

TEL AVIV UNIVERSITY

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING

Department of Electrical Engineering - Systems

**APPROXIMATING THE DISTANCE TO
MONOTONICITY AND CONVEXITY IN
SUBLINEAR TIME**

Thesis submitted toward the degree of
Master of Science in Electrical and Electronic Engineering
in Tel-Aviv University

by

Shahar Fattal

October 2006

TEL AVIV UNIVERSITY

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
Department of Electrical Engineering - Systems

**APPROXIMATING THE DISTANCE TO
MONOTONICITY AND CONVEXITY IN
SUBLINEAR TIME**

Thesis submitted toward the degree of
Master of Science in Electrical and Electronic Engineering
in Tel-Aviv University

by

Shahar Fattal

This research was carried out at Tel-Aviv University
in the Department of Electrical Engineering - Systems,
Faculty of Engineering
under the supervision of Prof. Dana Ron

October 2006

Abstract

In this thesis we study the problems of distance approximation to monotonicity and distance approximation to convexity. Namely, we are interested in (randomized) sublinear algorithms that approximate the Hamming distance between a given function and the closest monotone/convex function.

For the monotonicity property, we focus on functions over the d -dimensional hyper-cube, $[n]^d$, with any finite range. Previous work on distance approximation to monotonicity focused on the one-dimensional case and the only extension to higher dimensions was with an approximation factor exponential in the dimension d . We describe a reduction from the case of functions over the d -dimensional hyper-cube to the case of functions over the k -dimensional hyper-cube, where $k < d$. This reduction is efficient. That is, polynomial only in the additive error allowed, and not dependent on the size of the domain, range, or dimension. The quality of estimation that this reduction provides is linear in the size of the dimension and logarithmic in the size of the range. Using this reduction and a known distance approximation algorithm for the one dimensional case, we suggest a distance approximation algorithm for functions over the d -dimensional hyper-cube, with any finite range.

For the case of the Boolean range, we present solutions for distance approximation to monotonicity of functions over one dimension, two dimensions, and the k -dimensional hyper-cube (for any $k \geq 1$). Applying these algorithms and the reduction described above, we suggest a variety of distance approximation algorithms for Boolean range functions over the d -dimensional hyper-cube, which suggest a trade-off between quality of estimation and efficiency of computation.

For the convexity property, we present an efficient distance approximation algorithm for functions over one dimension, with any range. No solution for this problem was known before.

Contents

1	Introduction	4
1.1	Monotonicity and Convexity	5
1.1.1	Testing Monotonicity	5
1.1.2	Distance Approximation to Monotonicity	6
1.1.3	Testing Convexity	7
1.2	Our Results	7
1.2.1	Distance Approximation to Monotonicity	7
1.2.2	An Improved Testing Algorithm for Boolean Functions	8
1.2.3	Distance Approximation to Convexity	8
1.3	Techniques	9
1.3.1	Distance Approximation to Monotonicity	9
1.3.2	Testing of Monotonicity	10
1.3.3	Distance Approximation to Convexity	10
1.4	Organization	10
2	Distance Approximation to Monotonicity	12
2.1	Preliminaries	12
2.2	Dimension Reduction	13
2.3	Range Reduction	18
2.4	A Distance Approximation Algorithm based on Dimension Reduction	23
2.5	Distance Approximation for $\{0, 1\}$ Range and Low Dimension	25
2.5.1	Distance Approximation for $\Sigma^k \mapsto \{0, 1\}$ Functions ($k \geq 1$)	25
2.5.2	Distance Approximation for $\Sigma \mapsto \{0, 1\}$ Functions	32

2.5.3	Distance Approximation for $\Sigma^2 \mapsto \{0, 1\}$ Functions	39
2.6	An Improved Testing Algorithm for $\Sigma^d \mapsto \{0, 1\}$ Functions	45
2.7	Directions for Further Research	48
3	Distance Approximation to Convexity	51
3.1	On the Relation between Monotonicity and Convexity	51
3.2	Preliminaries	52
3.3	Pairs and the Co-convexity Property	53
3.4	<i>β-bigness</i> - An Indication to the Distance to Convexity	56
3.5	A Distance Approximation Algorithm	60
	Bibliography	68
	A Chernoff Bounds	70

List of Figures

2.1	An illustration of the violation graph of the given function f , and the generated function g . The domain is partitioned into 4 subsets, according to the labels of the function. The location of elements A, B, C and D is a result of their label in f and g respectively. That is, $f(A) \leq \frac{r}{2} - 2, f(B) = \frac{r}{2} - 1, f(C) \geq \frac{r}{2} + 1, f(D) \geq \frac{r}{2} + 1, g(A) = \frac{r}{2}, g(B) = \frac{r}{2}, g(C) \geq \frac{r}{2} + 1, g(D) = \frac{r}{2} - 1$. The dotted circle denotes the minimum vertex cover of $G_{viol}(f)$ ($A, B, D \in VC(G_{viol}(f))$ and $C \notin VC(G_{viol}(f))$). Recall that edges in the violation graph denote violations of monotonicity.	20
2.2	An illustration of a monotone function $f : \Sigma^2 \mapsto \{0, 1\}$ and a suitable g . In this case, $\Sigma = \{1, \dots, 15\}$ and $\delta = 1/5$. The labels of f are written in black. For example, $f(5, 9) = 0$ and $f(5, 10) = 1$. The bold dotted line stresses the limit between the 0 label area and the 1 label area of f . The labels of g are 0 in the gray areas of the domain, and 1 in the white areas of the domain.	41
2.3	Consider the case that $\Sigma = \{1, \dots, 15\}$ and $\delta = 1/5$. The domain of $f^{3,4}$ and $g^{3,4}$ is in gray. The domain of $f^{3,3}$ and $g^{3,3}$ is surrounded by a bold dotted line. The domain of $f^{4,4}$ and $g^{4,4}$ is surrounded by a bold line. By the time the algorithm calculates $dist(f^{3,4}, g^{3,4})$ it has already calculated $dist(f^{3,3}, g^{3,3})$ and $dist(f^{4,4}, g^{4,4})$	43
2.4	An illustration of f^u for the case of $d = 2, \Sigma = \{1, \dots, 9\}$ and $\epsilon_m = 1/3$	49
2.5	An illustration of f^ℓ for the case of $d = 2, \Sigma = \{1, \dots, 9\}$ and $\epsilon_m = 1/3$	49

Chapter 1

Introduction

Property Testing is a relaxation of decision problems. In a typical decision problem it is required to determine whether an object has or does not have a given property P . In the original notion [RS96, GGR98], a property testing algorithm is given access to the input object, which is usually represented as a function, via membership queries (i.e., the algorithm gives a value x , and receives $f(x)$). The algorithm is required to determine with high probability whether the function has property P , or whether it is *far* from having property P . In *far* from having the property we mean that the distance of the function to any function that has the property is above some given threshold ϵ . Distance is measured according to some natural distance measure such as the hamming distance. This relaxation allows for very efficient algorithms, with complexity that is *sub-linear* in the input size, and in many cases even *independent* of the input size. Examples of objects for which testing algorithms have been developed in the past few years are graphs, strings, functions, and geometrical objects (see [Gol98, Fis01, Ron01] for surveys).

We refer to the original notion of property testing as *standard* property testing. Namely, a standard property tester for property P is given a parameter ϵ and should accept with high probability objects that have the property, and reject with high probability objects that are ϵ -*far* from having the property. It is common to distinguish between two types of testers: One-sided error testers, which always accept objects that have the property, and Two-sided error testers, which accept, with high probability, objects that have the property. *Tolerant property testing* is a generalization of standard property testing where the algorithms are required to be more *tolerant* with respect to objects that do not have the property but are close to having the property. Namely, A *tolerant property testing algorithm* is required with high probability, to accept objects that are ϵ_1 -*close* to having a given property P and reject objects that are ϵ_2 -*far* from having property P , for some parameters $0 \leq \epsilon_1 < \epsilon_2 \leq 1$. It is of course desirable for the tolerant algorithm to run for any given ϵ_1 and ϵ_2 , and for its query complexity (and running time) to be sublinear in the input size and polynomial in $1/(\epsilon_2 - \epsilon_1)$. Observe that setting $\epsilon_1 = 0$ and allowing ϵ_2 to be a parameter gives the standard definition of property testing.

Another natural extension of standard property testing is *distance approximation (estimation)*. Let $\epsilon_P(f)$ denote the distance between f and the closest function that satisfies P . We say that $\hat{\epsilon}$ is an (α, δ) -*estimate* of $\epsilon_P(f)$ for $\alpha \geq 1$ and $0 \leq \delta \leq 1$ if it satisfies:

$$\frac{1}{\alpha}\epsilon_P(f) - \delta \leq \hat{\epsilon} \leq \epsilon_P(f) . \tag{1.1}$$

Observe that if $\alpha = 1$ and $\delta = 0$ then $\hat{\epsilon} = \epsilon_P(f)$. If an algorithm outputs an (α, δ) -estimate with high constant probability, then it is an (α, δ) -*estimate algorithm*. We note that an algorithm that provides an estimate $\hat{\epsilon}$ that satisfies $\epsilon_P(f) - \delta \leq \hat{\epsilon} \leq \alpha \epsilon_P(f) + \delta$ with high probability implies an $(\alpha, 2\delta)$ -estimate algorithm. Therefore, since the notion of (α, δ) -estimate is only used for a high level description of our final results, we will refer to both notions of approximation.

In some cases it is possible to obtain a purely additive approximation, that is a $(1, \delta)$ -estimate of f , where δ is a parameter to the algorithm, and the complexity of the algorithm has polynomial dependence on $1/\delta$. However, often it is only known how to get an estimate with larger α , and sometimes α is not a constant but dependent on the input in some way. We also note that it is possible to transform an algorithm that has an additive error to an algorithm that has a purely multiplicative error at the cost of a dependence of the complexity on $1/\epsilon_P(f)$.

Tolerant property testing and Distance approximation were first explicitly studied by Parnas et. al. [PRR04]. Following that work, there have been several results on distance approximation, both positive [ACCL04, GR05, FN05, MR06] and negative [FF05]. These works considered properties of functions and strings [PRR04, ACCL04, FF05, GR05], ensembles of points [PRR04], and graphs [FN05, MR06].

1.1 Monotonicity and Convexity

Our focus in this thesis is on distance approximation to monotonicity and convexity. Monotone and convex functions play an important role in many disciplines and applications, including combinatorial optimization, game theory, probability theory, and electronic trade.

We start by summarizing previous results on testing monotonicity, which is a property that has been extensively studied in the property testing literature [GGL⁺00, BRW05, EKK⁺00, DGL⁺99, Fis04, FLN⁺02, FN01, HK04, HK03, HK05]. We then turn to what is known about distance approximation to monotonicity [PRR04, ACCL04, AC05], and about testing of convexity [PRR03]. When stating the complexity of the algorithm we mean its query complexity. In all cases discussed below the running time is either linear in the query complexity or at most polynomial in it for a low degree polynomial.

1.1.1 Testing Monotonicity

Let V be some partially (or fully) ordered finite set, and let Ξ be some fully ordered set. A function $f : V \mapsto \Xi$ is *monotone* if every $x, y \in V$ such that $x \leq y$ satisfy $f(x) \leq f(y)$. Most previous work deals with the case that $V = \Sigma^d$ for a fully ordered set $\Sigma = \{1, \dots, n\}$ and $d \geq 1$. The (partial) order over Σ^d is simply the product order.

Ergun et. al. [EKK⁺00] considered the case of one-dimensional functions, that is, $d = 1$. In this case a function $f : \Sigma \rightarrow \Xi$ is monotone if and only if $f(1) \leq f(2) \leq \dots \leq f(n)$ (recall that $\Sigma = \{1, \dots, n\}$). They referred to the problem as “Spot checking of sorting” and gave an algorithm whose query complexity is $O(\log n/\epsilon)$. Batu et. al. [BRW05] extended the algorithm of [EKK⁺00] to higher dimensions at an exponential cost in the dimension. Namely, the complexity of their algorithm is $O((2 \log n)^d/\epsilon)$.

Goldreich et. al. [GGL⁺00] considered the case that $d \geq 1$ but $\Sigma = \Xi = \{0, 1\}$, and showed that in this case, a linear dependence on d suffices. Namely, their algorithm has query complexity $O(d/\epsilon)$. For general Σ (but keeping the range $\Xi = \{0, 1\}$), they gave one variant of their algorithm whose complexity is $O(d \log |\Sigma|/\epsilon)$ and another variant whose complexity is $O((d/\epsilon)^2)$. They also dealt with a general range Ξ at a multiplicative cost of $|\Xi|$. The dependence on $|\Xi|$ was reduced to logarithmic by Dodis et. al. [DGL⁺99], giving an algorithm for any function $f : \Sigma^d \rightarrow \Xi$ whose complexity is $O(d \log |\Sigma| \log |\Xi|/\epsilon)$. When $|\Xi| > |\Sigma| = n$ the upper bound is actually $O(d \log^2 n/\epsilon)$. We build on both the *dimension reduction* technique applied in [GGL⁺00] and [DGL⁺99] and on the *range reduction* technique applied in [DGL⁺99].

For small d (i.e., $d = O(\log \log n)$) Halevy and Kushilevitz [HK04] were able to obtain an improved query complexity of $O(d4^d \log n/\epsilon)$, and Ailon and Chazelle [AC05] further improved this bound to $O(d2^d \log n/\epsilon)$.

Halevy and Kushilevitz [HK03] also studied the problem of *distribution free* testing of monotonicity. In this variant of the problem, distance between functions is measured with respect to an unknown underlying distribution that is not necessarily uniform as in the standard definition (and the results described above). They gave a distribution free tester for monotonicity whose complexity is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$. If the underlying distribution D is a product distribution then $O(\frac{2^d H_D}{\epsilon})$ queries suffice [AC05] where H_D is the entropy of D .

Fischer et. al. [FLN⁺02] considered the case in which V is a general partially ordered set (poset). They showed that testing monotonicity of Boolean functions over general posets is equivalent to the problem of testing 2CNF assignments (namely, testing whether a given assignment satisfies a fixed 2CNF formula or is far from any such assignment). They also showed that for every poset, it is possible to test monotonicity over the poset with a number of queries that is sublinear in the size of the domain poset; specifically, the complexity grows like a square-root of the size of the poset. Finally, they gave some efficient algorithms for several special classes of posets (e.g., posets that are defined by trees).

Lower Bounds

Ergun et al. [EKK⁺00] gave a lower bound for non-adaptive testers, which, combined with a result of Fischer [Fis04], implies a lower bound of $\Omega(\log n)$ for one-dimensional functions (and a general range Ξ). For $d > 1$, Fischer et. al. [FLN⁺02] showed that every non-adaptive 1-sided error monotonicity tester requires $\Omega(\sqrt{d})$ queries, which implies an $\Omega(\log d)$ lower bound for the adaptive case. They also showed that every non-adaptive 2-sided error monotonicity tester requires $\Omega(\log d)$ queries, which implies an $\Omega(\log \log d)$ lower bound for the adaptive case. For Boolean functions over general posets [FLN⁺02] gave a lower bound of $N^{\Omega(\frac{1}{\log \log N})}$ on the query complexity of any non-adaptive tester, which implies a lower bound of $\Omega(\frac{\log N}{\log \log N})$ for any adaptive tester. Finally, for the distribution-free case, Halevy and Kushilevitz [HK05] showed that $2^{\Omega(d)}$ queries are necessary, so that an exponential dependence on the dimension is unavoidable in this case.

1.1.2 Distance Approximation to Monotonicity

Parnas et. al. [PRR04] studied the problem of distance approximation to monotonicity of one-dimensional functions $f : \Sigma \rightarrow \Xi$. Their algorithm is a $(2, \delta)$ -estimate algorithm and its query

complexity is $\tilde{O}((\log n)^7/\delta^4)$.¹

Ailon et. al. [ACCL04] improved on this result. The quality of their estimate is similar (and in particular there is a factor 2 error in the estimate), but their dependence on $\log n$ is linear, which is the best possible. They build on an idea from [EKK⁺00], which we adapt in our algorithm for distance approximation of convexity.

For higher dimensions and general Σ and Ξ , it is observed in [PRR04] that using a lemma of [AC05] (which improves on [HK04]), it is possible to get a $(d \cdot 2^{d+1}, \delta)$ -estimate using a number of queries as in the $d = 1$ case up-to logarithmic factors. Namely, the quality of the estimate degrades exponentially with the dimension.

For the special case of $\Sigma = \Xi = \{0, 1\}$, the algorithm and analysis in [GGL⁺00] imply that it is possible to get a $(2d, \delta)$ -estimate using $O(1/\delta^2)$ queries.

1.1.3 Testing Convexity

Definition 1.1.1 *Let $f : X \mapsto \mathbb{R}$ where X is a discrete domain. The function f is convex if for all $x, y \in X$ and for all $0 \leq \alpha \leq 1$ such that $\alpha x + (1 - \alpha)y \in X$, it holds that $f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$.*

Parnas et. al. [PRR03] studied both the above notion of convexity as well as an extension to two-dimensional functions/matrices (submodularity). For the case of convexity in one dimension they describe a testing algorithm whose complexity is $O(\frac{\log n}{\epsilon})$.

1.2 Our Results

Our results are summarized in the theorems below. We refer to algorithms that are described and analyzed in the chapters that follow. In some cases these algorithms execute as subroutines known algorithms (e.g., the one-dimensional algorithm of [AC05]). Recall that $\Sigma = \{1, \dots, n\}$.

1.2.1 Distance Approximation to Monotonicity

Theorem 1.2.1 is our most general result.

Theorem 1.2.1 (Distance approximation to monotonicity for $\Sigma^d \mapsto \Xi$ functions)

Algorithm 2.4.1 combined with the algorithm of [ACCL04] is a $(4.01 \cdot d \log |\Xi|, \delta)$ -estimate algorithm for monotonicity of $\Sigma^d \mapsto \Xi$ functions. The time and query complexity of the algorithm is $\tilde{O}\left(\frac{\log n}{\delta^3}\right)$.

When dealing with Boolean functions we can obtain the following results, which give trade-offs between the quality of the estimate and the complexity of the algorithm. As opposed to Theorem 1.2.1, in all these results there is no dependence on n .

¹The notation $\tilde{O}(g(\cdot))$ for a function g (possibly of several parameters), means $O(g(\cdot) \cdot \text{polylog}(g(\cdot)))$.

Theorem 1.2.2 (Distance approximation to monotonicity for $\Sigma^d \mapsto \{0,1\}$ functions)

1. For any given parameter $k \in [d]$ such that d/k is an integer, Algorithm 2.4.1 combined with Algorithm 2.5.1 provides a $((4d/k), \delta)$ -estimate algorithm for monotonicity of $\Sigma^d \mapsto \{0,1\}$ functions. The time and query complexity of the algorithm is $\frac{1}{\delta^2} \log \frac{1}{\delta} \cdot \left(O\left(\left(\frac{k}{\delta}\right)^2 \log \frac{k}{\delta}\right)\right)^{k+2} = \left(\tilde{O}\left(\frac{k}{\delta}\right)\right)^{2k+6}$.
2. Algorithm 2.4.1 combined with Algorithm 2.5.5 provides a (d, δ) -estimate algorithm for monotonicity of $\Sigma^d \mapsto \{0,1\}$ functions. The time and query complexity of the algorithm is $\tilde{O}\left(\frac{1}{\delta^6}\right)$.
3. Algorithm 2.4.1 combined with Algorithm 2.5.2 provides a $(2d, \delta)$ -estimate algorithm for monotonicity of $\Sigma^d \mapsto \{0,1\}$ functions. The time and query complexity of the algorithm is $\tilde{O}\left(\frac{1}{\delta^4}\right)$.
4. Algorithm 2.4.1 combined with Algorithm 2.5.4 provides a $(4d, \delta)$ -estimate algorithm for monotonicity of $\Sigma^d \mapsto \{0,1\}$ functions. The expected time and query complexity of the algorithm is $\tilde{O}\left(\frac{1}{\delta^3}\right)$.

1.2.2 An Improved Testing Algorithm for Boolean Functions

Recall that for Boolean functions (over Σ^d such that $|\Sigma| = n > 1$), [GGL⁺00] give algorithms whose complexity is either $O(d \log n / \epsilon)$ or $O((d/\epsilon)^2)$. By taking a slightly different approach we get the following result which is an improvement when d is significantly smaller than n .

Theorem 1.2.3 (Improved standard testing of Boolean functions) Algorithm 2.6.1 combined with Algorithm 2.6.2 provides a 1-sided error testing algorithm for monotonicity of $\Sigma^d \mapsto \{0,1\}$ functions. The time and query complexity of the algorithm is $\tilde{O}\left(\frac{d}{\epsilon}\right)$.

1.2.3 Distance Approximation to Convexity

For convexity there was no previously known distance approximation algorithm. We show:

Theorem 1.2.4 (Distance Approximation to convexity for $\Sigma \mapsto \mathbb{R}$ functions) Algorithm 3.5.2 is a $(25, \delta)$ -estimate algorithm for convexity of $\Sigma \mapsto \mathbb{R}$ functions. The expected time and query complexity of the algorithm is $\tilde{O}\left(\frac{\log n}{\max\{\epsilon_{con}(f), \delta\}}\right)$ where $\epsilon_{con}(f)$ is the distance between f and the closest convex function.

Observe that Theorem 1.2.4 implies that, by setting $\delta = 0$ we can get a purely multiplicative error (where the complexity depends on $1/\epsilon_{con}(f)$). We note that all our other results can be modified so that we obtain a similar type of result, but for simplicity we leave them as stated above.

1.3 Techniques

1.3.1 Distance Approximation to Monotonicity

One possible approach for distance approximation to monotonicity of high dimensional functions is to reduce this problem to the problem of approximating the distance to monotonicity of functions over the line (i.e., one-dimensional functions from Σ to Ξ) or to the problem of distance approximation of functions over the k -dimensional hyper-cube (i.e., $\Sigma^k \mapsto \Xi$, where $k < d$). For that matter, we build on the work of [DGL⁺99].

The algorithm of Dodis et. al. [DGL⁺99] works by selecting random pairs of points in Σ^d and checking whether monotonicity is violated by these pairs. Each pair belongs to the same line (projection of f to one dimension) and the pairs are selected according to a particular (non-trivial) distribution. They show that the probability of selecting a pair that violates monotonicity is lower bounded by $\Omega\left(\frac{\epsilon_{mon}(f)}{d \log |\Sigma| \log |\Xi|}\right)$ where $\epsilon_{mon}(f)$ is the distance between f and the closest monotone function (recall that $f : \Sigma^d \mapsto \Xi$). Hence, if $\epsilon_{mon}(f) > \epsilon$, then by selecting $\Theta(d \log |\Sigma| \log |\Xi| / \epsilon)$ pairs, violation of monotonicity is detected with high probability. However, the probability of selecting a violating pair is not necessarily upper bounded in a similar manner. In particular, the distribution over pairs is such that for some functions whose distance to monotonicity is very small, the probability of getting a violating pair is very big. This occurs because the distribution over pairs assigns large weight to some points (This difficulty does not arise in the special case of $\Sigma = \{0, 1\}$ because each line contains only a single pair). This means that estimating the distance to monotonicity in general requires to modify this approach.

