Comandur Seshadhri. Domain reduction for monotonicity testing: A o(d) tester for boolean functions in d-dimensions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1975–1994, 2020.

[8] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

[9] Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *29th Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

[10] Deeparnab Chakrabarty and Comandur Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 419–428, 2013.

[11] Deeparnab Chakrabarty and Comandur Seshadhri. An o(n) monotonicity tester for boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2):461–472, 2016.

[12] Xi Chen, Anindya De, Rocco A Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015.

[13] Xi Chen, Rocco A Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *55th Annual Symposium on Foundations of Computer Science*, pages 286–295, 2014.

[14] Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 523–536, 2017.

[15] Fan RK Chung and Prasad Tetali. Isoperimetric inequalities for cartesian products of graphs. *Combinatorics Probability and Computing*, 7(2):141–148, 1998.

[16] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, pages 97–108, 1999.

[17] Funda Ergün, Sampath Kannan, S Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-Checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.

[18] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 474–483, 2002.

[19] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[20] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[21] Oded Goldreich, Shafi Goldwassert, Eric Lehman, and Dana Ron. Testing monotonicity. In *Proceedings 39th Annual Symposium on Foundations of Computer Science*, pages 426–435, 1998.

[22] Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Struct. Algorithms*, 33(1):44–67, 2008.

[23] Nathaniel Harms. Testing halfspaces over Rotation-Invariant distributions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, Proceedings, pages 694–713. 2019.

[24] Christian Houdré and Prasad Tetali. Isoperimetric invariants for product markov chains and graph products. *Combinatorica*, 24(3):359–388, 2004.

[25] Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM Journal on Computing*, 47(6):2238–2276, 2018.

[26] Sofya Raskhodnikova. Approximate testing of visual properties. In *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pages 370–381, 2003.

[27] Sofya Raskhodnikova. Transitive-Closure spanners: A survey. In Oded Goldreich, editor, *Property Testing: Current Research and Surveys*, pages 167–196. Berlin, Heidelberg, 2010.

[28] Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends® in Machine Learning*, 1(3):307–402, 2008.

[29] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.