What we take from [GGL⁺00] and [DGL⁺99] is the notions of *dimension reduction* and *range reduction*. Dimension reduction, which was touched upon in the previous paragraphs, means that we are interested in the relation between $\epsilon_{mon}(f)$ and $\epsilon_{mon}(f')$ for functions f' which correspond to projections of f to lower dimensional hypercubes Σ^k . We first extend the upper bound on $\epsilon_{mon}(f)$ in [DGL⁺99, Lemma 6], which applies to lines ($k = 1$), to higher dimensional hypercubes. We also give a simple lower bound on $\epsilon_{mon}(f)$. The upper bound holds only for the range $\{0, 1\}$, and hence we turn to giving a range reduction. In a range reduction we mean establishing a relation between the quality of estimates of the distance to monotonicity for functions with a general range Ξ , to the quality of such estimates in the case of $|\Xi| = 2$. Here we adapt a technique of [DGL⁺99] to our needs, and present an analysis which we believe is more intuitive and easier to follow.

Based on the dimension and range reduction lemmas, we get an algorithm that, combined with any distance approximation algorithm for low-dimensional functions, gives a distance approximation algorithm for higher dimensions. For general ranges we derive Theorem 1.2.1 by using the algorithm of [ACCL04] for one-dimensional functions as a subroutine. For the case $\Xi = \{0, 1\}$ we give several algorithms for low-dimensional functions. These algorithms, which are based on a variety of approaches, differ in the quality of the estimate they provide and their complexity. Using them we obtain the results stated in Theorem 1.2.2

1.3.2 Testing of Monotonicity

Our work on distance approximation led us to return to the problem of standard testing of monotonicity. We show that by using the technique of “bucketing” (e.g. [GR02]) it is possible to remove the dependence of the complexity of the algorithm on n (the size of Σ), at a cost of $\log^2(d/\epsilon)$. Thus, the complexity of our algorithm is $\tilde{O}(d/\epsilon)$ as compared to $O(d \log n/\epsilon)$ for this case in [GGL⁺00, DGL⁺99]. This is an improvement for low dimensions and large n .

1.3.3 Distance Approximation to Convexity

A basic idea behind some algorithms for distance approximation to monotonicity is to estimate the number of violations to monotonicity on the domain of the function, or on some parts of the domain of the function, according to some deterministic or probabilistic rule, and show that this estimation provides a fairly tight estimation of the distance to monotonicity. In our work we implement a similar idea for the convexity case. The first question that should be raised now is: What is a violation of convexity? In the monotonicity case this is quite simple. Two elements i and j violate monotonicity if $i < j$ and $f(i) > f(j)$, in which case a monotone function cannot be equal to f on both i and j . Also, determining whether i and j violate monotonicity requires $O(1)$ calculations. In the convexity case however, two elements in the domain can not contradict convexity (a function over a domain of size 2 is always convex). Consider some $i < j$. We easily show that if $f : \Sigma \mapsto \mathfrak{R}$ is convex then

$$f(i+1) - f(i) \leq \frac{f(j) - f(i+1)}{j - i - 1} \leq f(j+1) - f(j) \quad (1.2)$$

That is, if the pairs $\langle i, i+1 \rangle$ and $\langle j, j+1 \rangle$ don't satisfy Equation (1.2) then there is no convex function that is equal to f on all $\{i, i+1, j, j+1\}$. In this case we say that $\langle i, i+1 \rangle$ and $\langle j, j+1 \rangle$ violate convexity. The next thing to do is find some relation between those local violations to convexity and $\epsilon_{con}(f)$.

Consider the work of [ACCL04]. They define the term δ -big. An element i is δ -big if there exists j such that the number of violations to monotonicity between i and elements in $\{i+1, \dots, j\}$ (or $\{j, \dots, i-1\}$ if $j < i$) is large enough. Then they show a relation between δ -bigness and the distance to monotonicity (details in [ACCL04, Section 2.2]). We build on their work, where the violations to convexity as we showed above replace the violations to monotonicity.

1.4 Organization

In Chapter 2 we prove Theorems 1.2.1, 1.2.2 and 1.2.3. Specifically,

- In Section 2.2 we show a lower bound on $\epsilon_{mon}(f)$ for $f : \Sigma^d \mapsto \Xi$, and an upper bound on $\epsilon_{mon}(f)$ for $f : \Sigma^d \mapsto \{0, 1\}$.
- In Section 2.3 we extend the upper bound of Section 2.2 for $f : \Sigma^d \mapsto \{0, 1\}$ to an upper bound for $f : \Sigma^d \mapsto \Xi$, where Ξ is any finite set.

- In Section 2.4 we use the bounds of previous sections to provide a reduction from the problem of distance approximation of monotonicity of $\Sigma^d \mapsto \Xi$ functions to the problem of distance approximation of monotonicity of $\Sigma^k \mapsto \Xi$ functions, where $k < d$. Based on that we prove Theorem 1.2.1.
- In Section 2.5 we present distance approximation algorithms for $\Sigma^k \mapsto \{0, 1\}$ functions (Subsection 2.5.1), $\Sigma \mapsto \{0, 1\}$ functions (Subsection 2.5.2) and $\Sigma^2 \mapsto \{0, 1\}$ functions (Subsection 2.5.3). Based on those algorithms and the reduction of Section 2.4 we prove Theorem 1.2.2.
- In section 2.6 we show a reduction from the problem of testing monotonicity of $\Sigma^d \mapsto \{0, 1\}$ functions to the problem of testing monotonicity of $\Sigma \mapsto \{0, 1\}$ functions, and prove Theorem 1.2.3.

In Chapter 3 we prove Theorem 1.2.4. Specifically,

- In Section 3.3 we present the co-convexity property, and show the relevant implications of that property.
- In Section 3.4 we present the notion of β -bigness and show its relation to the distance of $f : \Sigma \mapsto \mathfrak{R}$ to convexity.
- In section 3.5 we present a distance approximation algorithm for convexity and prove Theorem 1.2.4.

Chapter 2

Distance Approximation to Monotonicity

In this chapter we prove Theorems 1.2.1, 1.2.2 and 1.2.3 (see pages 7 - 8).

2.1 Preliminaries

Let $d \in \mathcal{Z}$ and let $[d] = \{1, 2, \dots, d\}$. Let $\Sigma = \{1, \dots, n\}$ and Ξ be any fully ordered set. Given two strings x, y , let xy denote the concatenating of the two strings. Let $x, y \in \Sigma^d$ such that $x = x_1x_2\dots x_d$ and $y = y_1y_2\dots y_d$. We say that $x \leq y$ if every $i \in [d]$ satisfies $x_i \leq y_i$, and that $x < y$ if $x \leq y$ and there exists i such that $x_i < y_i$.

Definition 2.1.1 *In the following items, let V be some partially ordered finite set, and observe that Σ^d is a special case of V .*

- A function $f : V \mapsto \Xi$ is monotone if every $x \leq y$ satisfy $f(x) \leq f(y)$.
- Let the Violation Graph $G_{viol}(f) = (V, E_{viol}(f))$ of a function $f : V \mapsto \Xi$ be an undirected graph such that for every $x, y \in V$, $(x, y) \in E_{viol}(f)$ if x, y violate monotonicity (i.e. $x \leq y$ and $f(x) > f(y)$ or $y \leq x$ and $f(y) > f(x)$). Also, let $VC(G_{viol}(f)) \subseteq V$ denote the minimum vertex cover of $G_{viol}(f)$.
- Let $\epsilon_{mon}(f)$ denote the (relative) distance of $f : V \mapsto \Xi$ from the class of monotone functions, which is the minimum of $dist(f, g) = |\{x \in V : f(x) \neq g(x)\}|/|V|$ over all monotone functions $g : V \mapsto \Xi$.
- A function $f : V \mapsto \Xi$ is ϵ -far from being monotone if $\epsilon_{mon}(f) \geq \epsilon$, and ϵ -close to being monotone otherwise.

The Minimum Vertex Cover of graph G is denoted $VC(G)$. The following Lemma is quoted from [HK04].

Lemma 2.1.1 *Let $f : V \mapsto \Xi$ where V is some partially ordered finite set. Given $S \subseteq V$, if for every $(i, j) \in E_{\text{viol}}(f)$, either $i \in S$ or $j \in S$, then there exists a monotone function $f' : V \mapsto \Xi$ that differs from f only on elements in S .*

As a corollary of Lemma 2.1.1 we get the following lemma.

Lemma 2.1.2 *Let $f : V \mapsto \Xi$ where V is some partially ordered finite set.*

$$|VC(G_{\text{viol}}(f))| = \epsilon_{\text{mon}}(f) \cdot |V|$$

Given a function $f : \Sigma^d \mapsto \Xi$ and $1 \leq i \leq j \leq d$ let $f_{i,j,\alpha,\beta}(x) = f(\alpha x \beta)$ such that $\alpha \in \Sigma^{i-1}, x \in \Sigma^{j-i+1}, \beta \in \Sigma^{d-j}$. Note that $f_{i,j,\alpha,\beta}$ is a $\Sigma^{j-i+1} \mapsto \Xi$ function. To simplify our notation, for a fixed $k \in \mathcal{Z}$, let $f_{q,\alpha,\beta} = f_{(q-1) \cdot k + 1, q \cdot k, \alpha, \beta}$.

For $1 \leq i \leq j \leq d$ let (i, j) -cubes of a function f (and i -lines if $i = j$) denote the set $\{f_{i,j,\alpha,\beta} : \alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}\}$. For any $i \in [d]$ we say that a function f is *monotone in dimension i* , if the i -lines of f are all monotone functions. For a set $T \subseteq [d]$ we say that f is *monotone in dimensions T* , if for every $i \in T$, f is monotone in dimension i .

Definition 2.1.2 *Let $f : \Sigma^d \mapsto \Xi$, where Ξ is a fully ordered set. For every $i \in [d]$, the function $S_i[f] : \Sigma^d \mapsto \Xi$ is defined as follows: for every $\alpha \in \Sigma^{i-1}$ and $\beta \in \Sigma^{d-i}$, let $S_i[f](\alpha 1 \beta), \dots, S_i[f](\alpha n \beta)$ be assigned the values of $f(\alpha 1 \beta), \dots, f(\alpha n \beta)$ in sorted order. In other words, S_i acts on f by sorting its i -lines. Let the multi-dimensional sorting operator, $S_{i,j}[f] = S_j[S_{j-1}[\dots[S_i[f]\dots]]$ (note that $j \geq i$).*

The additive and multiplicative chernoff bounds which we often use can be found in Appendix A.

2.2 Dimension Reduction

The result of this section is presented in Lemmas 2.2.1 and 2.2.2, which provide a lower and an upper bound on $\epsilon_{\text{mon}}(f)$, respectively.

Lemma 2.2.1, which gives a lower bound on $\epsilon_{\text{mon}}(f)$ applies to functions with any range and its proof is quite simple. Lemma 2.2.2, which gives an upper bound on $\epsilon_{\text{mon}}(f)$ applies only to the range $\{0, 1\}$ and it is a generalization of [GGL⁺00, Lemma 9]. We Later extend the result of Lemma 2.2.2 to any fully ordered finite range, at a certain cost (see Lemma 2.3.1 on page 18).

Lemma 2.2.1 *Let $f : \Sigma^d \mapsto \Xi$, and $k \in \mathcal{Z}$ such that $\frac{d}{k} \in \mathcal{Z}$. The set of functions $\{f_{q,\alpha,\beta} : \Sigma^k \mapsto \Xi : q \in \{1, \dots, \frac{d}{k}\}, \alpha \in \Sigma^{(q-1) \cdot k}, \beta \in \Sigma^{d-q \cdot k}\}$ satisfies the following property.*

$$\epsilon_{\text{mon}}(f) \geq E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]$$

Lemma 2.2.2 Let $f : \Sigma^d \mapsto \{0, 1\}$, and $k \in \mathcal{Z}$ such that $\frac{d}{k} \in \mathcal{Z}$. The set of functions $\{f_{q,\alpha,\beta} : \Sigma^k \mapsto \{0, 1\} : q \in \{1, \dots, \frac{d}{k}\}, \alpha \in \Sigma^{(q-1) \cdot k}, \beta \in \Sigma^{d-q \cdot k}\}$ satisfies the following property.

$$\epsilon_{\text{mon}}(f) \leq \frac{2d}{k} E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]$$

Proof of Lemma 2.2.1: If f is monotone, then it is monotone in all dimensions. However, if f is monotone in a set of dimensions $\{i, i+1, \dots, j\}$, it is not necessarily monotone. Therefore, for every $1 \leq i \leq j \leq d$ the following inequality holds.

$$\epsilon_{\text{mon}}(f) \cdot n^d \geq \sum_{\alpha,\beta} \epsilon_{\text{mon}}(f_{i,j,\alpha,\beta}) \cdot n^{j-i+1} \quad (2.1)$$

Therefore,

$$\begin{aligned} \epsilon_{\text{mon}}(f) \cdot n^d &\geq \max_{q \in \{1, \dots, \frac{d}{k}\}} \sum_{\alpha,\beta} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}) \cdot n^k \\ &\geq \frac{k}{d} \sum_{q=1}^{d/k} \sum_{\alpha,\beta} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}) \cdot n^k \\ &\geq \frac{k \cdot n^k}{d} \sum_{q,\alpha,\beta} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}) \quad (2.2) \\ &= \frac{k \cdot n^k}{d} \cdot \frac{d}{k} \cdot n^{d-k} E_{q,\alpha,\beta}(\epsilon_{\text{mon}}(f_{q,\alpha,\beta})) \\ &= E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})] \cdot n^d \quad (2.3) \end{aligned}$$

■

Lemma 2.2.2 is based on the following three claims, which are established in Lemmas 2.2.4 and 2.2.3.

- When sorting a $\Sigma^d \mapsto \{0, 1\}$ function on some set of dimensions (using the sorting operator of Definition 2.1.2), the function remains sorted (if it was so before) on other dimensions (details in part 1 of Lemma 2.2.3).
- When sorting a $\Sigma^d \mapsto \{0, 1\}$ function on dimension $q \in [d]$, the (i, j) -cubes of f become closer to being monotone (details in part 2 of Lemma 2.2.3).
- The distance of $f : \Sigma^d \mapsto \{0, 1\}$ from its sorted version (i.e., $S_{1,d}[f]$) is similar to $\epsilon_{\text{mon}}(f)$ (details in Lemma 2.2.4).

Lemma 2.2.3 is a generalization of [GGL⁺00, Lemma 8].

Lemma 2.2.3 Let $h : \Sigma^d \mapsto \{0, 1\}$.

1. If h is monotone in dimensions $T \subseteq [d]$ then $S_j[h]$ is monotone in dimensions $T \cup \{j\}$.
2. For every $1 \leq i \leq j \leq d$ and $q \notin \{i, \dots, j\}$,

$$\sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta}) \geq \sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(S_q[h]_{i,j,\alpha,\beta})$$

Note that by part 1 of Lemma 2.2.3, $S_{1,d}[h]$ is a monotone function.

Proof: The first part of the lemma appears in [GGL⁺00, Lemma 8], and we include it for completeness. We turn to the second part of the lemma. Let us first present a claim, which we later prove.

Claim 2.2.1 *For every $1 \leq i \leq j \leq d$ and $q \notin \{i, \dots, j\}$, the function $f : \Sigma^{q-1} \times \{0, 1\} \times \Sigma^{d-q} \mapsto \{0, 1\}$ satisfies*

$$\sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(f_{i,j,\alpha,\beta}) \geq \sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(S_q[f]_{i,j,\alpha,\beta})$$

We now use Claim 2.2.1 to prove the second part of the lemma. Observe that the computation of $S_q[h]$ can be divided to sub-computations (in a similar way to the Bubble-Sort algorithm). That is, in every such sub-computation

- Pick some k and $k+1$.
- For every $\alpha \in \Sigma^{q-1}$ and $\beta \in \Sigma^{d-q}$, if $h_{q,q,\alpha,\beta}(k) > h_{q,q,\alpha,\beta}(k+1)$ then set $h_{q,q,\alpha,\beta}(k) = 0$ and $h_{q,q,\alpha,\beta}(k+1) = 1$.

Let h^t denote the resulted function after such t steps (sub-computations). Let k_t denote the ' k ' that was picked in step t . There exists a finite number t' such that $h^{t'} = S_q[h]$. Assume without loss of generality that $q > j$. Claim 2.2.1 implies that every $t \leq t'$ satisfies

$$\begin{aligned} & \sum_{\alpha \in \Sigma^{i-1}, \beta_1 \in \Sigma^{q-1-j}, \beta_2 \in \Sigma^{d-q}} \left(\sum_{r=k_t}^{k_t+1} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^{t-1}) \right) \\ & \geq \sum_{\alpha \in \Sigma^{i-1}, \beta_1 \in \Sigma^{q-1-j}, \beta_2 \in \Sigma^{d-q}} \left(\sum_{r=k_t}^{k_t+1} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^t) \right) \end{aligned}$$

Therefore,

$$\begin{aligned} & \sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta}^{t-1}) \\ & = \sum_{\alpha \in \Sigma^{i-1}, \beta_1 \in \Sigma^{q-1-j}, \beta_2 \in \Sigma^{d-q}} \left(\sum_{r \in \Sigma \setminus \{k_t, k_t+1\}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^{t-1}) + \sum_{r=k_t}^{k_t+1} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^{t-1}) \right) \\ & \geq \sum_{\alpha \in \Sigma^{i-1}, \beta_1 \in \Sigma^{q-1-j}, \beta_2 \in \Sigma^{d-q}} \left(\sum_{r \in \Sigma \setminus \{k_t, k_t+1\}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^t) + \sum_{r=k_t}^{k_t+1} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta_1 r \beta_2}^t) \right) \\ & = \sum_{\alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-j}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta}^t) \end{aligned}$$

which inductively proves the second part of the lemma.

Proof of Claim 2.2.1: Let $f^s = S_q[f]$. Let $f^M : \Sigma^{q-1} \times \{0, 1\} \times \Sigma^{d-q} \mapsto \{0, 1\}$ be a function such that for every $\alpha \in \Sigma^{i-1}$, $\beta \in \Sigma^{d-j}$, $f_{i,j,\alpha,\beta}^M$ is monotone and

$$\text{dist}(f_{i,j,\alpha,\beta}^M, f_{i,j,\alpha,\beta}) = \epsilon_{\text{mon}}(f_{i,j,\alpha,\beta}) \quad (2.4)$$

In order to prove the claim, it is left to show that there exists a function $f' : \Sigma^{q-1} \times \{0, 1\} \times \Sigma^{d-q} \mapsto \{0, 1\}$, such that for every $\alpha \in \Sigma^{i-1}$, $\beta \in \Sigma^{d-j}$, $f'_{i,j,\alpha,\beta}$ is monotone and

$$\sum_{\alpha,\beta} \text{dist}(f_{i,j,\alpha,\beta}^s, f'_{i,j,\alpha,\beta}) \leq \sum_{\alpha,\beta} \text{dist}(f_{i,j,\alpha,\beta}, f_{i,j,\alpha,\beta}^M) \quad (2.5)$$

Before we do so, we need some more notation. For simplicity, and without loss of generality, assume that $q = d, i = 1$ and $j < d$. Let $r \in \{0, 1\}$ and $\gamma \in \Sigma^{d-j-1}$. For every $x \in \Sigma^j$ let

$$\begin{aligned} g_{\gamma,r}(x) &= f(x\gamma r) \\ g'_{\gamma,r}(x) &= f'(x\gamma r) \\ g_{\gamma,r}^s(x) &= f^s(x\gamma r) \\ g_{\gamma,r}^M(x) &= f^M(x\gamma r) \end{aligned}$$

We now define f' (and g'). For every $x \in \Sigma^j$

$$\begin{aligned} g'_{\gamma,0}(x) &= f'(x\gamma 0) \\ &= \min\{g_{\gamma,0}^M(x), g_{\gamma,1}^M(x)\} \\ g'_{\gamma,1}(x) &= f'(x\gamma 1) \\ &= \max\{g_{\gamma,0}^M(x), g_{\gamma,1}^M(x)\} \end{aligned} \quad (2.6)$$

First we claim that $g'_{\gamma,0}, g'_{\gamma,1}$ are indeed monotone functions.

For every $y > x$,

$$\begin{aligned} g'_{\gamma,0}(y) &= \min\{g_{\gamma,0}^M(y), g_{\gamma,1}^M(y)\} \\ &\geq \min\{g_{\gamma,0}^M(x), g_{\gamma,1}^M(x)\} \\ &= g'_{\gamma,0}(x) \\ g'_{\gamma,1}(y) &= \max\{g_{\gamma,0}^M(y), g_{\gamma,1}^M(y)\} \\ &\geq \max\{g_{\gamma,0}^M(x), g_{\gamma,1}^M(x)\} \\ &= g'_{\gamma,1}(x) \end{aligned} \quad (2.7)$$

We also need to show that,

$$\sum_{\gamma} \text{dist}(g_{\gamma,0}^s, g'_{\gamma,0}) + \text{dist}(g_{\gamma,1}^s, g'_{\gamma,1}) \leq \sum_{\gamma} \text{dist}(g_{\gamma,0}, g_{\gamma,0}^M) + \text{dist}(g_{\gamma,1}, g_{\gamma,1}^M) \quad (2.8)$$

So we claim that for every γ ,

$$\text{dist}(g_{\gamma,0}^s, g'_{\gamma,0}) + \text{dist}(g_{\gamma,1}^s, g'_{\gamma,1}) \leq \text{dist}(g_{\gamma,0}, g_{\gamma,0}^M) + \text{dist}(g_{\gamma,1}, g_{\gamma,1}^M) \quad (2.9)$$

Proving Equation (2.9) is a technical process of going over the 16 possible values for the combinations of $g_{\gamma,r}, g_{\gamma,r}^M, r \in \{0, 1\}$. For every x_1 and γ such that

$$\begin{aligned} g_{\gamma,0}(x_1) = 0, \quad g_{\gamma,1}(x_1) = 1, \quad g_{\gamma,0}^M(x_1) = 1, \quad g_{\gamma,1}^M(x_1) = 0 \\ \text{OR} \\ g_{\gamma,0}(x_1) = 1, \quad g_{\gamma,1}(x_1) = 0, \quad g_{\gamma,0}^M(x_1) = 0, \quad g_{\gamma,1}^M(x_1) = 1 \end{aligned} \quad (2.10)$$

it is not hard to see that

$$\begin{aligned} &(g_{\gamma,0}^s(x_1) \oplus g'_{\gamma,0}(x_1)) + (g_{\gamma,1}^s(x_1) \oplus g'_{\gamma,1}(x_1)) \\ &= (g_{\gamma,0}(x_1) \oplus g_{\gamma,0}^M(x_1)) + (g_{\gamma,1}(x_1) \oplus g_{\gamma,1}^M(x_1)) - 2 \end{aligned} \quad (2.11)$$

otherwise

$$\begin{aligned} & (g_{\gamma,0}^s(x_1) \oplus g'_{\gamma,0}(x_1)) + (g_{\gamma,1}^s(x_1) \oplus g'_{\gamma,1}(x_1)) \\ = & (g_{\gamma,0}(x_1) \oplus g_{\gamma,0}^M(x_1)) + (g_{\gamma,1}(x_1) \oplus g_{\gamma,1}^M(x_1)) \end{aligned} \quad (2.12)$$

and the claim follows. \blacksquare

\blacksquare (Lemma 2.2.3)

Lemma 2.2.4 *Let $f : \Sigma^k \mapsto \{0, 1\}$. Then,*

$$\epsilon_{mon}(f) \leq \text{dist}(f, S_{1,k}[f]) \leq 2\epsilon_{mon}(f)$$

Proof: Part 1 of Lemma 2.2.3 implies that $S_{1,k}[f]$ is a monotone function, and so $\epsilon_{mon}(f) \leq \text{dist}(f, S_{1,k}[f])$.

Let $f^s = S_{1,k}[f]$. When sorting a function, the labels of the function are relocated. So, for every \hat{x} , if $f(\hat{x}) = 1$ and $f^s(\hat{x}) = 0$, then there must be some \hat{y} , such that, $f(\hat{y}) = 0$ and $f^s(\hat{y}) = 1$. Considering the operation of the multi-dimensional sorting operator $S_{1,k}$, it is not hard to see that $\hat{y} > \hat{x}$. In words, 1 labels of f , during the sorting, can only be moved to larger elements on the domain. Also, observe that $f(\hat{x}) > f(\hat{y})$ is a contradiction to monotonicity, and therefore (\hat{x}, \hat{y}) is an edge in $G_{viol}(f)$.

Let $X^0 = \{x \in \Sigma^k : f^s(x) = 0\}$ and $\bar{X}^0 = \{x_1, \dots, x_r\} \subseteq X^0$ be some set of elements such that $f(x_1) = f(x_2) = \dots = f(x_r) = 1$. Let $X^1 = \{x \in \Sigma^k : f^s(x) = 1\}$, $X^> = \{x \in \Sigma^k : \exists 1 \leq i \leq r. x > x_i\}$, and $X^{1,>} = X^1 \cap X^>$. Observe that the 1 labels on x_1, \dots, x_r , must be relocated (during the sorting) to elements that are in $X^{1,>}$. Also observe that if $x \in X^{1,>}$ then every $y > x$ is also in $X^{1,>}$. Therefore, 1 labels that are on elements in $X^{1,>}$ before the sorting, can not move out of that area during the sorting. Therefore, $|\{x \in X^{1,>} : f(x) = 0\}| \geq r = |\bar{X}^0|$, or else there are not enough places in $X^{1,>}$ for those 1 labels. Also observe that if $y \in X^{1,>}$ and $f(y) = 0$, then there must be $x_i \in \bar{X}^0$ such that (x_i, y) is an edge in $G_{viol}(f)$.

By Hall's theorem, the above implies that there exists a matching in $G_{viol}(f)$, between X^0 and X^1 , of size $e = |\{x \in X^0 : f(x) = 1\}|$. Therefore, $\epsilon_{mon}(f) \cdot n^k \geq e$.

Also, it is obvious that $|\{x \in X^0 : f(x) = 1\}| = |\{x \in X^1 : f(x) = 0\}|$. Therefore, $\text{dist}(f, f^s) \cdot n^k = 2e$, where $e = |\{x \in X^0 : f(x) = 1\}| = |\{x \in X^1 : f(x) = 0\}|$. Hence,

$$\epsilon_{mon}(f) \cdot n^k \geq e = \frac{\text{dist}(f, f^s) \cdot n^k}{2} \quad (2.13)$$

\blacksquare

Proof of Lemma 2.2.2: Let $f_q = S_{1,(q-1)k}[f]$. We claim that:

$$\epsilon_{mon}(f) \leq \text{dist}(f, f_{d/k+1}) \leq \sum_{q=1}^{d/k} \text{dist}(f_q, f_{q+1}) \quad (2.14)$$

The first inequality comes from the fact that $f_{d/k+1}$ is monotone, which is a result of part 1 of Lemma 2.2.3. The second comes from the triangle inequality.

For $\alpha \in \Sigma^{(q-1) \cdot k}$, $\beta \in \Sigma^{d-kq}$, let $g_{q,\alpha,\beta}(x) = f_q(\alpha x \beta)$. Now, we claim that for every q :

$$\begin{aligned}
n^d \cdot \text{dist}(f_q, f_{q+1}) &= \sum_{\alpha,\beta} |\{x \in \Sigma^k : f_q(\alpha x \beta) \neq f_{q+1}(\alpha x \beta)\}| \\
&= \sum_{\alpha,\beta} |\{x \in \Sigma^k : g_{q,\alpha,\beta}(x) \neq S_{1,k}[g_{q,\alpha,\beta}](x)\}| \\
&\leq \sum_{\alpha,\beta} \epsilon_{\text{mon}}(g_{q,\alpha,\beta}) \cdot 2n^k \\
&= 2n^d \cdot E_{\alpha,\beta}[\epsilon_{\text{mon}}(g_{q,\alpha,\beta})]
\end{aligned} \tag{2.15}$$

Inequality (2.15) is justified by Lemma 2.2.4.

Part 2 of Lemma 2.2.3 implies that for every $1 \leq q \leq \frac{d}{k}$,

$$E_{\alpha,\beta}[\epsilon_{\text{mon}}(g_{q,\alpha,\beta})] \leq E_{\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})] \tag{2.16}$$

and finally,

$$\begin{aligned}
\epsilon_{\text{mon}}(f) &\leq \sum_{q=1}^{d/k} \text{dist}(f_q, f_{q+1}) \\
&\leq \sum_{q=1}^{d/k} 2E_{\alpha,\beta}[\epsilon_{\text{mon}}(g_{q,\alpha,\beta})] \\
&\leq \sum_{q=1}^{d/k} 2E_{\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})] \\
&= \frac{2d}{k} E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]
\end{aligned} \tag{2.17}$$

and the lemma follows. \blacksquare

2.3 Range Reduction

In this section we present a reduction from the case of a general range Ξ to the case of the range $\{0, 1\}$. As a result we obtain Lemma 2.3.1.

Lemma 2.3.1 *Let $f : \Sigma^d \mapsto \Xi$, where Ξ is a fully ordered finite set. For every $k \in \mathcal{Z}$ such that $\frac{d}{k} \in \mathcal{Z}$, the set of functions $\{f_{q,\alpha,\beta} : \Sigma^k \mapsto \Xi : q \in \{1, \dots, \frac{d}{k}\}, \alpha \in \Sigma^{(q-1) \cdot k}, \beta \in \Sigma^{d-q \cdot k}\}$ satisfies the following property:*

$$\epsilon_{\text{mon}}(f) \leq \log |\Xi| \cdot \frac{2d}{k} E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]$$

The bound of Lemma 2.3.1, along with the bound of Lemma 2.2.1, implies a reduction algorithm from distance approximation for d -dimensional functions with finite range (of size $|\Xi|$), to k -dimensional functions (for $k \in \mathcal{Z}$ s.t. $\frac{d}{k} \in \mathcal{Z}$) with the same range. The cost of the reduction is a multiplicative factor of $2 \log |\Xi| \cdot d/k$ in the estimation (see Section 2.4 for details).

In Lemma 2.3.4, we present properties of operators that bound the distance to monotonicity for boolean range functions. We show that for operators with such properties, the relation to $\epsilon_{\text{mon}}(f)$ can be extended to functions with a larger range. In Lemma 2.3.1 we use it to extend the result of Lemma 2.2.2 to larger ranges.

The idea behind the range reduction: Consider the violation graph $G_{viol}(f)$ of a function $f : V \mapsto \Xi$, where V is some partially ordered domain, and $\Xi = [0, r - 1]$ is a fully ordered finite set. Lemma 2.1.2 implies that by changing the values of elements in the minimum vertex cover of $G_{viol}(f)$ (i.e. changing the values of the relevant elements in the domain V), f becomes monotone. The domain V can be divided into two subsets: The “Low” subset, which contains all elements x for which $f(x) \leq r/2 - 1$, and the “High” subset, which contains all elements x for which $f(x) \geq r/2$. Consider the bipartite graph $G' = (V, E')$ where $E' \subset E_{viol}(f)$ contains all the edges with one endpoint in the “Low” subset, and one endpoint in the “High” subset. We show in this section, that moving all elements in the minimum vertex cover of G' , from the low subset to the high subset, and from the high subset to the low subset (as described in Figure 2.1) generates a new function (name it g) for which there are no edges (in the violation graph of g) crossing from the low subset to the high subset (with respect to g). We also show that this process does not create any new edges in the violation graph. This allows us to separate the violations of the new generated function, g , to two groups: “Low” violations and “High” violations. This separation enables us to prove that the size of the range has a logarithmic effect on the bound that has been established in Lemma 2.2.2, details follow. Our proof is based on [DGL⁺99, section 4], in which a similar result is presented, followed, however, by a somehow less intuitive proof.

Notation: Given a function $f : V \mapsto [0, r - 1]$, where V is a partially ordered finite domain, and $[0, r - 1]$ is a fully ordered range of size r , where r is a power of 2. Recall that $G_{viol}(f) = (V, E_{viol}(f))$ is the violation graph of f , and $VC(G_{viol}(f))$ is a minimum vertex cover in $G_{viol}(f)$. Let $V_L(f)$ be the “low value” subset of V with respect to f . That is, $x \in V_L(f)$ iff $f(x) \leq \frac{r}{2} - 1$. Similarly, let $V_H(f)$ be the “high value” subset of V with respect to f . That is, $x \in V_H(f)$ iff $f(x) \geq \frac{r}{2}$. Let $f' : V \mapsto [0, 1]$ be defined as follows:

$$\begin{aligned} \text{if } x \in V_L(f) \text{ then } f'(x) &= 0 \\ \text{if } x \in V_H(f) \text{ then } f'(x) &= 1 \end{aligned} \tag{2.18}$$

Note that $E_{viol}(f')$ is the set of all edges $(x, y) \in E_{viol}(f)$ such that $x \in V_L(f)$, $y \in V_H(f)$ and $x > y$. Let $g : V \mapsto [r - 1]$ be defined as follows:

$$\begin{aligned} \text{if } x \in VC(G_{viol}(f')) \text{ and } x \in V_H(f) \text{ then } g(x) &= \frac{r}{2} - 1 \\ \text{if } x \in VC(G_{viol}(f')) \text{ and } x \in V_L(f) \text{ then } g(x) &= \frac{r}{2} \\ \text{if } x \notin VC(G_{viol}(f')) \text{ then } g(x) &= f(x) \end{aligned} \tag{2.19}$$

Observe that $dist(f, g) = \epsilon_{mon}(f')$.

Lemma 2.3.2 For function g as defined in Equation (2.19)

1. There are no edges in the violation graph of g , connecting vertices in $V_L(g)$ to vertices in $V_H(g)$ (i.e. There is no $(x, y) \in E_{viol}(g)$ such that $x \in V_L(g)$ and $y \in V_H(g)$).
2. $E_{viol}(g) \subseteq E_{viol}(f)$

Proof: We begin with the first part of the lemma. Assume that there is $(x, y) \in E_{viol}(g)$ such that $x \in V_L(g)$, $y \in V_H(g)$ and $x > y$. There can be two cases for x :

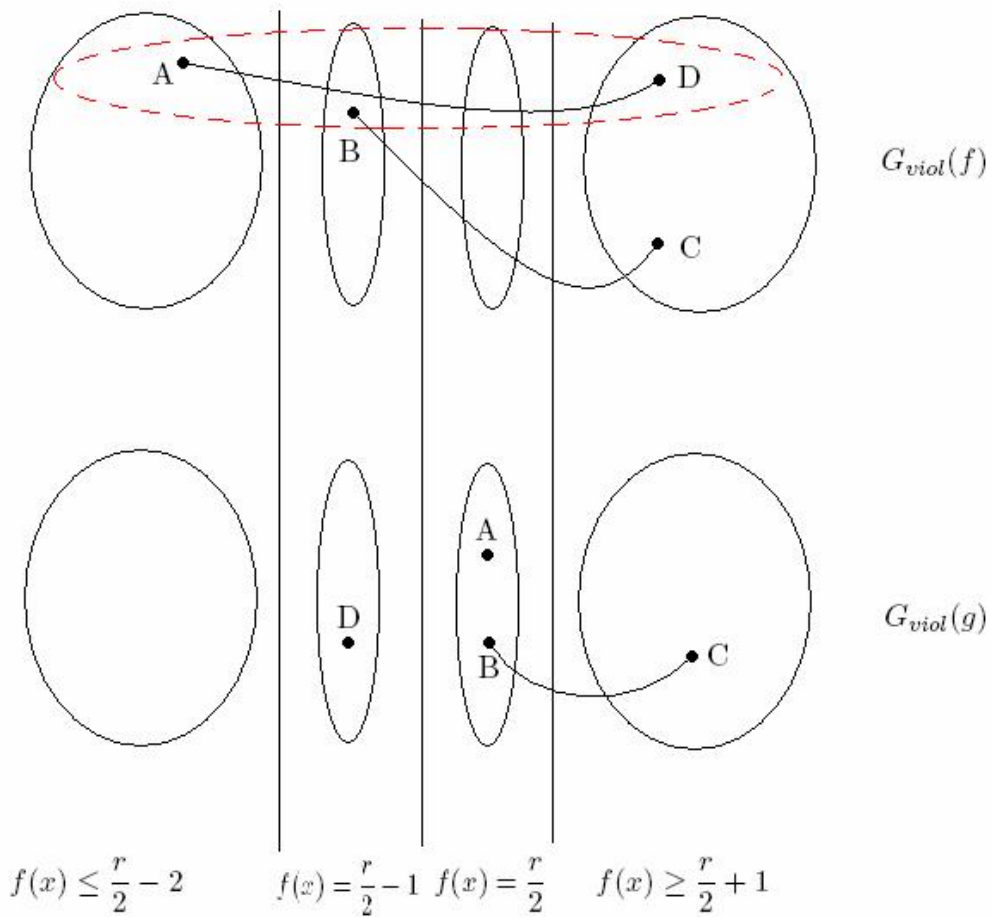


Figure 2.1: An illustration of the violation graph of the given function f , and the generated function g . The domain is partitioned into 4 subsets, according to the labels of the function. The location of elements A, B, C and D is a result of their label in f and g respectively. That is, $f(A) \leq \frac{r}{2} - 2$, $f(B) = \frac{r}{2} - 1$, $f(C) \geq \frac{r}{2} + 1$, $f(D) \geq \frac{r}{2} + 1$, $g(A) = \frac{r}{2}$, $g(B) = \frac{r}{2}$, $g(C) \geq \frac{r}{2} + 1$, $g(D) = \frac{r}{2} - 1$. The dotted circle denotes the minimum vertex cover of $G_{viol}(f)$ ($A, B, D \in VC(G_{viol}(f))$ and $C \notin VC(G_{viol}(f))$). Recall that edges in the violation graph denote violations of monotonicity.

1. $x \in V_L(f)$ and $x \notin VC(G_{viol}(f'))$
2. $x \in V_H(f)$ and $x \in VC(G_{viol}(f'))$

and two cases for y :

1. $y \in V_L(f)$ and $y \in VC(G_{viol}(f'))$
2. $y \in V_H(f)$ and $y \notin VC(G_{viol}(f'))$

Consider the first case for x . If $x \in V_L(f)$ and $x \notin VC(G_{viol}(f'))$ then y must be in case 1. (otherwise $y \in V_H(f)$ and since $x \in V_L(f)$ then $f'(x) < f'(y)$ which means that x or y must be in $VC(G_{viol}(f'))$, a contradiction). This means that $y \in V_L(f)$ and $y \in VC(G_{viol}(f'))$. Therefore there must exist $z \in V_H(f)$ such that $x > y > z$ and $z \notin VC(G_{viol}(f'))$. The fact that $z \in V_H(f)$ and $x \in V_L(f)$ means that $1 = f'(z) > f'(x) = 0$ which means that one of x, z must be in $VC(G_{viol}(f'))$. A contradiction.

Now consider the second case for x . If $x \in V_H(f)$ and $x \in VC(G_{viol}(f'))$ then there must exist $z > x > y$ such that $z \notin VC(G_{viol}(f'))$ and $z \in V_L(f)$. Therefore $y \in V_L(f)$ and $y \in VC(G_{viol}(f'))$ (otherwise $y \in V_H(f)$ and $y \notin VC(G_{viol}(f'))$ which cannot be since $f'(z) < f'(y)$ and $z > y$ means that $y \in VC(G_{viol}(f'))$). Now, since $y \in V_L(f)$ and $y \in VC(G_{viol}(f'))$, then there must exist $w < y$ such that $w \in V_H(f)$ and $w \notin VC(G_{viol}(f'))$. So $w < y < x < z$, $f'(w) = 1 > 0 = f'(z)$, which cannot be since both $w \notin VC(G_{viol}(f'))$ and $z \notin VC(G_{viol}(f'))$. The first part of the lemma follows.

For the second part of the lemma, consider an edge $(x, y) \in E_{viol}(g)$. By part 1 of the lemma, either x, y are both in $V_L(g)$, or they are both in $V_H(g)$. Assume (the other case is similar) that x, y are both in $V_L(g)$, $g(x) < g(y)$ and $x > y$.

First we show that x must be in $V_L(f)$ and not in $VC(G_{viol}(f'))$. Assume otherwise, i.e. $x \in V_H(f)$ and $x \in VC(G_{viol}(f'))$. In this case, $g(x) = \frac{r}{2} - 1$, and since we assume that $y \in V_L(g)$ then $g(y) \leq g(x)$ which contradicts our assumption on x, y . Therefore, $x \in V_L(f)$ and $x \notin VC(G_{viol}(f'))$, which means that $f(x) = g(x)$.

There can be two cases for y :

1. $y \in V_L(f)$ and $y \notin VC(G_{viol}(f'))$
2. $y \in V_H(f)$ and $y \in VC(G_{viol}(f'))$

If $y \in V_L(f)$ and $y \notin VC(G_{viol}(f'))$ then $f(y) = g(y)$ and $(x, y) \in E_{viol}(f)$. If, on the other hand, $y \in V_H(f)$ and $y \in VC(G_{viol}(f'))$, then $f(y) > \frac{r}{2} - 1 \geq g(x) = f(x)$ and $(x, y) \in E_{viol}(f)$. Part 2 of the lemma follows. ■

Let $g^L : V \mapsto [0, \frac{r}{2} - 1]$ be defined as follows:

$$\begin{array}{ll}
\text{if } x \in V_L(g) & \text{then } g^L(x) = g(x) \\
\text{if } x \in V_H(g) & \text{then } g^L(x) = \frac{r}{2} - 1
\end{array} \tag{2.20}$$

Let $g^H : V \mapsto [\frac{r}{2}, r - 1]$ be defined as follows:

$$\begin{aligned} \text{if } x \in V_H(g) & \quad \text{then } g^H(x) = g(x) \\ \text{if } x \in V_L(g) & \quad \text{then } g^H(x) = \frac{r}{2} \end{aligned} \tag{2.21}$$

Part 1 of Lemma 2.3.2 implies that:

- if $x \in V_H(g)$ then x is not connected to an edge in $E_{viol}(g^L)$.
- if $x \in V_L(g)$ then x is not connected to an edge in $E_{viol}(g^H)$.

Lemma 2.3.3 For g^L and g^H as defined in Equations (2.20) and (2.21), respectively,

1. $E_{viol}(f) \supseteq E_{viol}(g) = E_{viol}(g^L) \cup E_{viol}(g^H)$
2. $E_{viol}(f) \supseteq E_{viol}(f')$
3. $\epsilon_{mon}(f) \leq \epsilon_{mon}(f') + \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L)$

Proof: Part 1 of Lemma 2.3.2 implies that $E_{viol}(g^L) \cup E_{viol}(g^H) = E_{viol}(g)$. Part 2 of Lemma 2.3.2 implies that $E_{viol}(g) \subseteq E_{viol}(f)$ and part 1 of this lemma follows. Part 2 of this lemma is trivial from the definition of f' .

Regarding part 3. Part 1 of Lemma 2.3.2 and the fact that $V^H(g) \cap V^L(g) = \emptyset$ imply that $VC(G_{viol}(g^L)) \cup VC(G_{viol}(g^H)) = VC(G_{viol}(g))$ and $VC(G_{viol}(g^L)) \cap VC(G_{viol}(g^H)) = \emptyset$. Therefore $\epsilon_{mon}(g) = \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L)$. Let $g_{mon} : V \mapsto [0, r - 1]$ be a monotone function such that $dist(g, g_{mon}) = \epsilon_{mon}(g)$. Also recall that $\epsilon_{mon}(f') = dist(f, g)$. Therefore,

$$\begin{aligned} \epsilon_{mon}(f) & \leq dist(f, g_{mon}) \\ & \leq dist(f, g) + dist(g, g_{mon}) \\ & = \epsilon_{mon}(f') + \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L) \end{aligned} \tag{2.22}$$

and part 3 of the lemma follows. ■

Lemma 2.3.4 Let F be the set of all functions that map V to a finite range. Let $\Gamma : F \mapsto \mathfrak{R}$ be an operator that satisfies the following properties for every $f \in F$ (let f', g, g^L, g^H be defined as above with respect to f):

1. $\Gamma(f') \leq \Gamma(f)$
2. $\Gamma(g^H) + \Gamma(g^L) \leq \Gamma(f)$

If for every $f : V \mapsto \{0, 1\}$,

$$\epsilon_{mon}(f) \leq C \cdot \Gamma(f) \tag{2.23}$$

then for every $f : V \mapsto \Xi$,

$$\epsilon_{mon}(f) \leq C \cdot \lceil \log |\Xi| \rceil \cdot \Gamma(f)$$

Proof: Let $s = \lceil \log |\Xi| \rceil$. We prove the lemma by induction on s . The base of the induction follows directly from Equation (2.23). Now assume the lemma holds for $s-1$, and let $f : V \mapsto \Xi$. Therefore,

$$\epsilon_{mon}(f) \leq \epsilon_{mon}(f') + \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L) \quad (2.24)$$

$$\leq C \cdot \Gamma(f') + C \cdot (s-1) \cdot \Gamma(g^H) + C \cdot (s-1) \cdot \Gamma(g^L) \quad (2.25)$$

$$\leq C \cdot \Gamma(f) + C \cdot (s-1) \cdot \Gamma(f) \quad (2.26)$$

$$= C \cdot s \cdot \Gamma(f)$$

Inequality (2.24) is a result of part 3 of Lemma 2.3.3. Inequality (2.25) is based on the induction hypothesis, and the fact that the range of f' is of size 2, and the ranges of g^H and g^L are of size 2^{s-1} . Inequality (2.26) is based on the properties of Γ . ■

Proof of Lemma 2.3.1: Let $\Gamma(f) = \frac{2d}{k} E_{q,\alpha,\beta}(\epsilon_{mon}(f_{q,\alpha,\beta}))$. We need to show that the properties of Γ as described in Lemma 2.3.4 hold.

For every q, α, β , let $E_{viol}(f_{q,\alpha,\beta})$ denote the set of edges in the violation graph of $f_{q,\alpha,\beta}$ (i.e. $G_{viol}(f_{q,\alpha,\beta}) = (V, E_{viol}(f_{q,\alpha,\beta}))$). Part 2 of Lemma 2.3.3 implies that $E_{viol}(f') \subseteq E_{viol}(f)$. Therefore, for every q, α, β , we have that $E_{viol}(f'_{q,\alpha,\beta}) \subseteq E_{viol}(f_{q,\alpha,\beta})$, which means that $\epsilon_{mon}(f'_{q,\alpha,\beta}) \leq \epsilon_{mon}(f_{q,\alpha,\beta})$. Therefore Property 1 holds (i.e. $\Gamma(f') \leq \Gamma(f)$).

Part 1 of Lemma 2.3.3, and the fact that $V_H(g) \cap V_L(g) = \emptyset$, and $E_{viol}(g^H) \cap E_{viol}(g^L) = \emptyset$ imply that for every q, α, β ,

$$|VC(G_{viol}(g_{q,\alpha,\beta}))| \leq |VC(G_{viol}(f_{q,\alpha,\beta}))| \quad (2.27)$$

and

$$\begin{aligned} VC(G_{viol}(g_{q,\alpha,\beta}^H)) \cup VC(G_{viol}(g_{q,\alpha,\beta}^L)) &= VC(G_{viol}(g_{q,\alpha,\beta})) \\ VC(G_{viol}(g_{q,\alpha,\beta}^H)) \cap VC(G_{viol}(g_{q,\alpha,\beta}^L)) &= \emptyset \end{aligned} \quad (2.28)$$

and so

$$\begin{aligned} |VC(G_{viol}(g_{q,\alpha,\beta}^H))| + |VC(G_{viol}(g_{q,\alpha,\beta}^L))| &= |VC(G_{viol}(g_{q,\alpha,\beta}))| \\ &\leq |VC(G_{viol}(f_{q,\alpha,\beta}))| \end{aligned} \quad (2.29)$$

Therefore, Lemma 2.1.2 implies that for every q, α, β we have

$$\begin{aligned} \epsilon_{mon}(g_{q,\alpha,\beta}^H) + \epsilon_{mon}(g_{q,\alpha,\beta}^L) &\leq \epsilon_{mon}(g_{q,\alpha,\beta}) \\ &\leq \epsilon_{mon}(f_{q,\alpha,\beta}) \end{aligned} \quad (2.30)$$

and Property 2 holds (i.e. $\Gamma(g^H) + \Gamma(g^L) \leq \Gamma(f)$).

Lemma 2.2.2 implies that for every $f : V \mapsto \{0, 1\}$, $\epsilon_{mon}(f) \leq \Gamma(f)$ and the lemma follows. ■

2.4 A Distance Approximation Algorithm based on Dimension Reduction

Given some fixed $k \in [d]$ such that $d/k \in \mathcal{Z}$, let $A(f, \delta, \gamma)$ be an algorithm that is given query access to a function $f : \Sigma^k \mapsto \Xi$, and returns $\hat{\epsilon}$ such that with probability larger than $1 - \gamma$,

$\hat{\epsilon}$ satisfies $\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \eta \epsilon_{mon}(f) + \delta$ (for some $\eta \geq 1$). The query/time complexity of $A(f, \delta, \gamma)$ is $T(\delta, \gamma)$. We show here, based on Lemmas 2.2.1 and 2.3.1, and Algorithm A , a distance approximation algorithm (Algorithm 2.4.1) for monotonicity of $\Sigma^d \mapsto \Xi$ functions. Algorithm 2.4.1 is given query access to a function f and a parameter $0 < \delta \leq 1$ and returns $\hat{\epsilon}$ such that

Lemma 2.4.1 *At the end of Algorithm 2.4.1, with high probability,*

$$\frac{k}{2d \log |\Xi|} \epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \eta \epsilon_{mon}(f) + \frac{1 + \eta}{2} \delta$$

The query/time complexity of Algorithm 2.4.1 is $\frac{1}{\delta^2} T(\delta/2, \delta^2/6)$.

Before we prove Lemma 2.4.1, let us use it to prove Theorem 1.2.1.

Proof of Theorem 1.2.1: Consider the distance approximation algorithm for $\Sigma \mapsto \Xi$ functions presented by [ACCL04]. Their result implies the existence of a distance approximation algorithm A' that is given query access to a function $f : \Sigma \mapsto \Xi$, and parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$. Algorithm A' returns $\hat{\epsilon}$ such that with probability at least $1 - \gamma$, $\hat{\epsilon}$ satisfies $\epsilon_{mon}(f) - \Theta(\delta) \leq \hat{\epsilon} \leq 2.005 \epsilon_{mon}(f) + \Theta(\delta)$. Its time and query complexity is $T'(\delta, \gamma) = \tilde{O}\left(\frac{\log n}{\delta} \log \frac{1}{\gamma}\right)$. Lemma 2.4.1 implies that Algorithm 2.4.1, combined with Algorithm A' , provides a $(4.01 \cdot d \log |\Xi|, \delta)$ -estimate algorithm for $\Sigma^d \mapsto \Xi$ functions. The time and query complexity of the algorithm is $\tilde{O}\left(\frac{\log n}{\delta^3}\right)$. ■

Algorithm 2.4.1 *Approximating the Distance to Monotonicity by Dimension Reduction*

Given query access to a function $f : \Sigma^d \mapsto \Xi$ and a parameter δ :

Initialize $\hat{\epsilon} = 0$.

for $j = 1$ to $m = \Theta\left(\frac{1}{\delta^2}\right)$:

- Randomly and uniformly select $q \in \{1, \dots, \frac{d}{k}\}$, $\alpha \in \Sigma^{(q-1) \cdot k}$, $\beta \in \Sigma^{d-q \cdot k}$.
- Set $\hat{\epsilon}_j = A(f_{q, \alpha, \beta}, \frac{\delta}{2}, \frac{1}{6m})$.
- $\hat{\epsilon} = \hat{\epsilon} + \frac{\hat{\epsilon}_j}{m}$.

return $\hat{\epsilon}$.

Proof of Lemma 2.4.1: For convenience, set $\epsilon_j = \epsilon_{mon}(f_{q, \alpha, \beta})$, where $q \in \{1, \dots, \frac{d}{k}\}$, $\alpha \in \Sigma^{(q-1) \cdot k}$, $\beta \in \Sigma^{d-q \cdot k}$ were selected in iteration j of Algorithm 2.4.1. For every j , with probability larger than $1 - \frac{1}{6m}$, $\epsilon_j - \delta/2 \leq \hat{\epsilon}_j \leq \eta \cdot \epsilon_j + \delta/2$. Therefore, by the union bound, with probability larger than $1 - \frac{1}{6}$,

$$\begin{aligned} \frac{1}{m} \sum_{j=1}^m \hat{\epsilon}_j = \hat{\epsilon} &\geq \frac{1}{m} \sum_{j=1}^m (\epsilon_j - \delta/2) \\ &= \frac{1}{m} \left(\sum_{j=1}^m \epsilon_j \right) - \delta/2 \end{aligned} \tag{2.31}$$

$$\begin{aligned}
\frac{1}{m} \sum_{j=1}^m \hat{\epsilon}_j = \hat{\epsilon} &\leq \frac{1}{m} \sum_{j=1}^m (\eta \epsilon_j + \delta/2) \\
&= \frac{\eta}{m} \left(\sum_{j=1}^m \epsilon_j \right) + \delta/2
\end{aligned} \tag{2.32}$$

For convenience, set $p = E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]$. By the additive chernoff bound, $\Pr[|\frac{1}{m} \sum_{j=1}^m \epsilon_j - p| > \delta/2] < \frac{1}{6}$. Therefore, by the union bound, with probability larger the $1 - \frac{2}{6}$,

$$\begin{aligned}
\frac{1}{m} \sum_{j=1}^m \hat{\epsilon}_j = \hat{\epsilon} &\geq \frac{1}{m} \left(\sum_{j=1}^m \epsilon_j \right) - \delta/2 \\
&\geq p - \delta/2 - \delta/2 \\
&= E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})] - \delta
\end{aligned} \tag{2.33}$$

$$\begin{aligned}
\frac{1}{m} \sum_{j=1}^m \hat{\epsilon}_j = \hat{\epsilon} &\leq \frac{\eta}{m} \left(\sum_{j=1}^m \epsilon_j \right) + \delta/2 \\
&\leq \eta p + \eta \delta/2 + \delta/2 \\
&= \eta E_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})] + \frac{(1+\eta)}{2} \delta
\end{aligned} \tag{2.34}$$

Based on this result and Lemmas (2.2.1),(2.3.1) we get that with high probability,

$$\frac{k}{2d \log |\Xi|} \epsilon_{\text{mon}}(f) - \delta \leq \hat{\epsilon} \leq \eta \epsilon_{\text{mon}}(f) + \frac{1+\eta}{2} \delta \tag{2.35}$$

■

2.5 Distance Approximation for $\{0, 1\}$ Range and Low Dimension

In this section we present a variety of distance approximation algorithms for functions with $\{0, 1\}$ range and low dimension. Observe that each of these algorithms may be used with the dimension reduction algorithm (Algorithm 2.4.1).

2.5.1 Distance Approximation for $\Sigma^k \mapsto \{0, 1\}$ Functions ($k \geq 1$)

2.5.1.1 Results

The result of this subsection is Algorithm 2.5.1 (see page 29), which is a distance approximation algorithm for $\Sigma^k \mapsto \{0, 1\}$ functions. Algorithm 2.5.1 is given parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$

and query access to a function $f : \Sigma^k \mapsto \{0, 1\}$. It returns the value $\hat{\epsilon}$ such that (see Lemma 2.5.3 on page 29) with probability at least $1 - \gamma$

$$\epsilon_{mon}(f) - (k + 1)\delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(f) + (k + 2)\delta \quad (2.36)$$

Its time and query complexity is

$$\left(O\left(\frac{1}{\delta^2} \log(1/\delta)\right)\right)^{k+2} \log(1/\gamma)$$

Proof of Item 1 of Theorem 1.2.2: Algorithm 2.5.1 may also be viewed as an algorithm that returns $\hat{\epsilon}$ such that with probability at least $1 - \gamma$, $\hat{\epsilon}$ satisfies $\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(f) + \delta$, but with running time and query complexity of $\left(O\left(\left(\frac{k}{\delta}\right)^2 \log\left(\frac{k}{\delta}\right)\right)\right)^{k+2} \log(1/\gamma)$. Selecting $k \in \mathcal{Z}$ such that $d/k \in \mathcal{Z}$, applying this result on $\Sigma^k \mapsto \{0, 1\}$ functions, and using it with the reduction algorithm (see Lemma 2.4.1 and Algorithm 2.4.1), provides a distance approximation algorithm that is given a query access to $f : \Sigma^d \mapsto \{0, 1\}$ and returns $\hat{\epsilon}$ such that with high probability

$$\frac{k}{2d}\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(f) + \frac{3}{2}\delta \quad (2.37)$$

The time and query complexity of the algorithm is

$$\frac{1}{\delta^2} \left(O\left(\left(\frac{k}{\delta}\right)^2 \log\left(\frac{k}{\delta}\right)\right)\right)^{k+2} \cdot \log\frac{1}{\delta} \quad (2.38)$$

■

Observe the trade off that k provides. The larger k is, the less efficient the algorithm is, but the estimation of $\epsilon_{mon}(f)$ becomes more tight.

2.5.1.2 The Family $\{f_i : \Sigma^k \mapsto \{0, 1, *\}\}_{i=0}^k$

Lemma 2.2.4 (see page 17) implies that estimating the distance between a boolean range function to its sorted version (see Definition 2.1.2 for the sorting operator) provides a 2-factor distance approximation for monotonicity. Therefore, Algorithm 2.5.1 estimates the distance to the sorted version of the boolean function.

First we comment that when referring to a triplet (i, j_ℓ, j_h) such that $i \in [k]$, $j_\ell \in \Sigma^{i-1}$ and $j_h \in \Sigma^{k-i}$ then if $i = 1$ this actually means a pair $(1, j_h)$ such that $j_h \in \Sigma^{k-1}$ and if $i = k$ this actually means a pair (k, j_ℓ) such that $j_\ell \in \Sigma^{k-1}$. Let the order on the set $\{0, 1, *\}$ be $0 < * < 1$.

We would like to estimate the distance between f and $S_{1,k}[f]$. Therefore, we now define a set of functions $\{f_i : \Sigma^k \mapsto \{0, 1, *\}\}_{i=0}^k$ such that for every i , f_i is very close to $S_{1,i}[f]$. We show that for every i and $y \in \Sigma^k$, if $f_i(y) \neq *$ then $f_i(y) = S_{1,i}[f](y)$. We also show that the part of the domain for which the labels of f_i are $*$ is very small.

Definition 2.5.1 Given a function $f : \Sigma^k \mapsto \{0, 1\}$, let the set of functions $\{f_i : \Sigma^k \mapsto \{0, 1, *\}\}_{i=0}^k$ be defined recursively as follows. For every $0 < i \leq k$, $j_\ell \in \Sigma^{i-1}$ and $j_h \in \Sigma^{k-i}$ let

$$\begin{aligned} O_{i,j_\ell,j_h} &= |\{\acute{y} \in \Sigma : f_{i-1}(j_\ell \acute{y} j_h) = 1\}| \\ Z_{i,j_\ell,j_h} &= |\{\acute{y} \in \Sigma : f_{i-1}(j_\ell \acute{y} j_h) = 0\}| \end{aligned} \quad (2.39)$$

For $i = 0$, let $f_0 = f$. For every $1 \leq i \leq k$, $f_i : \Sigma^k \mapsto \{0, 1, *\}$ is defined as follows. For every $j_\ell \in \Sigma^{i-1}, j_h \in \Sigma^{k-i}$ and $\acute{y} \in \Sigma$

$$\begin{aligned}
& \text{if } \acute{y} \leq Z_{i,j_\ell,j_h} - \frac{\delta n}{2} \text{ then } f_i(j_\ell \acute{y} j_h) = 0 \\
& \text{if } \acute{y} \geq n - O_{i,j_\ell,j_h} + 1 + \frac{\delta n}{2} \text{ then } f_i(j_\ell \acute{y} j_h) = 1 \\
& \text{Else } f_i(j_\ell \acute{y} j_h) = *
\end{aligned} \tag{2.40}$$

Observe the following. For every $i \in [k]$

- f_i is ordered in the i^{th} dimension. That is, $S_i[f_i] = f_i$.
- For every $y \in \Sigma^k$

$$\text{if } f_i(y) = 0 \text{ then } S_i[f_{i-1}](y) = 0 \tag{2.41}$$

$$\text{if } f_i(y) = 1 \text{ then } S_i[f_{i-1}](y) = 1 \tag{2.42}$$

- For every $j_\ell \in \Sigma^{i-1}$ and $j_h \in \Sigma^{k-i}$

$$|\{\acute{y} \in \Sigma : f_i(j_\ell \acute{y} j_h) = *\}| \leq |\{\acute{y} \in \Sigma : f_{i-1}(j_\ell \acute{y} j_h) = *\}| + \delta n \tag{2.43}$$

Lemma 2.5.1 Consider the family $\{f_i : \Sigma^k \mapsto \{0, 1, *\}\}_{i=0}^k$ of Definition 2.5.1.

1. For every $y \in \Sigma^k$

$$\begin{aligned}
& \text{if } f_k(y) = 0 \text{ then } S_{1,k}[f](y) = 0 \\
& \text{if } f_k(y) = 1 \text{ then } S_{1,k}[f](y) = 1
\end{aligned} \tag{2.44}$$

2. For every $0 \leq i \leq k$, let $X_*(f_i) = \{y \in \Sigma^k : f_i(y) = *\}$.

$$|X_*(f_k)| \leq \delta \cdot k \cdot n^k$$

Proof: We prove the first item of the lemma by induction on i (the dimension).

Induction Base:

Equations (2.41) and (2.42) imply that for every $y \in \Sigma^k$

$$\begin{aligned}
& \text{if } f_1(y) = 0 \text{ then } S_{1,1}[f](y) = S_1[f_0](y) = 0 \\
& \text{if } f_1(y) = 1 \text{ then } S_{1,1}[f](y) = S_1[f_0](y) = 1
\end{aligned} \tag{2.45}$$

Induction Hypothesis:

Assume that for some $1 \leq i \leq k - 1$ we have that for every $y \in \Sigma^k$

$$\begin{aligned}
& \text{if } f_i(y) = 0 \text{ then } S_{1,i}[f](y) = 0 \\
& \text{if } f_i(y) = 1 \text{ then } S_{1,i}[f](y) = 1
\end{aligned} \tag{2.46}$$

This means that

$$\begin{aligned} \text{if } S_{i+1}[f_i](y) = 0 \text{ then } S_{i+1}[S_{1,i}[f]](y) = S_{1,i+1}[f](y) = 0 \\ \text{if } S_{i+1}[f_i](y) = 1 \text{ then } S_{i+1}[S_{1,i}[f]](y) = S_{1,i+1}[f](y) = 1 \end{aligned} \quad (2.47)$$

Equations (2.41) and (2.42) imply that

$$\begin{aligned} \text{if } f_{i+1}(y) = 0 \text{ then } S_{i+1}[f_i](y) = 0 \\ \text{if } f_{i+1}(y) = 1 \text{ then } S_{i+1}[f_i](y) = 1 \end{aligned} \quad (2.48)$$

And together we have that for every $y \in \Sigma^k$

$$\begin{aligned} \text{if } f_{i+1}(y) = 0 \text{ then } S_{1,i+1}[f](y) = 0 \\ \text{if } f_{i+1}(y) = 1 \text{ then } S_{1,i+1}[f](y) = 1 \end{aligned} \quad (2.49)$$

The first item of the lemma follows.

For the second item of the lemma, for every $i \in [k]$, Equation (2.43) implies that

$$|X_*(f_i)| \leq |X_*(f_{i-1})| + \delta \cdot n^k \quad (2.50)$$

Recall that $|X_*(f_0)| = 0$ and therefore for every $i \in [k]$

$$|X_*(f_i)| \leq \delta \cdot i \cdot n^k \quad (2.51)$$

and the second item of the lemma follows. ■

2.5.1.3 The Algorithm

We now present Algorithm 2.5.1, which estimates the distance between f_k and f . Lemma 2.5.1 implies that f_k and $S_{1,k}[f]$ are very close. Based on that and on Lemma 2.2.4 we show that this estimation is also an estimation of $\epsilon_{mon}(f)$.

Algorithm 2.5.1 *Distance Approximation for k Dimensions and $\{0, 1\}$ Range*

Given query access to a function $f : \Sigma^k \mapsto \{0, 1\}$ and parameters δ, γ : Let $\hat{m} = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\delta}))$. For every $0 \leq i \leq k$ and $y = (y_1, \dots, y_k) \in \Sigma^k$ consider the following procedure:

Estimate(i, y) {

- if $i = 0$ return $f(y)$
- Let $\hat{M} = \{\hat{y}_j\}_{j=1}^{\hat{m}}$ be a set of \hat{m} uniformly selected elements in Σ (including repetitions). For every $j \in [\hat{m}]$, let $r_j = \mathbf{Estimate}(i - 1, y_1 \cdots y_{i-1} \hat{y}_j y_{i+1} \cdots y_k)$.
- Let

$$\tilde{Z} = \frac{n}{\hat{m}} |\{j \in [\hat{m}] : r_j = 0\}|$$

- – if $y_i \leq \tilde{Z}$ then return 0.
- if $y_i \geq \tilde{Z} + 1$ then return 1.

}

Let $m = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\gamma}))$, and let $M = \{x^j\}_{j=1}^m$ be a set of m uniformly selected elements in Σ^k (including repetitions). For every $1 \leq j \leq m$ let $s_j = \mathbf{Estimate}(k, x^j)$.

Let

$$\hat{\epsilon} = \frac{1}{m} |\{j \in [m] : s_j \neq f(x^j)\}|$$

return $\hat{\epsilon}$

The time and query complexity of Algorithm 2.5.1 is

$$(\hat{m})^k \cdot m = \left(O\left(\frac{1}{\delta^2} \log(1/\delta)\right)\right)^{k+2} \log(1/\gamma) \quad (2.52)$$

Lemma 2.5.2 *For every $y \in \Sigma^k$, if $f_k(y) \neq *$ then with probability at least $1 - \frac{\delta}{8}$, $\mathbf{Estimate}(k, y) = f_k(y)$.*

Before we prove Lemma 2.5.2, we use it to prove that Algorithm 2.5.1 is indeed a distance approximation algorithm.

Lemma 2.5.3 *At the end of Algorithm 2.5.1, with probability at least $1 - \gamma$*

$$\epsilon_{mon}(f) - (k + 1)\delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(f) + (k + 2)\delta$$

Proof: Let

$$Y_1 = \{y \in \Sigma^k : f(y) \neq S_{1,k}[f](y)\} \quad (2.53)$$

Lemma 2.2.4 implies that

$$\epsilon_{mon}(f) \cdot n^k \leq |Y_1| \leq 2\epsilon_{mon}(f) \cdot n^k \quad (2.54)$$

Let

$$Y_2 = \{y \in \Sigma^k \setminus X_*(f_k) : f(y) \neq S_{1,k}[f](y)\} \quad (2.55)$$

Item (2) of Lemma 2.5.1 implies that

$$|Y_1| - k \cdot \delta n^k \leq |Y_2| \leq |Y_1| \quad (2.56)$$

Let

$$\hat{Y}_2 = \{x^j \in M \setminus X_*(f_k) : f(x^j) \neq S_{1,k}[f](x^j)\} \quad (2.57)$$

By the additive chernoff bound, with probability at least $1 - \frac{\gamma}{4}$

$$\left| \frac{|Y_2|}{n^k} - \frac{|\hat{Y}_2|}{m} \right| \leq \frac{\delta}{2} \quad (2.58)$$

Assume that this is the case. Item (1) of Lemma 2.5.1 implies that also

$$\hat{Y}_2 = \{x^j \in M \setminus X_*(f_k) : f(x^j) \neq f_k(x^j)\} \quad (2.59)$$

Lemma 2.5.2 and the additive chernoff bound imply that with probability at least $1 - \frac{\gamma}{2}$,

$$|\{j \in [m] : f_k(x^j) \neq * \text{ and } Estimate(k, x^j) \neq f_k(x^j)\}| \leq \frac{\delta}{4}m \quad (2.60)$$

Assume that this is the case. Let

$$\hat{Y}_3 = \{x^j \in M \setminus X_*(f_k) : f(x^j) \neq Estimate(k, x^j)\} \quad (2.61)$$

Equation (2.60) implies that

$$|\hat{Y}_2| - |\hat{Y}_3| \leq \frac{\delta}{4}m \quad (2.62)$$

Let

$$\hat{X} = \{x^j \in M : f_k(x^j) = *\} \quad (2.63)$$

By the additive chernoff bound, with probability at least $1 - \frac{\gamma}{4}$

$$\left| \frac{|\hat{X}|}{m} - \frac{|X_*(f_k)|}{n^k} \right| \leq \delta \quad (2.64)$$

Assume that this is indeed the case. Equation (2.64) and Item (2) of Lemma 2.5.1 imply that

$$\frac{|\hat{X}|}{m} \leq (k+1)\delta \quad (2.65)$$

Observe that $M \setminus \hat{X} = M \setminus X_*(f_k)$ and therefore

$$\hat{Y}_3 = \{x^j \in M \setminus \hat{X} : f(x^j) \neq Estimate(k, x^j)\} \quad (2.66)$$

Now recall that

$$\hat{\epsilon} = \frac{1}{m} |\{x^j \in M : f(x^j) \neq Estimate(k, x^j)\}| \quad (2.67)$$

Obviously $\frac{|\hat{Y}_3|}{m} \leq \hat{\epsilon}$. This and Equations (2.65) and (2.66) imply that

$$\frac{|\hat{Y}_3|}{m} \leq \hat{\epsilon} \leq \frac{|\hat{Y}_3|}{m} + \frac{|\hat{X}|}{m} \leq \frac{|\hat{Y}_3|}{m} + (k+1) \cdot \delta \quad (2.68)$$

Summing up the probabilities that Equation (2.58), Equation (2.60) or Equation (2.64) are not correct, and using Equations (2.54), (2.56), (2.58), (2.62) and (2.68) we have that with probability at least $1 - \gamma$

$$\epsilon_{mon}(f) - (k + 1)\delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(f) + (k + 2)\delta \quad (2.69)$$

■

Proof of Lemma 2.5.2: We show by induction on i that for every $y \in \Sigma^k$, if $f_i(y) \neq *$ then with probability at least $1 - \frac{\delta}{8}$, $Estimate(i, y) = f_i(y)$.

Induction Base:

Observe that for $i = 0$, $Estimate(0, y) = f(y) = f_0(y)$ for every $y \in \Sigma^k$.

Induction Hypothesis:

Consider any $y = y_1 \cdots y_k \in \Sigma^k$ and an iteration of $Estimate(i, y)$. Denote $y_\ell = y_1 \cdots y_{i-1}$ and $y_h = y_{i+1} \cdots y_k$.

By the induction hypothesis, for every $y \in \Sigma^k$, if $f_{i-1}(y) \neq *$ then with probability at least $1 - \frac{\delta}{8}$, $Estimate(i-1, y) = f_{i-1}(y)$.

Therefore, by the additive chernoff bound, with probability at least $1 - \frac{\delta}{16}$ we have that

$$\frac{n}{\hat{m}} |\{j \in [\hat{m}] : f_{i-1}(y_\ell y_j y_h) \neq * \text{ and } Estimate(i-1, y_\ell y_j y_h) \neq f_{i-1}(y_\ell y_j y_h)\}| \leq \frac{\delta}{4} n \quad (2.70)$$

Assume that this is indeed the case. Recall that

$$\tilde{Z} = \frac{n}{\hat{m}} |\{j \in [\hat{m}] : Estimate(i-1, y_\ell y_j y_h) = 0\}| \quad (2.71)$$

and let

$$\begin{aligned} \tilde{O} &= n - \tilde{Z} \\ &= \frac{n}{\hat{m}} |\{j \in [\hat{m}] : Estimate(i-1, y_\ell y_j y_h) = 1\}| \end{aligned} \quad (2.72)$$

Let

$$\begin{aligned} \hat{O} &= \frac{n}{\hat{m}} |\{j \in [\hat{m}] : f_{i-1}(y_\ell y_j y_h) = 1\}| \\ \hat{Z} &= \frac{n}{\hat{m}} |\{j \in [\hat{m}] : f_{i-1}(y_\ell y_j y_h) = 0\}| \end{aligned} \quad (2.73)$$

Equation (2.70) implies that

$$\begin{aligned} \tilde{O} &\geq \hat{O} - \frac{\delta}{4} n \\ \tilde{Z} &\geq \hat{Z} - \frac{\delta}{4} n \end{aligned} \quad (2.74)$$

By the additive chernoff bound, with probability at least $1 - \frac{\delta}{16}$,

$$\begin{aligned} \hat{O} &\geq O_{i, y_\ell, y_h} - \frac{\delta n}{4} \\ \hat{Z} &\geq Z_{i, y_\ell, y_h} - \frac{\delta n}{4} \end{aligned} \quad (2.75)$$

Observe that by the definition of f_i

$$\begin{aligned} O_{i,y_\ell,y_h} - \frac{\delta n}{2} &= |\{\acute{y} \in \Sigma : f_i(y_\ell \acute{y} y_h) = 1\}| \\ Z_{i,y_\ell,y_h} - \frac{\delta n}{2} &= |\{\acute{y} \in \Sigma : f_i(y_\ell \acute{y} y_h) = 0\}| \end{aligned} \quad (2.76)$$

Equations (2.74), (2.75) and (2.76) imply that

$$\begin{aligned} \tilde{O} &\geq |\{\acute{y} \in \Sigma : f_i(y_\ell \acute{y} y_h) = 1\}| \\ \tilde{Z} &\geq |\{\acute{y} \in \Sigma : f_i(y_\ell \acute{y} y_h) = 0\}| \end{aligned} \quad (2.77)$$

Summing up the probabilities that Equation (2.70) or Equation (2.75) are not correct, and using Equation (2.77) and the fact that f_i is sorted in the i^{th} dimension, we have that

- if $f_i(y) = 0$ then with probability at least $1 - \frac{\delta}{8}$, $y_i \leq \tilde{Z}$, which means that $Estimate(i, y) = 0$.
- if $f_i(y) = 1$ then with probability at least $1 - \frac{\delta}{8}$, $y_i \geq n - \tilde{O} + 1 = \tilde{Z} + 1$, which means that $Estimate(i, y) = 1$.

and the lemma follows. ■

2.5.2 Distance Approximation for $\Sigma \mapsto \{0, 1\}$ Functions

For the special case of $k = 1$, that is, for $\Sigma \mapsto \{0, 1\}$ functions, Algorithm 2.5.1 is a $(2, \delta)$ -estimate algorithm with time and query complexity of $\tilde{O}\left(\frac{1}{\delta^6} \log \frac{1}{\gamma}\right)$. As we show in this subsection, for this special case it is possible to get more efficient/tight results.

In the boolean range case, if the function is monotone, then there exists a *switch-point* $i \in \Sigma$ such that for every $j < i$, $f(j) = 0$, and for every $j \geq i$, $f(j) = 1$. Now let f be a function such that $\epsilon = \epsilon_{mon}(f)$. This means that there is an index (switch-point) $1 \leq i \leq n + 1$ such that $\epsilon \cdot n = |\{j : (j < i \text{ and } f(j) = 1) \text{ or } (j \geq i \text{ and } f(j) = 0)\}|$. The algorithms of this subsection are based on the idea of estimating this switch-point.

2.5.2.1 Approximation with no Multiplicative Factor

Algorithm 2.5.2 is given query access to a function $f : \Sigma \mapsto \{0, 1\}$ and parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$, and returns $\hat{\epsilon}$ such that

Lemma 2.5.4 *At the end of Algorithm 2.5.2, with probability larger than $1 - \gamma$,*

$$\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + \delta$$

The query and time complexity of Algorithm 2.5.2 is $\Theta\left(\frac{1}{\delta^2} \log\left(\frac{1}{\gamma \cdot \delta}\right)\right)$.

Proof of Item 3 of Theorem 1.2.2: Given a query access to a function $f : \Sigma^d \mapsto \{0, 1\}$, Algorithm 2.4.1 (see Lemma 2.4.1 on page 24), combined with Algorithm 2.5.2 (see Lemma 2.5.4), provides a distance to monotonicity estimation $\hat{\epsilon}$ such that with high probability

$$\frac{1}{2d}\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + \delta \quad (2.78)$$

The query and time complexity of the algorithm is $O(\frac{1}{\delta^4} \log(\frac{1}{\delta}))$. ■

Algorithm 2.5.2 *Distance Approximation for $\Sigma \mapsto \{0, 1\}$ Functions, Based on Partitioning*

Given query access to a function $f : \Sigma \mapsto \{0, 1\}$ and parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$:

- Take $m = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\gamma\delta}))$ random, uniform and independent samples of f .
- Let *switch-points* = $\{q \cdot \delta n + 1 : 0 \leq q \leq \frac{1}{\delta}\}$ be a set of $\frac{1}{\delta} + 1$ candidate switch-points, where the distance between every two consecutive candidate switch-points is δn .
- Let a_j be the index of the j^{th} sample. For each $i \in \text{switch-points}$ let $X_{i,j} = 1$ if $a_j < i$ and $f(a_j) = 1$ or $a_j \geq i$ and $f(a_j) = 0$. Otherwise let $X_{i,j} = 0$. Let $\hat{b}_i = \frac{1}{m} \sum_{j=1}^m X_{i,j}$. This can be most efficiently calculated in the following way: For every $0 \leq q < \frac{1}{\delta}$ Let

$$\begin{aligned} \text{ones}.q &= |\{j : q\delta + 1 \leq a_j \leq (q+1)\delta \text{ and } f(a_j) = 1\}| \\ \text{zeros}.q &= |\{j : q\delta + 1 \leq a_j \leq (q+1)\delta \text{ and } f(a_j) = 0\}| \end{aligned}$$

Hence

$$\begin{aligned} \hat{b}_1 &= \frac{1}{m} \sum_{q=0}^{\frac{1}{\delta}-1} \text{zeros}.q \\ \hat{b}_{(q+1)\delta+1} &= \hat{b}_{q\delta+1} - \frac{1}{m} \text{zeros}.q + \frac{1}{m} \text{ones}.q \end{aligned}$$

- Let $k \in \text{switch-points}$ be the one for which \hat{b}_k is minimized.
- Let $\hat{\epsilon} = \hat{b}_k - \frac{\delta}{2}$. Return $\hat{\epsilon}$.

Proof of Lemma 2.5.4: For every i and ℓ let $Y_{i,\ell} = 1$ if $\ell < i$ and $f(\ell) = 1$ or $\ell \geq i$ and $f(\ell) = 0$. Otherwise let $Y_{i,\ell} = 0$. Let $b_i = \frac{1}{n} \sum_{\ell=1}^n Y_{i,\ell}$. Note that b_i is the distance between f and the monotone function whose switch-point is i . For every $i \in \text{switch-points}$, by the additive chernoff bound, with probability at least $1 - \gamma \cdot \delta$

$$b_i - \frac{\delta}{4} \leq \hat{b}_i \leq b_i + \frac{\delta}{4} \quad (2.79)$$

By the union bound, with probability at least $1 - \gamma$, this is true for all $i \in \text{switch-points}$. Let $s \in \Sigma$ be the optimal switch-point. That is $b_s = \epsilon_{mon}(f)$. There exists $i \in \text{switch-points}$ such that

$$s - \frac{\delta \cdot n}{2} \leq i \leq s + \frac{\delta \cdot n}{2} \quad (2.80)$$

and therefore

$$b_s \leq \hat{b}_i \leq b_s + \frac{\delta}{2} \quad (2.81)$$

We also know that $\hat{b}_k \leq \hat{b}_i$. Therefore inequality (2.79) imply that $b_k \leq b_i + \frac{\delta}{2}$. Using inequality (2.81) and the fact that $b_s \leq b_k$, we get that

$$b_s \leq b_k \leq b_s + \delta \quad (2.82)$$

Using inequality (2.79) again

$$b_s - \frac{\delta}{4} \leq \hat{b}_k \leq b_s + \frac{5\delta}{4} \quad (2.83)$$

and therefore

$$\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + \delta \quad (2.84)$$

■

2.5.2.2 Approximation with a Constant Multiplicative Factor

In this subsection we present Algorithm 2.5.4 (see page 36). It is given query access to a function $f : \Sigma \mapsto \{0, 1\}$ and parameters $0 \leq \delta \leq 1$, $0 < \gamma_1 \leq 1$, and returns $\hat{\epsilon}$ such that

Lemma 2.5.5 • *At the end of Algorithm 2.5.4, with probability larger than $1 - \gamma_1$,*

$$\min\{\epsilon_{mon}(f)/2, \epsilon_{mon}(f) - \delta\} \leq \hat{\epsilon} \leq \epsilon_{mon}(f)$$

• *Let $\alpha = \max(\epsilon_{mon}(f), \delta)$. The expected time and query complexity of Algorithm 2.5.4 is*

$$O\left(\frac{1}{\alpha} \cdot \log \frac{1}{\alpha} \log \frac{\log \frac{1}{\alpha}}{\alpha \gamma_1}\right)$$

The quality of estimation of Algorithm 2.5.4 is not better than the quality of estimation of Algorithm 2.5.1 (see page 29) for the case of $k = 1$. However, if $\epsilon_{mon}(f)$ is not very small then it is more efficient. Observe that if $\epsilon_{mon}(f) > \delta$ then the time and query complexity of Algorithm 2.5.4 is not dependent on δ whereas the complexity of Algorithm 2.5.1 is polynomial in $1/\delta$.

We should note that $\delta = 0$ is allowed as a parameter to Algorithm 2.5.4, in which case Algorithm 2.5.4 is a $(2, 0)$ -estimate algorithm, with time and query complexity that depends only on $\frac{1}{\epsilon_{mon}(f)}$. We should also note that the method in which we construct Algorithm 2.5.4 enables to construct an (η, δ) -estimate algorithm, for any $\eta > 1$.

Proof of Item 4 of Theorem 1.2.2: Algorithm 2.4.1 (see Lemma 2.4.1 on page 24), combined with Algorithm 2.5.4 (see Lemma 2.5.5), provides a $(4d, \delta)$ -estimate algorithm. The expected time and query complexity of the algorithm is $\tilde{O}\left(\frac{1}{\delta^3}\right)$. ■

Algorithm 2.5.3 is a tolerant testing algorithm for $\Sigma \mapsto \{0, 1\}$ functions, based also on the idea of partitioning. The parameter μ is the multiplicative factor of the algorithm, and c sets the size of the partition. Algorithm 2.5.4 uses Algorithm 2.5.3 to search $\epsilon_{mon}(f)$ in a way that reminds a binary search. Details follow.

Algorithm 2.5.3 *Tolerant Testing for One Dimension and $\{0, 1\}$ Range*

Given query access to a function $f : \Sigma \mapsto \{0, 1\}$ parameters $\epsilon, \gamma, c \geq 1, \mu$ such that $\mu > 1 + \frac{1}{4c} > 1$:

- Let $R = \sqrt{\frac{\mu}{1 + \frac{1}{4c}}}$. Take $m = \Theta(\frac{1}{(R-1)^2 \cdot \epsilon} \log \frac{c}{\gamma \cdot \epsilon})$ random, uniform and independent samples of f .
- Let *switch-points* = $\{q \cdot \frac{\epsilon n}{2c} + 1 : 0 \leq q \leq \frac{2c}{\epsilon}\}$ be a set of $\frac{2c}{\epsilon} + 1$ candidate switch-points, where the distance between every two consecutive candidate switch-points is $\frac{\epsilon n}{2c}$.
- Let a_j be the index of the j^{th} sample. For each $i \in \text{switch-points}$ let $X_{i,j} = 1$ if $a_j < i$ and $f(a_j) = 1$ or $a_j \geq i$ and $f(a_j) = 0$. Otherwise let $X_{i,j} = 0$. Let $\hat{b}_i = \frac{1}{m} \sum_{j=1}^m X_{i,j}$. This can be most efficiently calculated in the following way: For every $0 \leq q < \frac{2c}{\epsilon}$ Let

$$\begin{aligned} \text{ones}.q &= |\{j : q \frac{\epsilon n}{2c} + 1 \leq a_j \leq (q+1) \frac{\epsilon n}{2c} \text{ and } f(a_j) = 1\}| \\ \text{zeros}.q &= |\{j : q \frac{\epsilon n}{2c} + 1 \leq a_j \leq (q+1) \frac{\epsilon n}{2c} \text{ and } f(a_j) = 0\}| \end{aligned}$$

Hence

$$\begin{aligned} \hat{b}_1 &= \frac{1}{m} \sum_{q=0}^{\frac{2c}{\epsilon}-1} \text{zeros}.q \\ \hat{b}_{(q+1)\frac{\epsilon n}{2c}+1} &= \hat{b}_{q\frac{\epsilon n}{2c}+1} - \frac{1}{m} \text{zeros}.q + \frac{1}{m} \text{ones}.q \end{aligned}$$

- If there exists $k \in \text{switch-points}$ such that $\hat{b}_k < \frac{\mu}{R}\epsilon$ then accept. Else reject.

Observe that the query complexity of Algorithm 2.5.3 is $Q(\epsilon, \gamma, \mu, c) = O(\frac{1}{(R-1)^2 \cdot \epsilon} \log \frac{c}{\gamma \cdot \epsilon})$, and the time complexity is $T(\epsilon, \gamma, \mu, c) = Q(\epsilon, \gamma, \mu, c) + O(\frac{c}{\epsilon})$. The interesting case is when μ comes close to 1. Therefore, assume that $1 < \mu \leq (1.25)^3$, set $c = \frac{1}{4(\sqrt[3]{\mu}-1)}$, and we have $R = \sqrt[3]{\mu}$. The query complexity is $Q(\epsilon, \gamma, \mu) = O(\frac{1}{(\sqrt[3]{\mu}-1)^2 \cdot \epsilon} \log \frac{1}{\gamma \cdot \epsilon \cdot (\sqrt[3]{\mu}-1)})$. Also, note that $\frac{c}{\epsilon} = \frac{1}{4\epsilon \cdot (\sqrt[3]{\mu}-1)} = o(\frac{1}{(\sqrt[3]{\mu}-1)^2 \cdot \epsilon} \log \frac{1}{\gamma \cdot \epsilon \cdot (\sqrt[3]{\mu}-1)})$. Therefore, $T(\epsilon, \gamma, \mu) = Q(\epsilon, \gamma, \mu)$.

Lemma 2.5.6 *If $\epsilon_{\text{mon}}(f) < \epsilon$ then Algorithm 2.5.3 accepts with probability at least $1 - \gamma$. If $\epsilon_{\text{mon}}(f) \geq \mu \cdot \epsilon$ it rejects with probability at least $1 - \gamma$.*

Proof: For every i and ℓ let $Y_{i,\ell} = 1$ if $\ell < i$ and $f(\ell) = 1$ or $\ell \geq i$ and $f(\ell) = 0$. Otherwise let $Y_{i,\ell} = 0$. Let $b_i = \frac{1}{n} \sum_{\ell=1}^n Y_{i,\ell}$. Observe that b_i is the distance between f and the monotone function whose switch-point is i .

Observe that

$$\begin{aligned} \Pr\left[\hat{b}_i \geq \frac{\mu}{R}\epsilon \mid b_i < (1 + \frac{1}{4c})\epsilon\right] &\leq \Pr\left[\hat{b}_i \geq \frac{\mu}{R}\epsilon \mid b_i = (1 + \frac{1}{4c})\epsilon\right] \\ &= \Pr\left[\hat{b}_i \geq R \cdot b_i \mid b_i = (1 + \frac{1}{4c})\epsilon\right] \end{aligned} \quad (2.85)$$

Therefore, for every $i \in \text{switch-points}$, if $b_i < (1 + \frac{1}{4c})\epsilon$ then by the multiplicative chernoff bound, with probability at least $1 - \frac{\gamma \cdot \epsilon}{2c}$

$$\hat{b}_i < \frac{\mu}{R}\epsilon \quad (2.86)$$

Also, if $b_i \geq \mu\epsilon$ then with probability at least $1 - \frac{\gamma\epsilon}{2c}$

$$\hat{b}_i \geq \frac{1}{R}b_i \quad (2.87)$$

By the union bound, with probability at least $1 - \gamma$, Equations (2.86), (2.87) are true for all $i \in \text{switch-points}$.

Let $s \in \Sigma$ be the optimal switch-point. That is $b_s = \epsilon_{\text{mon}}(f)$. There exists $i \in \text{switch-points}$ such that

$$s - \frac{\epsilon \cdot n}{4c} \leq i \leq s + \frac{\epsilon \cdot n}{4c} \quad (2.88)$$

and therefore

$$b_i \leq b_s + \frac{\epsilon \cdot n}{4c} \quad (2.89)$$

So if $b_s = \epsilon_{\text{mon}}(f) < \epsilon$ then

$$b_i < \epsilon + \frac{\epsilon}{4c} = (1 + \frac{1}{4c})\epsilon \quad (2.90)$$

Therefore inequality (2.86) implies that

$$\hat{b}_i < \frac{\mu}{R}\epsilon \quad (2.91)$$

Now assume that $b_s = \epsilon_{\text{mon}}(f) \geq \mu\epsilon$. So for every i

$$b_i \geq b_s = \epsilon_{\text{mon}}(f) \geq \mu\epsilon \quad (2.92)$$

and inequality (2.87) implies that for every i

$$\hat{b}_i \geq \frac{\mu}{R}\epsilon \quad (2.93)$$

■

Next we use Algorithm 2.5.3 to provide a distance approximation algorithm. For the sake of simplicity, let $\mu = 1.25$ and $c = \frac{1}{4(\sqrt[3]{1.25}-1)}$, which provides a factor 1.25 tolerant testing algorithm, with time and query complexity of $O(\frac{1}{\epsilon} \log \frac{1}{\gamma\epsilon})$.

Algorithm 2.5.4 *Distance Approximation for One Dimension and $\{0, 1\}$ Range Based on Tolerant Testing*

Given query access to a function $f : \Sigma \mapsto \{0, 1\}$ and parameters $0 \leq \delta \leq 1$, $0 < \gamma_1 \leq 1$:

First define the following procedure:

Estimate-Iter(L, H, δ, ν) {

- if $H < 2L$ or $H - L < \delta$ then return L .
- Run Algorithm 2.5.3 with $\mu = 1.25$, $c = \frac{1}{4(\sqrt[3]{1.25}-1)}$, $\epsilon = \frac{H+L}{2.25}$ and $\gamma = \frac{\gamma_1}{2 \cdot \nu^2}$.
If it accepts return Estimate-Iter($L, \frac{L+2H}{3}, \delta, \nu + 1$).
Else return Estimate-Iter($\frac{2L+H}{3}, H, \delta, \nu + 1$) }

Return $\hat{\epsilon} = \text{Estimate-Iter}(0, 1, \delta, 1)$

Proof of Lemma 2.5.5 (see page 34): The probability that Algorithm 2.5.3 gives a wrong answer depends on ν at the time it is called and is $\frac{\gamma_1}{2\nu^2}$. Note that ν increases each time. Therefore, by the union bound, the probability that Algorithm 2.5.3 gives a wrong answer at least once is at most

$$\sum_{\nu=1}^{\infty} \frac{\gamma_1}{2\nu^2} < \gamma_1 \quad (2.94)$$

Now assume that Algorithm 2.5.3 gives a correct answer each time it is called. Let $\epsilon_\nu, L_\nu, H_\nu$ be the value of ϵ, L and H respectively in iteration ν . We show by induction that for every ν we have $L_\nu \leq \epsilon_{mon}(f) \leq H_\nu$. Obviously this is correct for $\nu = 1$. Assume that for some ν , $L_\nu \leq \epsilon_{mon}(f) \leq H_\nu$. Recall that $2L_\nu < H_\nu$. Which means that

$$\begin{aligned} \epsilon_\nu &= \frac{L_\nu + H_\nu}{2.25} \geq \frac{2L_\nu + H_\nu}{3} \\ 1.25\epsilon_\nu &= \frac{1.25(L_\nu + H_\nu)}{2.25} \leq \frac{L_\nu + 2H_\nu}{3} \end{aligned} \quad (2.95)$$

There are three cases for the relation between $\epsilon_{mon}(f)$ and ϵ_ν :

- $\epsilon_\nu \leq \epsilon_{mon}(f) \leq 1.25\epsilon_\nu$. Therefore $\epsilon_{mon}(f)$ satisfies,

$$\begin{aligned} L_{\nu+1} &\leq \frac{2L_\nu + H_\nu}{3} \leq \frac{L_\nu + H_\nu}{2.25} = \epsilon_\nu \\ &\leq \epsilon_{mon}(f) \\ &\leq 1.25\epsilon_\nu = \frac{1.25(L_\nu + H_\nu)}{2.25} \leq \frac{L_\nu + 2H_\nu}{3} \leq H_{\nu+1} \end{aligned}$$

- $\epsilon_{mon}(f) \geq 1.25\epsilon_\nu$, and then $L_{\nu+1} = \frac{2L_\nu + H_\nu}{3}$ and $H_{\nu+1} = H_\nu$ and so $\epsilon_{mon}(f)$ satisfies,

$$L_{\nu+1} = \frac{2L_\nu + H_\nu}{3} < 1.25\epsilon_\nu \leq \epsilon_{mon}(f) \leq H_\nu = H_{\nu+1} \quad (2.96)$$

- $\epsilon_{mon}(f) \leq \epsilon_\nu$, and then $L_{\nu+1} = L_\nu$ and $H_{\nu+1} = \frac{L_\nu + 2H_\nu}{3}$ and so $\epsilon_{mon}(f)$ satisfies,

$$L_{\nu+1} = L_\nu \leq \epsilon_{mon}(f) \leq \epsilon_\nu \leq \frac{L_\nu + 2H_\nu}{3} = H_{\nu+1} \quad (2.97)$$

Therefore for every ν

$$L_\nu \leq \epsilon_{mon}(f) \leq H_\nu \quad (2.98)$$

Assume that Algorithm 2.5.4 performs ν' steps. That is, $\hat{\epsilon} = L_{\nu'}$.

- If $H_{\nu'} - L_{\nu'} < \delta$ then $L_{\nu'}$ satisfies

$$\epsilon_{mon}(f) - \delta \leq H_{\nu'} - \delta \leq L_{\nu'} \leq \epsilon_{mon}(f) \quad (2.99)$$

- Else if $H_{\nu'} < 2L_{\nu'}$ then $L_{\nu'}$ satisfies

$$\frac{\epsilon_{mon}(f)}{2} \leq \frac{H_{\nu'}}{2} \leq L_{\nu'} \leq \epsilon_{mon}(f) \quad (2.100)$$

Therefore at the end of Algorithm 2.5.4 we have

$$\min\{\epsilon_{mon}(f)/2, \epsilon_{mon}(f) - \delta\} \leq \hat{\epsilon} \leq \epsilon_{mon}(f) \quad (2.101)$$

and the first part of the lemma follows.

For the second part of the lemma, we start by analyzing the complexity (time and query) in the case that Algorithm 2.5.3 gives a correct answer each time it is called. We need to upper bound $\frac{1}{H_\nu - L_\nu}$ and $\frac{1}{\epsilon_\nu}$.

- By the definition of δ , it is always smaller than the size of $H_\nu - L_\nu$. Also, the stopping condition of Algorithm 2.5.4 implies that $H_\nu > 2L_\nu$, which means that $H_\nu - L_\nu > 0.5H_\nu \geq 0.5\epsilon_{mon}(f)$. And so $\frac{1}{H_\nu - L_\nu} = O(1/\alpha)$.
- Equation (2.98) implies that $\epsilon_\nu = \frac{L_\nu + H_\nu}{2.25} \geq \frac{H_\nu}{2.25} \geq \frac{\epsilon_{mon}(f)}{2.25}$. Also, $\epsilon_\nu = \frac{L_\nu + H_\nu}{2.25} \geq \frac{H_\nu - L_\nu}{2.25} \geq \frac{\delta}{2.25}$. Therefore, $\frac{1}{\epsilon_\nu} = O(1/\alpha)$.

Therefore, the query and time complexity in case Algorithm 2.5.3 is always correct is

$$\begin{aligned} Q(\delta, \gamma) &= O\left(\log \frac{1}{\alpha} \cdot \left(\frac{1}{\alpha} \log \frac{\log \frac{1}{\alpha}}{\alpha \gamma_1}\right)\right) \\ &= \tilde{O}\left(\frac{1}{\alpha} \log \frac{1}{\gamma_1}\right) \end{aligned} \quad (2.102)$$

Now, assume Algorithm 2.5.3 may give an incorrect answer. As long as $L_\nu \leq \epsilon_{mon}(f) \leq H_\nu$ the above analysis still holds. Assume this is not the case. First we observe that

$$\text{if } \nu_1 > \nu_2 \text{ then } L_{\nu_2} \leq L_{\nu_1} < H_{\nu_1} \leq H_{\nu_2} \quad (2.103)$$

Therefore, if for some ν' , $\epsilon_{mon}(f) \leq L_{\nu'+1}$ (i.e. ν' is the smallest such one), then for every $\nu > \nu'$,

$$\epsilon_{mon}(f) \leq L_\nu \leq H_\nu \quad (2.104)$$

and the above complexity analysis still holds.

The second case is that for some ν' , $\epsilon_{mon}(f) \geq H_{\nu'+1}$ (i.e. ν' is the smallest such one). Therefore, for every $\nu \leq \nu'$ we have $\epsilon_{mon}(f) \leq H_\nu$, and for every $\nu > \nu'$ we have $\epsilon_{mon}(f) \geq H_\nu$. Let ν'' be the smallest ν such that $\nu > \nu'$ and for which Algorithm 2.5.3 gives a correct answer. Since $\epsilon_{mon}(f) \geq H_{\nu''} \geq \epsilon_{\nu''}$, then

$$L_{\nu''+1} = \frac{2L_{\nu''} + H_{\nu''}}{3} \geq \frac{H_{\nu''}}{3} \quad \text{and} \quad H_{\nu''+1} = H_{\nu''} \quad (2.105)$$

Recall that for every $\nu' \leq \nu < \nu''$, Algorithm 2.5.3 gives a wrong answer. Therefore, for every $\nu' \leq \nu < \nu''$, we have that

$$H_{\nu+1} = \frac{L_\nu + 2H_\nu}{3} \geq \frac{2H_\nu}{3} \quad (2.106)$$

Therefore, for every $\nu \geq \nu'' + 1$, ϵ_ν satisfies

$$\begin{aligned} \epsilon_\nu &\geq L_\nu \geq L_{\nu''+1} \geq \frac{1}{3}H_{\nu''} \geq (1/3) \cdot \left(\frac{2}{3}\right)^{\nu''-\nu'} H_{\nu'} \geq (1/3) \cdot \left(\frac{2}{3}\right)^{\nu''-\nu'} \epsilon_{mon}(f) \\ \epsilon_\nu &= \frac{H_\nu + L_\nu}{2.25} \geq \frac{\delta}{2.25} \end{aligned} \quad (2.107)$$

That is, $\frac{1}{\epsilon_\nu} = O\left(\left(\frac{3}{2}\right)^{\nu''-\nu'} \frac{1}{\alpha}\right)$. Also, the stopping condition of Algorithm 2.5.4 implies that

$$\begin{aligned} H_\nu - L_\nu &> 0.5H_\nu > 0.5L_\nu \geq 0.5 \cdot (1/3) \cdot \left(\frac{2}{3}\right)^{\nu''-\nu'} \cdot \epsilon_{mon}(f) \\ H_\nu - L_\nu &\geq \delta \end{aligned} \tag{2.108}$$

That is, $\frac{1}{H_\nu - L_\nu} = O\left(\left(\frac{3}{2}\right)^{\nu''-\nu'} \frac{1}{\alpha}\right)$. Equations (2.107) and (2.108) imply that the query and time complexity in this case is at most

$$O\left(\left(\frac{3}{2}\right)^{\nu''-\nu'} \cdot \text{Poly}(\nu'' - \nu') \cdot Q(\delta, \gamma)\right) \tag{2.109}$$

The probability that for all ν such that $\nu' \leq \nu < \nu''$ Algorithm 2.5.3 indeed gives a wrong answer is at most

$$\left(\frac{\gamma 1}{2\nu'^2}\right)^{\nu''-\nu'} < (1/3)^{\nu''-\nu'} \tag{2.110}$$

Therefore, the average complexity is at most

$$Q(\delta, \gamma) + \sum_{n=0}^{\infty} \left(\frac{1}{3}\right)^n \cdot \left(\left(\frac{3}{2}\right)^n \cdot \text{Poly}(n) \cdot Q(\delta, \gamma)\right) = Q(\delta, \gamma) \tag{2.111}$$

■

2.5.3 Distance Approximation for $\Sigma^2 \mapsto \{0, 1\}$ Functions

2.5.3.1 Results

The result of this subsection is Algorithm 2.5.5 (see page 44). It is given query access to a function $f : \Sigma^2 \mapsto \{0, 1\}$ and parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$ and returns $\hat{\epsilon}$ such that

Lemma 2.5.7 *At the end of Algorithm 2.5.5, with probability at least $1 - \gamma$*

$$\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + 3\delta$$

The query and time complexity of Algorithm 2.5.5 is $O\left(\frac{1}{\delta^4} \lg\left(\frac{1}{\delta\gamma}\right)\right)$.

Unlike Algorithm 2.5.1 (see page 29), Algorithm 2.5.5 provides a distance approximation with no multiplicative factor. Also, it is more efficient than Algorithm 2.5.1.

Proof of Item 2 of Theorem 1.2.2: Given a function $f : \Sigma^d \mapsto \{0, 1\}$, Algorithm 2.4.1 (see Lemma 2.4.1 on page 24), combined with Algorithm 2.5.5 (see Lemma 2.5.7), provides a distance to monotonicity estimation $\hat{\epsilon}$ such that with high probability

$$\frac{1}{d}\epsilon_{mon}(f) - 3\delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + 3\delta \tag{2.112}$$

The query and time complexity of the algorithm is $O\left(\frac{1}{\delta^6} \log\left(\frac{1}{\delta}\right)\right)$. ■

2.5.3.2 δ -block Functions

Note that the functions in this section are always $\Sigma^2 \mapsto \{0, 1\}$ functions, even if we do not specifically state it.

Definition 2.5.2 Let $\delta \in (0, 1]$.

- For every $1 \leq i, j \leq \frac{1}{\delta}$ let

$$\text{block}(i, j) = \left\{ (i', j') \in \Sigma^2 : (i-1)\delta n + 1 \leq i' \leq i\delta n \quad \wedge \quad (j-1)\delta n + 1 \leq j' \leq j\delta n \right\}$$

- A δ -block function f is a function such that for every i_1, i_2, j_1, j_2 , if there exists a pair (i, j) such that $(i_1, j_1) \in \text{block}(i, j)$ and $(i_2, j_2) \in \text{block}(i, j)$, then $f(i_1, j_1) = f(i_2, j_2)$. That is, the labels of f are equal within every block.

We show that for every monotone function $f : \Sigma^2 \mapsto \{0, 1\}$ there exists a monotone δ -block function $g : \Sigma^2 \mapsto \{0, 1\}$ such that $\text{dist}(f, g) \leq 2\delta$. This implies that for every $f : \Sigma^2 \mapsto \{0, 1\}$ (not necessarily monotone) there exists a monotone δ -block function g such that $\epsilon_{\text{mon}}(f) \leq \text{dist}(f, g) \leq \epsilon_{\text{mon}}(f) + 2\delta$. We then show how to estimate $\text{dist}(f, g)$.

Let $f : \Sigma^2 \mapsto \{0, 1\}$ be some monotone function, and $g : \Sigma^2 \mapsto \{0, 1\}$ be defined as follows. For every $i, j \in \{1, \dots, \frac{1}{\delta}\}$

- If all labels in $\text{block}(i, j)$ of f are 0, then all labels in $\text{block}(i, j)$ of g are also 0.
- Else all labels in $\text{block}(i, j)$ of g are 1.

Figure 2.2 illustrates such f and g .

Lemma 2.5.8 Function g as defined above is a monotone δ -block function.

Proof: Obviously, g is a δ -block function. We need to show that it is also monotone. Consider some i_1, j_1 such that $g(i_1, j_1) = 1$, and some $i_2 \geq i_1$ and $j_2 \geq j_1$. There can be two cases for i_2, j_2 ,

- If (i_1, j_1) and (i_2, j_2) are in the same block, then obviously $g(i_2, j_2) = g(i_1, j_1) = 1$.
- Else, (i_2, j_2) is in a block (name it block_2) whose indexes are larger than the indexes of the block of (i_1, j_1) (name it block_1). This means that for every index in block_1 , there is an index in block_2 that is larger. Recall that by the definition of g , there must be an index in block_1 for which the label of f is 1. This means that there must be an index in block_2 for which the label of f is also 1. This means that the labels of g in block_2 are 1, and therefore $g(i_2, j_2) = 1$.

The lemma follows. ■

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
5	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
10	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
11	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Figure 2.2: An illustration of a monotone function $f : \Sigma^2 \mapsto \{0, 1\}$ and a suitable g . In this case, $\Sigma = \{1, \dots, 15\}$ and $\delta = 1/5$. The labels of f are written in black. For example, $f(5, 9) = 0$ and $f(5, 10) = 1$. The bold dotted line stresses the limit between the 0 label area and the 1 label area of f . The labels of g are 0 in the gray areas of the domain, and 1 in the white areas of the domain.

Lemma 2.5.9 *The functions f and g as defined above satisfy*

$$\text{dist}(f, g) \leq 2\delta$$

Proof: It is easy to observe that for every i, j , if all labels of f in $\text{block}(i, j)$ are 0, then all labels of g in $\text{block}(i, j)$ are 0, and if all labels of f in $\text{block}(i, j)$ are 1, then all labels of g in $\text{block}(i, j)$ are 1. Therefore, we need to bound the number of blocks in which f 's labels are not all 0 and not all 1.

Consider some $1 \leq i \leq \frac{1}{\delta}$. Let $0 \leq j_\ell(i) < j_h(i) \leq \frac{1}{\delta} + 1$ be such that

- For every $1 \leq j \leq j_\ell(i)$, the labels of $\text{block}(i, j)$ of f are all 0 (if $j_\ell(i) = 0$ then there is no such j).
- For every $1/\delta \geq j \geq j_h(i)$, the labels of $\text{block}(i, j)$ of f are all 1 (if $j_h(i) = 1/\delta + 1$ then there is no such j).
- For every $j_\ell(i) < j < j_h(i)$ the labels of $\text{block}(i, j)$ of f are not all 0 and not all 1 (if $j_\ell(i) = j_h(i) - 1$ then there is no such j).

Since f is monotone, then there is no loss of generality in the above notation. Also, the fact that f is monotone implies that for every $i < 1/\delta$ we have that $j_h(i + 1) \leq j_\ell(i) + 2$. Therefore, the number of blocks in which the labels of f are not all 0 and not all 1 is at most

$$\begin{aligned} \sum_{i=1}^{1/\delta} j_h(i) - j_\ell(i) - 1 &\leq \left(\sum_{i=1}^{1/\delta-1} j_h(i) - (j_h(i+1) - 2) - 1 \right) + j_h(1/\delta) - j_\ell(1/\delta) - 1 \\ &= j_h(1) - j_\ell(1/\delta) + (1/\delta - 2) \\ &\leq \frac{2}{\delta} \end{aligned}$$

Recall that the number of elements in every block is exactly $\delta^2 n^2$ and therefore $dist(f, g) \leq 2\delta$. ■

Lemma 2.5.10 *For every monotone function $f : \Sigma^2 \mapsto \{0, 1\}$ there exists a δ -block monotone function $g : \Sigma^2 \mapsto \{0, 1\}$ such that*

$$dist(f, g) \leq 2\delta$$

Proof: Immediate from Lemmas 2.5.8 and 2.5.9. ■

2.5.3.3 The Algorithm

Consider the dynamic programming procedure BestSw (see page 44). It is given a matrix A of size $\frac{1}{\delta^2}$. For every i, j cell (i, j) in A holds the number of 1 labels and 0 labels in $block(i, j)$ of f . It returns the distance between f and the δ -block monotone function that is closest to f .

It is based on the following recursive principle. Let

- $f^{i,j} : \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\} \mapsto \{0, 1\}$ be such that for every $x \in \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\}$, $f^{i,j}(x) = f(x)$. (where i represents rows, and j represents columns)
- $g^{i,j} : \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\} \mapsto \{0, 1\}$ be the closest δ -block monotone function to $f^{i,j}$.

Let $k = \frac{1}{\delta}$ and $i, j \in \{1, \dots, k\}$. For simplicity of notation, let $|f^{i,j}|$ be the size of the domain of $f^{i,j}$. Consider the following recursive equation (its correctness is explained right after):

$$\begin{aligned} dist(f^{i,j}, g^{i,j}) \cdot |f^{i,j}| = \\ \min \left\{ \begin{aligned} & dist(f^{i+1,j}, g^{i+1,j}) \cdot |f^{i+1,j}| + \sum_{q=1}^j A[i, q].ones, \\ & dist(f^{i,j-1}, g^{i,j-1}) \cdot |f^{i,j-1}| + \sum_{p=i}^k A[p, j].zeros \end{aligned} \right\} \quad (2.113) \end{aligned}$$

An explanation:

- If the labels of $g^{i,j}$ are 0 on $block(i, j)$ then
 - For every $q \leq j$, the labels of $g^{i,j}$ are 0 on $block(i, q)$. This implies that the number of different labels between $g^{i,j}$ and $f^{i,j}$ in $\bigcup_{q=1}^j block(i, q)$ is exactly $\sum_{q=1}^j A[i, q].ones$.
 - For every $p > i$ and $q \leq j$, the labels of $g^{i,j}$ on $block(p, q)$ may be 0 or 1. This implies that the labels of $g^{i,j}$ on these blocks (which are optimal for minimizing $dist(f^{i,j}, g^{i,j})$ s.t. g is δ -block and monotone), are equal to the labels of $g^{i+1,j}$ on these blocks.
- If the labels of $g^{i,j}$ are 1 on $block(i, j)$ then
 - For every $p \geq i$, the labels of $g^{i,j}$ are 1 on $block(p, j)$. This implies that the number of different labels between $g^{i,j}$ and $f^{i,j}$ in $\bigcup_{p=i}^k block(p, j)$ is exactly $\sum_{p=i}^k A[p, j].zeros$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															

Figure 2.3: Consider the case that $\Sigma = \{1, \dots, 15\}$ and $\delta = 1/5$. The domain of $f^{3,4}$ and $g^{3,4}$ is in gray. The domain of $f^{3,3}$ and $g^{3,3}$ is surrounded by a bold dotted line. The domain of $f^{4,4}$ and $g^{4,4}$ is surrounded by a bold line. By the time the algorithm calculates $\text{dist}(f^{3,4}, g^{3,4})$ it has already calculated $\text{dist}(f^{3,3}, g^{3,3})$ and $\text{dist}(f^{4,4}, g^{4,4})$.

- For every $p \geq i$ and $q < j$, the labels of $g^{i,j}$ on $\text{block}(p, q)$ may be 0 or 1. This implies that the labels of $g^{i,j}$ on these blocks (which are optimal for minimizing $\text{dist}(f^{i,j}, g^{i,j})$ s.t. g is δ -block and monotone), are equal to the labels of $g^{i,j-1}$ on these blocks.

We calculate this recursive equation in the following order: $f^{k,1}, f^{k,2}, \dots, f^{k,k}, f^{k-1,1}, \dots, f^{k-1,k}, \dots, f^{1,1}, \dots, f^{1,k}$. Observe that $f^{1,k} = f$. Figure 2.3 illustrates this process. The results of this computation are saved in matrix D through the algorithm, so that at the end of the algorithm, for every i, j , $D[i, j] = \text{dist}(f^{i,j}, g^{i,j}) \cdot |f^{i,j}|$.

Procedure BestSw(A). Finding the Closest δ -block Monotone Function

Let $k = \frac{1}{\delta}$. A is a $\{1, \dots, k\} \times \{1, \dots, k\}$ matrix such that

- $A[i, j].ones$ = the number of ones in $block(i, j)$ of f .
- $A[i, j].zeros$ = the number of zeros in $block(i, j)$ of f .
- Let O and Z be $\{1, \dots, k\} \times \{1, \dots, k\}$ matrices such that
 - For every i and j , $O[i, j] = \sum_{q=1}^j A[i, q].ones$
 - For every i and j , $Z[i, j] = \sum_{q=i}^k A[q, j].zeros$
- Let D be a $\{1, \dots, k, k+1\} \times \{0, 1, \dots, k\}$ matrix (rows \times columns), with initial value of 0 in every cell.
- for $i = k$ downto 1
 - for $j = 1$ to k
 - * $D[i, j] = \min\{D[i+1, j] + O[i, j], D[i, j-1] + Z[i, j]\}$
- Let $\hat{\epsilon} = D[1, k]/n^2$. Return $\hat{\epsilon}$.

Note that the time complexity of BestSw is $O(k^2) = O(\frac{1}{\delta^2})$. Now consider Algorithm 2.5.5:

Algorithm 2.5.5 *Distance Approximation for Two Dimensions and $\{0, 1\}$ Range Based on Dynamic Programming*

Given query access to a function $f : \Sigma^2 \mapsto \{0, 1\}$ and parameters $0 < \delta \leq 1$, $0 < \gamma \leq 1$

- Let $k = \frac{1}{\delta}$ and $m = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\delta^2 \gamma}))$
- for $i = 1$ to k
 - for $j = 1$ to k
 - * take m samples of $block(i, j)$ of f . Let $X_r^{i,j} = f(i', j')$, where i', j' are the coordinates of the r^{th} sample.
 - * $\tilde{A}[i, j].ones = \frac{1}{m} \cdot \delta^2 n^2 \cdot \sum_{r=1}^m X_r^{i,j}$.
 - * $\tilde{A}[i, j].zeros = \delta^2 n^2 - \tilde{A}[i, j].ones$
- Let $\hat{\epsilon} = BestSw(\tilde{A})$
- return $\hat{\epsilon}$

Proof of Lemma 2.5.7: Let p_{ij} be the probability to find a 1 label when selecting uniformly in $block(i, j)$ of f . Note that $p_{ij} \delta^2 n^2$ is the exact amount of 1 labels in $block(i, j)$. For every $i, j \in \{1, \dots, \frac{1}{\delta}\}$, by the additive chernoff bound, with probability at least $1 - \gamma \cdot \delta^2$,

$$\left| \frac{1}{m} \sum_{r=1}^m X_r^{i,j} - p_{ij} \right| \leq \delta \quad (2.114)$$

Therefore, by the union bound, with probability at least $1 - \gamma$, this is true for all such i, j and

so

$$\begin{aligned} \left| \sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} \tilde{A}[i, j].ones - \sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} p_{ij} \delta^2 n^2 \right| &\leq \sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} \left| \tilde{A}[i, j].ones - p_{ij} \delta^2 n^2 \right| \\ &\leq \delta n^2 \end{aligned} \quad (2.115)$$

which means that with probability at least $1 - \gamma$, $\sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} \tilde{A}[i, j].ones$ and $\sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} \tilde{A}[i, j].zeros$ are estimations of the number of 1 labels and 0 labels (respectively) in f , with total error smaller than δn^2 . Assume that this is indeed the case.

Let $\tilde{f} : \Sigma^2 \mapsto \{0, 1\}$ be the closest function to f such that for every $i, j \in \{1, \dots, 1/\delta\}$, the number of 1 labels in $block(i, j)$ of \tilde{f} is exactly $\tilde{A}[i, j].ones$, and the number of 0 labels is $\tilde{A}[i, j].zeros$. Therefore, $dist(f, \tilde{f}) \leq \delta$.

Let \tilde{g} be the δ -block monotone function such that $dist(\tilde{f}, \tilde{g}) = BestSw(\tilde{A}) = \hat{\epsilon}$. Therefore,

$$\begin{aligned} \epsilon_{mon}(f) &\leq dist(f, \tilde{g}) \\ &\leq dist(f, \tilde{f}) + dist(\tilde{f}, \tilde{g}) \\ &\leq \delta + dist(\tilde{f}, \tilde{g}) \\ &= \delta + \hat{\epsilon} \end{aligned} \quad (2.116)$$

Let \tilde{f}_{mon} be the monotone function that is closest to \tilde{f} . That is, $\epsilon_{mon}(\tilde{f}) = dist(\tilde{f}, \tilde{f}_{mon})$. Lemma 2.5.10 implies that there is a δ -block monotone function, name it \tilde{g}_{mon} , such that $dist(\tilde{f}_{mon}, \tilde{g}_{mon}) \leq 2\delta$. Recall that \tilde{g} is the closest δ -block monotone function to \tilde{f} . Therefore,

$$\begin{aligned} dist(\tilde{f}, \tilde{g}) &\leq dist(\tilde{f}, \tilde{g}_{mon}) \\ &\leq dist(\tilde{f}, \tilde{f}_{mon}) + dist(\tilde{f}_{mon}, \tilde{g}_{mon}) \\ &\leq \epsilon_{mon}(\tilde{f}) + 2\delta \end{aligned} \quad (2.117)$$

Recall that $dist(f, \tilde{f}) \leq \delta$. Therefore, $\epsilon_{mon}(\tilde{f}) \leq \delta + \epsilon_{mon}(f)$. And so,

$$\begin{aligned} \hat{\epsilon} &= dist(\tilde{f}, \tilde{g}) \\ &\leq \delta + \epsilon_{mon}(f) + 2\delta \end{aligned} \quad (2.118)$$

and finally, equations (2.116) and (2.118) imply that

$$\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f) + 3\delta \quad (2.119)$$

■

2.6 An Improved Testing Algorithm for $\Sigma^d \mapsto \{0, 1\}$ Functions

[GGL⁺00] suggest a testing algorithm for $\Sigma^d \mapsto \{0, 1\}$ functions, based on dimension reduction. [GGL⁺00, Theorem 14] implies that there is a testing algorithm for $\Sigma^d \mapsto \{0, 1\}$ functions with time and query complexity of $O\left(\frac{d \cdot \log n}{\epsilon}\right)$. We present here an improvement of that result for the case of small d and large n .

Let $A(g, \epsilon)$ be a 1-sided error testing algorithm for monotonicity of $g : \Sigma \mapsto \{0, 1\}$. The time and query complexity of $A(g, \epsilon)$ is $T(\epsilon)$. We present here, based on Lemma 2.3.1 (see page 18) and algorithm A , a testing algorithm (Algorithm 2.6.1) for monotonicity of functions $\Sigma^d \mapsto \{0, 1\}$. The time and query complexity of Algorithm 2.6.1 is

$$O\left(\frac{\log \frac{d}{\epsilon} \cdot d}{\epsilon} \cdot \sum_{j=0}^{\log \frac{d}{\epsilon}} \frac{T(2^{-(j+1)})}{2^j}\right) \quad (2.120)$$

Later we describe a simple algorithm A with query complexity of $O\left(\frac{1}{\epsilon}\right)$ and time complexity of $O\left(\frac{1}{\epsilon} \log \frac{1}{\epsilon}\right)$ (see Algorithm 2.6.2). This enables us to prove Theorem 1.2.3.

Proof of Theorem 1.2.3: Lemmas 2.6.1 and 2.6.2 imply that Algorithm 2.6.1 combined with Algorithm 2.6.2 provides a 1-sided error testing algorithm for monotonicity of $\Sigma^d \mapsto \{0, 1\}$ functions. The query complexity of the algorithm is $O\left(\frac{d \log^2 \frac{d}{\epsilon}}{\epsilon}\right)$. The time complexity of the algorithm is $O\left(\frac{d \log^3 \frac{d}{\epsilon}}{\epsilon}\right)$. ■

Algorithm 2.6.1 *Testing Monotonicity for d -dimensions and $\{0, 1\}$ Range Using a Dimension Reduction*

Given query access to a function $f : \Sigma^d \mapsto \{0, 1\}$ and a parameter ϵ :
for $j = 0$ to $\ell = \lfloor \log \frac{4d}{\epsilon} \rfloor$:

- Repeat the following $\Theta\left(\frac{\ell \cdot d}{\epsilon \cdot 2^j}\right)$ times:
 - Randomly and uniformly select $i \in \{1, \dots, d\}, \alpha \in \Sigma^{i-1}, \beta \in \Sigma^{d-i}$.
 - if $A(f_{i, \alpha, \beta}, 2^{-(j+1)})$ rejects, then reject.

If the algorithm ends without rejecting then accept.

Lemma 2.6.1 *Algorithm 2.6.1 is a 1-sided error testing algorithm for monotonicity of $\Sigma^d \mapsto \{0, 1\}$ functions.*

Proof: Since A is a 1-sided error tester, it is obvious that if $\epsilon_{mon}(f) = 0$ then Algorithm 2.6.1 accepts. Assume $\epsilon < \epsilon_{mon}(f)$. For every $j \geq 0$ let

$$B_j = \{(i, \alpha, \beta) : 2^{-(j+1)} < \epsilon_{mon}(f_{i, \alpha, \beta}) \leq 2^{-j}\} \quad (2.121)$$

Therefore,

$$\begin{aligned} \epsilon &< \epsilon_{mon}(f) \\ &\leq 2d \cdot E_{i, \alpha, \beta}[\epsilon_{mon}(f_{i, \alpha, \beta})] \\ &= \frac{2}{n^{d-1}} \sum_{i, \alpha, \beta} \epsilon_{mon}(f_{i, \alpha, \beta}) \\ &\leq \frac{2}{n^{d-1}} \sum_{j=0}^{\infty} |B_j| \cdot 2^{-j} \end{aligned} \quad (2.122)$$

$$\begin{aligned}
&= \frac{2}{n^{d-1}} \sum_{j=0}^{\lfloor \log \frac{4d}{\epsilon} \rfloor} |B_j| \cdot 2^{-j} + \frac{2}{n^{d-1}} \sum_{j=\lceil \log \frac{4d}{\epsilon} \rceil}^{\infty} |B_j| \cdot 2^{-j} \\
&\leq \frac{2}{n^{d-1}} \sum_{j=0}^{\lfloor \log \frac{4d}{\epsilon} \rfloor} |B_j| \cdot 2^{-j} + \frac{2}{n^{d-1}} \cdot d \cdot n^{d-1} \cdot 2^{\log \frac{\epsilon}{4d}} \\
&= \frac{2}{n^{d-1}} \sum_{j=0}^{\lfloor \log \frac{4d}{\epsilon} \rfloor} |B_j| \cdot 2^{-j} + \frac{\epsilon}{2}
\end{aligned}$$

Inequality (2.122) is a result of Lemma 2.3.1. Therefore

$$\frac{\epsilon \cdot n^{d-1}}{4} < \sum_{j=0}^{\lfloor \log \frac{4d}{\epsilon} \rfloor} |B_j| \cdot 2^{-j} \quad (2.123)$$

Recall that $\ell = \lfloor \log \frac{4d}{\epsilon} \rfloor$. Therefore, there exists $j \leq \ell$ such that

$$|B_j| \cdot 2^{-j} \geq \frac{\epsilon \cdot n^{d-1}}{4 \cdot \ell} \quad (2.124)$$

hence

$$\frac{|B_j|}{d \cdot n^{d-1}} \geq \frac{\epsilon \cdot 2^j}{4 \cdot \ell \cdot d} \quad (2.125)$$

Which means that by randomly, uniformly, and independently selecting $\Theta(\frac{\ell \cdot d}{\epsilon \cdot 2^j})$ triplets (i, α, β) , with high probability at least one of the selected triplets is in B_j . Which means that for this triplet

$$\epsilon_{\text{mon}}(f_{i,\alpha,\beta}) > 2^{-(j+1)} \quad (2.126)$$

and $A(f_{i,\alpha,\beta}, 2^{-(j+1)})$ rejects with high probability. ■

We now give a very simple testing algorithm for monotonicity of $\Sigma \mapsto \{0, 1\}$ functions.

Algorithm 2.6.2 *Testing Monotonicity for $\Sigma \mapsto \{0, 1\}$ Functions*

Given query access to a function $f : \Sigma \mapsto \{0, 1\}$ and a parameter $0 < \epsilon \leq 1$:

- Let $M = \{x^i\}_{i=1}^m$ be a set of $m = \Theta(\frac{1}{\epsilon})$ uniformly independently selected elements.
- Sort M (requires $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ time).
- If all $i, j \in [m]$ satisfy

$$\text{if } x^i < x^j \text{ then } f(x^i) \leq f(x^j)$$

then accept. Else reject. Observe that since M is sorted then checking this condition can be done in $O(\frac{1}{\epsilon})$ time.

The query complexity of Algorithm 2.6.2 is $O(\frac{1}{\epsilon})$. The time complexity of Algorithm 2.6.2 is $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$.

Lemma 2.6.2 *Algorithm 2.6.2 is a 1-sided error testing algorithm for monotonicity of $\Sigma \mapsto \{0, 1\}$ functions.*

Proof: It is obvious that if f is monotone than Algorithm 2.6.2 accepts. Therefore, assume that $\epsilon_{mon}(f) > \epsilon$. Obviously $S_1[f]$ is monotone, and so $dist(f, S_1[f]) > \epsilon$. Let $z(f)$ denote the number of 0 labels in f (and in $S_1[f]$). Observe that

$$\begin{aligned} |\{x \leq z(f) : f(x) = 1\}| &= |\{x > z(f) : f(x) = 0\}| \\ &= \frac{dist(f, S_1[f]) \cdot n}{2} \\ &> \frac{\epsilon \cdot n}{2} \end{aligned} \tag{2.127}$$

Also observe that if $x \in \{x \leq z(f) : f(x) = 1\}$ and $y \in \{x > z(f) : f(x) = 0\}$ then (x, y) is an edge in $G_{viol}(f)$. Therefore, the probability that there are no violations of monotonicity in M is at most the probability that $M \cap \{x \leq z(f) : f(x) = 1\} = \emptyset$ or $M \cap \{x > z(f) : f(x) = 0\} = \emptyset$, which by Equation (2.127) is upper bounded by $2(1 - \epsilon/2)^m < 1/3$. The lemma follows. ■

2.7 Directions for Further Research

One approach for estimating ϵ_{mon} of $\Sigma^d \mapsto \Xi$ functions is the one implied by Lemmas 2.2.1 and 2.3.1 and presented in Section 2.4. That is, to estimate the average of the distance to monotonicity of the induced k -dimensional cubes of f . Lemma 2.7.1 implies that the bounds presented in Lemmas 2.2.1 and 2.3.1 are tight (in terms of d/k) which means that this approach cannot achieve a tighter estimation than the one we present in Section 2.4. That is, better than a $(\Theta(\frac{d}{k}), \delta)$ -estimate.

Lemma 2.7.1 *Consider any $k \in [d]$ such that $d/k \in \mathcal{Z}$. For every $\epsilon_m < 1/2$ (such that $\epsilon_m n$ is an integer) there exist functions $f^u : \Sigma^d \mapsto \{0, 1\}$ and $f^\ell : \Sigma^d \mapsto \{0, 1\}$ such that*

$$\begin{aligned} \epsilon_{mon}(f^u) &= \frac{d}{k} E_{q,\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^u)] = \epsilon_m \\ \epsilon_{mon}(f^\ell) &\sim E_{q,\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^\ell)] \sim \epsilon_m/2 \end{aligned}$$

Where $A \sim B$ means $\lim_{n \rightarrow \infty} A = \lim_{n \rightarrow \infty} B$.

Proof: We begin with f^u . Let it be defined as follows. For every $x = (x_\ell x_h) \in \Sigma^d$ such that $x_\ell \in \Sigma$ and $x_h \in \Sigma^{d-1}$, if $x_\ell \leq \epsilon_m \cdot n$ then $f^u(x) = 1$, else $f^u(x) = 0$. Figure 2.4 illustrates f^u .

Observe the following.

- $\epsilon_{mon}(f^u) = \epsilon_m$ (By modifying all 1 labels to 0 labels, f becomes monotone. Also, there is a matching of size $\epsilon_m n^d$ in $G_{viol}(f^u)$).
- Every $1 < q \leq d/k$ and $\alpha \in \Sigma^{(q-1)k}$, $\beta \in \Sigma^{d-qk}$ satisfy $\epsilon_{mon}(f_{q,\alpha,\beta}^u) = 0$ (The labels of $f_{q,\alpha,\beta}^u$ are either all 0 or all 1).

	1	2	3	4	5	6	7	8	9
1	1	1	1	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0
5	1	1	1	0	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0
8	1	1	1	0	0	0	0	0	0
9	1	1	1	0	0	0	0	0	0

Figure 2.4: An illustration of f^u for the case of $d = 2$, $\Sigma = \{1, \dots, 9\}$ and $\epsilon_m = 1/3$.

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	1	1	1	1
2	1	0	1	1	1	1	1	1	1
3	0	1	0	1	1	1	1	1	1
4	1	0	1	1	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1
6	1	0	1	1	1	1	1	1	1
7	0	1	0	1	1	1	1	1	1
8	1	0	1	1	1	1	1	1	1
9	0	1	0	1	1	1	1	1	1

Figure 2.5: An illustration of f^ℓ for the case of $d = 2$, $\Sigma = \{1, \dots, 9\}$ and $\epsilon_m = 1/3$.

- For the case of $q = 1$, every $\alpha \in \Sigma^0$ (i.e., α is an empty string) and every $\beta \in \Sigma^{d-k}$ satisfy $\epsilon_{mon}(f_{1,\alpha,\beta}^u) = \epsilon_m$.

Therefore,

$$\begin{aligned}
E_{q,\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^u)] &= \frac{k}{d} \left(E_{\alpha,\beta}[\epsilon_{mon}(f_{1,\alpha,\beta}^u)] + \sum_{q=2}^{d/k} E_{\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^u)] \right) \\
&= \frac{k}{d} (\epsilon_m + (d/k - 1) \cdot 0) \\
&= \frac{k}{d} \epsilon_{mon}(f^u)
\end{aligned}$$

We now turn to f^ℓ . Let it be defined as follows. Let $par : \Sigma^d \mapsto \{0, 1\}$ denote the parity function. That is, let $S(x)$ denote the the sum of elements in $x \in \Sigma^d$, if $S(x)$ is odd then $par(x) = 1$, else $par(x) = 0$. For every $x = (x_\ell x_h) \in \Sigma^d$ such that $x_\ell \in \Sigma$ and $x_h \in \Sigma^{d-1}$, if $x_\ell \leq \epsilon_m \cdot n$ then $f^\ell(x) = par(x)$, else $f^\ell(x) = 1$. Figure 2.5 illustrates f^ℓ .

Observe the following.

- $\epsilon_{mon}(f^\ell) \sim \epsilon_m/2$.
- For every $1 < q \leq d/k$ consider any $\alpha = (\alpha_\ell \alpha_h) \in \Sigma^{(q-1)k}$ such that $\alpha_\ell \in \Sigma$ and $\alpha_h \in \Sigma^{(q-1)k-1}$, and any $\beta \in \Sigma^{d-qk}$.

- If $\alpha_\ell \leq \epsilon_m \cdot n$ then $\epsilon_{mon}(f_{q,\alpha,\beta}^\ell) \sim 0.5$.
- If $\alpha_\ell > \epsilon_m \cdot n$ then $\epsilon_{mon}(f_{q,\alpha,\beta}^\ell) = 0$.
- For the case of $q = 1$, every $\alpha \in \Sigma^0$, $\beta \in \Sigma^{d-k}$ satisfy $\epsilon_{mon}(f_{1,\alpha,\beta}^\ell) \sim \epsilon_m/2$.

Therefore,

$$\begin{aligned}
E_{q,\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^\ell)] &= \frac{k}{d} \left(E_{\alpha,\beta}[\epsilon_{mon}(f_{1,\alpha,\beta}^\ell)] + \sum_{q=2}^{d/k} E_{\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^\ell)] \right) \\
&\sim \frac{k}{d} \left(\epsilon_m/2 + (d/k - 1) \cdot \left(\frac{\epsilon_m n^{d-k} \cdot 0.5 + (1 - \epsilon_m) n^{d-k} \cdot 0}{n^{d-k}} \right) \right) \\
&= \epsilon_m/2 \\
&\sim \epsilon_{mon}(f^\ell)
\end{aligned}$$

The lemma follows. ■

This of course does not mean that a tighter estimation can't be obtained.

- For the case of $\Sigma^d \mapsto \{0, 1\}$ functions we present Algorithm 2.5.1 which is a $(2, \delta)$ -estimate algorithm with time and query complexity of the form $\left(\frac{d \log d}{\delta}\right)^d$. Does there exist a $(q(d), \delta)$ -estimate algorithm (where $q(d) = o(d)$) for $\Sigma^d \mapsto \{0, 1\}$ functions (such as Algorithm 2.5.1) with query and time complexity that is better than exponential in d ?
- For the case of $d = 2$ we present a $(1, \delta)$ -estimate algorithm based on dynamic programming (Algorithm 2.5.5). We failed to extend this algorithm to higher dimension, even in time complexity that is exponential in d .
- A similar question is asked regarding $\Sigma^d \mapsto \Xi$ functions (where $|\Xi| > 2$). Distance approximation algorithms for the case of $d = 1$ were presented by [PRR04] and [ACCL04]. Combined with our dimension reduction (Algorithm 2.4.1) they provide (efficiently) an estimation that is no better than $(\Theta(d \log |\Xi|), \delta)$ -estimate. Is it possible to achieve a tighter estimation?

Chapter 3

Distance Approximation to Convexity

In this chapter we prove Theorem 1.2.4 (see page 8).

3.1 On the Relation between Monotonicity and Convexity

Lemma 3.1.1 is quoted from [PRR03, Claim 1].

Lemma 3.1.1 *A function $f : \{1, \dots, n\} \mapsto \mathfrak{R}$ is convex if and only if every $2 \leq i \leq n - 1$ satisfies*

$$f(i) - f(i - 1) \leq f(i + 1) - f(i) \tag{3.1}$$

Let $g : \{2, \dots, n\} \mapsto \mathfrak{R}$ be such that for every i , $g(i) = f(i) - f(i - 1)$. Lemma 3.1.1 implies that g is monotone if and only if f is convex. In other words, $\epsilon_{con}(f) = 0$ if and only if $\epsilon_{mon}(g) = 0$. Therefore an intuitive question is raised: Is there some tight relation between $\epsilon_{mon}(g)$ and $\epsilon_{con}(f)$? If there was, then the problem of Distance Approximation to Convexity over domain $\{1, \dots, n\}$ could be easily reduced to the problem of Distance Approximation to Monotonicity, for which there already are some solutions. Unfortunately, there is no such relation. For example, let $f : \{1, \dots, n\} \mapsto \mathfrak{R}$ be defined as follows.

- For every $1 \leq i \leq 3n/4$ let $f(i) = 1$.
- For every $3n/4 < i \leq n$ let $f(i) = 0$.

Observe that $\epsilon_{con}(f) = 0.25$. Also observe that

- For every $2 \leq i \leq 3n/4$, $g(i) = 0$.
- $g(3n/4 + 1) = -1$
- For every $3n/4 < i \leq n - 1$, $g(i) = 0$.

That is, $\epsilon_{mon}(g) = \frac{1}{n-1}$.

Although there is no such simple relation, the methods of [ACCL04] for distance approximation to monotonicity can be extended and provide distance approximation to convexity, as we do in this chapter.

3.2 Preliminaries

- Let X be any discrete domain. Let $\epsilon_{con}(g)$ denote the (relative) distance of $g : X \mapsto \mathfrak{R}$ from the class of convex functions, which is the minimum of $dist(g, \bar{g}) = |\{x \in X : g(x) \neq \bar{g}(x)\}|/|X|$ over all convex functions $\bar{g} : X \mapsto \mathfrak{R}$.
- Let $\Sigma = \{1, 2, \dots, n\}$, and for every $d \in \mathcal{Z}$ let $[d] = \{1, 2, \dots, d\}$.
- Let $f : \Sigma \mapsto \mathfrak{R}$.
- Let $X \subseteq \Sigma$ be some set. Let $f_{\bar{X}} : \Sigma \setminus X \mapsto \mathfrak{R}$ be such that for every $k \in \Sigma \setminus X$, $f_{\bar{X}}(k) = f(k)$.
- For every $1 \leq i \leq n-1$, let $C(i) = \langle i, i+1 \rangle$ (i.e., $C(i)$ is an ordered pair).
- Given a set of pairs X , let $P(X) = \{k \in \Sigma \mid C(k) \in X \vee C(k-1) \in X\}$. Observe that $|P(X)| \leq 2 \cdot |X|$.

Definition 3.2.1 For every $1 \leq i \leq j \leq n-1$, $C(i)$ and $C(j)$ are co-convex if they satisfy the following

- If $j \leq i+1$ then

$$f(i+1) - f(i) \leq f(j+1) - f(j) \quad (3.2)$$

- If $j > i+1$ we also ask that

$$f(i+1) - f(i) \leq \frac{f(j) - f(i+1)}{j-i-1} \quad \text{and} \quad \frac{f(j) - f(i+1)}{j-i-1} \leq f(j+1) - f(j) \quad (3.3)$$

We also denote $C(i) \smile C(j)$ if $C(i)$ and $C(j)$ are co-convex, and $C(i) \not\smile C(j)$ otherwise. Observe that $C(i) \smile C(j)$ is equivalent to $C(j) \smile C(i)$.

Definition 3.2.2 Given $0 < \beta < 1/2$ and $i \in \Sigma$, $C(i)$ is β -big if there exists $j > i$ such that

$$\left| \left\{ k : i < k \leq j \ \& \ C(i) \not\smile C(k) \right\} \right| \geq (1/2 - \beta)(j - i + 1) \quad (3.4)$$

or, similarly, $j < i$ such that

$$\left| \left\{ k : j \leq k < i \ \& \ C(k) \not\smile C(i) \right\} \right| \geq (1/2 - \beta)(i - j + 1) \quad (3.5)$$

We say that j is a witness to $C(i)$'s β -bigness, and that $C(i)$ is β -big with respect to j . Let $B_\beta(f)$ denote the set of β -big pairs in f .

The additive and multiplicative chernoff bounds which we often use can be found in Appendix A.

3.3 Pairs and the Co-convexity Property

In this section we show that the co-convexity property is a transitive property, and show a relation between the co-convexity of pairs and the convexity of the function.

Our discussion is not focused only on domains of the form $\{1, \dots, n\}$, and we need to generalize Lemma 3.1.1 (see page 51) to the following.

Lemma 3.3.1 *Let $X \subset \mathcal{Z}$ be a finite set. A function $g : X \mapsto \mathfrak{R}$ is convex if and only if all $i, j, \ell \in X$ such that $i < j < \ell$ satisfy*

$$\frac{g(j) - g(i)}{j - i} \leq \frac{g(\ell) - g(j)}{\ell - j}$$

Proof: If g is convex then let $\alpha = \frac{\ell - j}{\ell - i}$ and observe that $\alpha i + (1 - \alpha)\ell = j$. Therefore, $g(j) \leq \alpha g(i) + (1 - \alpha)g(\ell)$. This implies that

$$(\ell - i)g(j) \leq (\ell - j)g(i) + (j - i)g(\ell) \quad (3.6)$$

and so

$$(\ell - j)g(j) + (j - i)g(j) \leq (\ell - j)g(i) + (j - i)g(\ell) \quad (3.7)$$

and

$$\frac{g(j) - g(i)}{j - i} \leq \frac{g(\ell) - g(j)}{\ell - j} \quad (3.8)$$

For the second direction, assume that there exists α and $i < \ell$ such that $\alpha i + (1 - \alpha)\ell \in X$ but $g(\alpha i + (1 - \alpha)\ell) > \alpha g(i) + (1 - \alpha)g(\ell)$. Denote $j = \alpha i + (1 - \alpha)\ell$ and observe that $\alpha = \frac{\ell - j}{\ell - i}$. This implies that

$$(\ell - i)g(j) > (\ell - j)g(i) + (j - i)g(\ell) \quad (3.9)$$

and so

$$(\ell - j)g(j) + (j - i)g(j) > (\ell - j)g(i) + (j - i)g(\ell) \quad (3.10)$$

and

$$\frac{g(j) - g(i)}{j - i} > \frac{g(\ell) - g(j)}{\ell - j} \quad (3.11)$$

■

Lemma 3.3.2 *If $Q \subseteq \Sigma$ is a set such that $f_{\overline{Q}}$ is convex, then there exists a convex function $g : \Sigma \mapsto \mathfrak{R}$ that is equal to f for every $x \in \Sigma \setminus Q$.*

Observe that Lemma 3.3.2 implies that if $f_{\overline{Q}}$ is convex, then $\epsilon_{con}(f) \leq \frac{|Q|}{n}$.

Proof of Lemma 3.3.2: We assume here that $|\Sigma \setminus Q| \geq 2$. Otherwise the proof is trivial. Let $g : \Sigma \mapsto \mathfrak{R}$ be defined as follows, and later we show that it is convex.

- For every $k \in \Sigma \setminus Q$, $g(k) = f_{\overline{Q}}(k) = f(k)$.
- For other elements (i.e., elements of Q), let g be defined as follows:
 - Consider all $k \in Q$ such that there exist elements in $\Sigma \setminus Q$ that are smaller than k , and elements in $\Sigma \setminus Q$ that are larger than k . Let i be the largest element in $\Sigma \setminus Q$ that is smaller than k , and j be the smallest element in $\Sigma \setminus Q$ that is larger than k . Let $g(k) = g(i) + \frac{g(j)-g(i)}{j-i} \cdot (k-i)$.
 - If there are $k \in Q$ such that there is no $i \in \Sigma \setminus Q$ such that $i < k$, then let j be the smallest element in $\Sigma \setminus Q$ that is larger than those k 's. That is, $1 \leq k < j$. Observe that since $|\Sigma \setminus Q| \geq 2$, then there must be such j , that $j \leq n-1$, and that $g(j+1)$ is already defined. Let $g(1)$ be large enough so that $\frac{g(j)-g(1)}{j-1} \leq g(j+1) - g(j)$, and let $g(k) = g(1) + \frac{g(j)-g(1)}{j-1} \cdot (k-1)$ for every such $1 < k < j$.
 - If there are $k \in Q$ such that there is no $j \in \Sigma \setminus Q$ such that $j > k$, then let i be the largest element in $\Sigma \setminus Q$ that is smaller than those k 's. That is, $i < k \leq n$. Observe that since $|\Sigma \setminus Q| \geq 2$, then there must be such i , that $i \geq 2$, and that $g(i-1)$ is already defined. Let $g(n)$ be large enough so that $\frac{g(n)-g(i)}{n-i} \geq g(i) - g(i-1)$, and let $g(k) = g(i) + \frac{g(n)-g(i)}{n-i} \cdot (k-i)$ for every such $i < k < n$.

We need to show that g is a convex function. Recall that $f_{\overline{Q}}$ is convex. Therefore, Lemma 3.3.1 implies that for every $i, j, \ell \in \Sigma \setminus Q$ such that $i < j < \ell$,

$$\frac{f_{\overline{Q}}(j) - f_{\overline{Q}}(i)}{j-i} \leq \frac{f_{\overline{Q}}(\ell) - f_{\overline{Q}}(j)}{\ell-j} \quad (3.12)$$

Using this fact, a technical review over g would show that every $2 \leq i \leq n-1$ satisfies

$$g(i) - g(i-1) \leq g(i+1) - g(i) \quad (3.13)$$

and Lemma 3.1.1 (see page 51) implies that g is convex. ■

Lemma 3.3.3 shows the transitivity of the co-convexity property.

Lemma 3.3.3 *For every $1 \leq i < k < j \leq n-1$, if $C(i) \smile C(k)$, and $C(k) \smile C(j)$, then $C(i) \smile C(j)$.*

Proof: We assume that $k > i+1$ and $j > k+1$ (the case of $k = i+1$ or $j = k+1$ is easier). Assume that $C(i) \smile C(k)$ and $C(k) \smile C(j)$. By the definition we have

$$f(i+1) - f(i) \leq \frac{f(k) - f(i+1)}{k-i-1} \leq f(k+1) - f(k) \leq \frac{f(j) - f(k+1)}{j-k-1} \leq f(j+1) - f(j)$$

Therefore,

$$\begin{aligned} & \frac{f(j) - f(i+1)}{j-i-1} \\ &= \frac{\left((j-k-1) \frac{f(j)-f(k+1)}{j-k-1} \right) + \left(f(k+1) - f(k) \right) + \left((k-i-1) \frac{f(k)-f(i+1)}{k-i-1} \right)}{j-i-1} \\ &\geq \frac{\left((j-k-1)(f(i+1) - f(i)) \right) + \left(f(i+1) - f(i) \right) + \left((k-i-1)(f(i+1) - f(i)) \right)}{j-i-1} \\ &= f(i+1) - f(i) \end{aligned} \quad (3.14)$$

and

$$\begin{aligned}
& \frac{f(j) - f(i+1)}{j - i - 1} \\
&= \frac{\left((j - k - 1) \frac{f(j) - f(k+1)}{j - k - 1} \right) + \left(f(k+1) - f(k) \right) + \left((k - i - 1) \frac{f(k) - f(i+1)}{k - i - 1} \right)}{j - i - 1} \\
&\leq \frac{\left((j - k - 1)(f(j+1) - f(j)) \right) + \left(f(j+1) - f(j) \right) + \left((k - i - 1)(f(j+1) - f(j)) \right)}{j - i - 1} \\
&= f(j+1) - f(j)
\end{aligned} \tag{3.15}$$

and the lemma follows. ■

Lemma 3.3.4 *Let X be a set of pairs. If every i, j such that $C(i) \notin X$ and $C(j) \notin X$ satisfy $C(i) \sim C(j)$ then $f_{\overline{P(X)}}$ is a convex function.*

Proof: Assume by contradiction that $f_{\overline{P(X)}}$ is not a convex function. We show here that there exist i, j such that $C(i) \notin X$ and $C(j) \notin X$ and $C(i)$ and $C(j)$ are not co-convex.

Assuming $f_{\overline{P(X)}}$ is not convex, Lemma 3.3.1 implies that there exist $i < j < \ell$ such that $i, j, \ell \notin P(X)$ and

$$\frac{f_{\overline{P(X)}}(j) - f_{\overline{P(X)}}(i)}{j - i} > \frac{f_{\overline{P(X)}}(\ell) - f_{\overline{P(X)}}(j)}{\ell - j} \tag{3.16}$$

Therefore also

$$\frac{f(j) - f(i)}{j - i} > \frac{f(\ell) - f(j)}{\ell - j} \tag{3.17}$$

Since $i, j, \ell \notin P(X)$ then also $C(i), C(j), C(\ell - 1) \notin X$. Consider the following cases:

- The case that $j = \ell - 1$, so that $C(j) = C(\ell - 1)$. Equation (3.17) implies that

$$\frac{f(j) - f(i)}{j - i} > f(j+1) - f(j) \tag{3.18}$$

If $C(i) \sim C(j)$ (We assume that $j > i + 1$. The case that $j = i + 1$ is easier.) then

$$f(i+1) - f(i) \leq \frac{f(j) - f(i+1)}{j - i - 1} \leq f(j+1) - f(j) \tag{3.19}$$

Which implies that

$$\begin{aligned}
\frac{f(j) - f(i)}{j - i} &= \frac{f(i+1) - f(i) + \left((j - i - 1) \frac{f(j) - f(i+1)}{j - i - 1} \right)}{j - i} \\
&\leq \frac{\frac{f(j) - f(i+1)}{j - i - 1} + \left((j - i - 1) \frac{f(j) - f(i+1)}{j - i - 1} \right)}{j - i} \\
&= \frac{f(j) - f(i+1)}{j - i - 1} \\
&\leq f(j+1) - f(j)
\end{aligned} \tag{3.20}$$

Which is a contradiction to Equation (3.18). Therefore $C(i)$ and $C(j)$ are not co-convex.

- Now we may assume that $j < \ell - 1$. We show here that it is not possible that both $C(i) \smile C(j)$ and $C(j) \smile C(\ell - 1)$. Assume by contradiction that it is possible. The fact that $C(i) \smile C(j)$ and Equation (3.20) imply that

$$\frac{f(j) - f(i)}{j - i} \leq f(j + 1) - f(j) \quad (3.21)$$

The fact that $C(j) \smile C(\ell - 1)$ (We assume that $\ell - 1 > j + 1$. The case that $\ell - 1 = j + 1$ is easier.) implies that

$$f(j + 1) - f(j) \leq \frac{f(\ell - 1) - f(j + 1)}{\ell - j - 2} \leq f(\ell) - f(\ell - 1) \quad (3.22)$$

Which implies that

$$\begin{aligned} \frac{f(\ell) - f(j)}{\ell - j} &= \frac{(f(j + 1) - f(j)) + ((\ell - j - 2)\frac{f(\ell - 1) - f(j + 1)}{\ell - j - 2}) + (f(\ell) - f(\ell - 1))}{\ell - j} \\ &\geq \frac{(f(j + 1) - f(j)) + ((\ell - j - 2)(f(j + 1) - f(j))) + (f(j + 1) - f(j))}{\ell - j} \\ &= f(j + 1) - f(j) \end{aligned} \quad (3.23)$$

Equations (3.21) and (3.23) contradict Equation (3.17). Therefore it is not possible that $C(i) \smile C(j)$ and $C(j) \smile C(\ell - 1)$.

The lemma follows. ■

Lemma 3.3.5 *If $f : \Sigma \mapsto \mathfrak{R}$ is a convex function then for every $1 \leq i < j \leq n - 1$, $C(i)$ and $C(j)$ are co-convex.*

Proof: The lemma follows directly from Lemma 3.3.1. ■

3.4 β -bigness - An Indication to the Distance to Convexity

The result of this section is presented in Lemma, 3.4.1. The proof of Lemma 3.4.1 builds on [ACCL04, Lemma 3].

Lemma 3.4.1 *Let $f : \Sigma \mapsto \mathfrak{R}$.*

1. $\frac{\epsilon_{con}(f)}{2}n \leq |B_0(f)|$
2. $|B_\beta(f)| \leq 12(1 + \frac{2\beta}{1-2\beta})\epsilon_{con}(f)n$

Proof:

Item 1. Let S be a set of minimum size of pairs such that $f_{\overline{P(S)}}$ is convex. Lemma 3.3.2 implies that $\epsilon_{con}(f)n \leq |P(S)|$. Consider some $i, j \in \Sigma$ such that $C(i)$ and $C(j)$ are not co-convex (if there are no such i, j then Lemma 3.3.4 implies that f is a convex function and $\frac{\epsilon_{con}(f)}{2}n = |B_0(f)| = 0$). Without loss of generality suppose that $i < j$. Lemma 3.3.3 implies that for every k such that $i < k \leq j$, either $C(i) \not\prec C(k)$ or $C(k) \not\prec C(j)$.

Therefore, by the definition of 0-big, at least $C(i)$ or $C(j)$ or both are 0-big. This implies that the set of 0-big pairs in f satisfies the property that if for some i and j , $C(i)$ and $C(j)$ are both not 0-big, then $C(i) \sim C(j)$. Lemma 3.3.4 implies that $f_{\overline{B_0(f)}}$ is a convex function. Therefore $|S| \leq |B_0(f)|$ and all together: $\epsilon_{con}(f)n \leq |P(S)| \leq 2 \cdot |S| \leq 2 \cdot |B_0(f)|$.

Item 2. For each β -big pair $C(i)$, we choose a unique witness j_i to its β -bigness. If $j_i > i$ then $C(i)$ is called right-big. If $j_i < i$ then $C(i)$ is called left-big. That is, if $C(i)$ is right-big then

$$\left| \left\{ k : i < k \leq j_i \ \& \ C(i) \not\prec C(k) \right\} \right| \geq (1/2 - \beta)(j_i - i + 1) \quad (3.24)$$

and if $C(i)$ is left-big then

$$\left| \left\{ k : j_i \leq k < i \ \& \ C(k) \not\prec C(i) \right\} \right| \geq (1/2 - \beta)(i - j_i + 1) \quad (3.25)$$

Let RB and LB denote the set of right-big pairs and left-big pairs respectively. Observe that RB and LB are disjoint sets and that $B_\beta(f) = RB \cup LB$.

Let T be a minimum set such that $f_{\overline{T}}$ is convex. Lemma 3.3.2 implies that there exists a convex function that is equal to f for every $x \notin T$, and that $|T| = \epsilon_{con}(f)n$. Let

$$\begin{aligned} T' &= \left\{ k \mid k-1 \in T \vee k \in T \vee k+1 \in T \right\} \\ CT' &= \left\{ C(k) \mid k \leq n-1 \wedge k \in T' \right\} \end{aligned} \quad (3.26)$$

Observe that $|CT'| \leq |T'| \leq 3|T|$.

Consider the following **Crediting Procedure**. Each pair is assigned credit. We define $\sigma(C(i))$ to be the credit that $C(i)$ is being assigned in the algorithm. Initially, each pair in $RB \cap CT'$ is assigned 1 credit, and all other pairs are assigned 0 credit. Each $C(i) \in RB \setminus CT'$ among $C(n-1), \dots, C(1)$ in this order, *spreads* one credit among all pairs $C(t)$ such that $i < t \leq j_i$ and $C(t) \not\prec C(i)$. The word spread means that the credit always goes to whoever has the least credit among those pairs.

Crediting Procedure

For every $i \in [n - 1]$

- If $C(i) \in RB \cap CT'$ then $\sigma(C(i)) = 1$.
- Else $\sigma(C(i)) = 0$.

For $i = n - 1$ downto 1 do

- If $C(i) \in RB \setminus CT'$
 - $L = \{C(t) \mid i < t \leq j_i \wedge C(i) \not\sim C(t)\}$
 - $\alpha = 1$
 - *While*($\alpha > 0$)
 - * $C(k) = \operatorname{argmin}_{\sigma(C(t))} L$
 - * $Q = \{C(t) \in L \mid \sigma(C(t)) = \sigma(C(k))\}$
 - * $C(k') = \operatorname{argmin}_{\sigma(C(t))} (L \setminus Q)$
 - * $\Delta = \min \left\{ \frac{\alpha}{|Q|}, \sigma(C(k')) - \sigma(C(k)) \right\}$
 - * For every $C(t) \in Q$, $\sigma(C(t)) = \sigma(C(t)) + \Delta$
 - * $\alpha = \alpha - |Q| \cdot \Delta$

Consider the following two claims, which we later prove.

Claim 3.4.1 *Only pairs in CT' are assigned credit during the crediting procedure.*

Claim 3.4.2 *At the end of the crediting procedure all $k \in [n - 1]$ satisfy $\sigma(C(k)) \leq 2 + 4\beta/(1 - 2\beta)$*

Before we proceed, consider the following notation. Let $\mathcal{T}_{i'}$ denote the time in the crediting procedure when i is assigned the value i' . Also let $L_{i'}$ denote the value of L when $\mathcal{T}_{i'-1}$. For every pair $C(j)$ let $\sigma_{i'}(C(j))$ denote the value of $\sigma(C(j))$ when $\mathcal{T}_{i'-1}$. Observe that i is decreasing, and so $\mathcal{T}_{i'}$ denotes the end of the iteration in which $i = i' + 1$ and the beginning of the iteration in which $i' = i$. Also observe that $L_{i'}$ and $\sigma_{i'}(C(j))$ denote the values of L and $\sigma(C(j))$, respectively, at the end of the iteration in which $i = i'$. Let us now formalize the notion of *spread*.

- For every $j_1 \neq j_2$, the credit that pair $C(j_1)$ spreads on pair $C(j_2)$ is the credit that pair $C(j_2)$ is assigned after \mathcal{T}_{j_1} and before \mathcal{T}_{j_1-1} . That is, $\sigma_{j_1}(C(j_2)) - \sigma_{j_1+1}(C(j_2))$.
- For every j , the credit that pair $C(j)$ spreads on itself is the credit that it is assigned before \mathcal{T}_{n-1} . That is, $\sigma_n(C(j))$.

Before we prove Claims 3.4.1 and 3.4.2, let us complete the proof of the lemma. Observe that each pair in $RB \cap CT'$ spreads exactly 1 credit on itself and no credit on other pairs. Also observe that each pair in $RB \setminus CT'$ spreads exactly 1 credit on other pairs, and no credit on itself. In other words, every pair in RB spreads exactly 1 credit. Therefore at the end of the crediting procedure,

$$|RB| \leq \sum_{j \in [n-1]} \sigma(C(j)) \tag{3.27}$$

Also, Claims 3.4.1 and 3.4.2 imply that

$$\sum_{j \in [n-1]} \sigma(C(j)) \leq |CT'| \cdot (2 + 4\beta/(1 - 2\beta)) \quad (3.28)$$

Equations (3.27) and (3.28) imply that

$$|RB| \leq |CT'| \cdot (2 + 4\beta/(1 - 2\beta)) \quad (3.29)$$

A similar (and symmetrical) analysis would show that

$$|LB| \leq |CT'| \cdot (2 + 4\beta/(1 - 2\beta)) \quad (3.30)$$

Therefore

$$|B_\beta(f)| = |RB| + |LB| \leq (4 + 8\beta/(1 - 2\beta))|CT'| \quad (3.31)$$

Recall that $|CT'| \leq |T'| \leq 3 \cdot |T| = 3 \cdot \epsilon_{con}(f)n$ and the second part of the lemma follows.

It is left to prove Claims 3.4.1 and 3.4.2.

Proof of Claim 3.4.1: Let $k < n$ be such that $\sigma(C(k)) > 0$ in the end of the crediting procedure. Therefore,

- Either $C(k)$ spreads 1 credit on itself at the beginning of the procedure, so that necessarily $C(k) \in CT'$.
- Or there exists $j < k$ such that $C(j)$ spreads credit on $C(k)$. Therefore, $C(j) \in RB \setminus CT'$ and $C(j) \not\sim C(k)$. The fact that $C(j) \notin CT'$ and $j < n$ implies that $j \notin T'$. Therefore $j \notin T$, and $j + 1 \notin T$. Lemma 3.3.2 implies that there is a function, name it $g : \Sigma \mapsto \mathfrak{R}$, that is convex and equal to f on every element in $\Sigma \setminus T$. Lemma 3.3.5 implies that every two pairs in g are co-convex. Therefore, the fact that $C(j)$ and $C(k)$ are not co-convex and $j, j + 1 \notin T$ implies that $g(k) \neq f(k)$ or $g(k + 1) \neq f(k + 1)$. That is, $k \in T$ or $k + 1 \in T$. Therefore $k \in T'$, which means that $C(k) \in CT'$.

In both cases, $C(k) \in CT'$. ■

Proof of Claim 3.4.2: Suppose by contradiction that after the crediting procedure ends, some pair $C(k)$ satisfies $\sigma(C(k)) > 2 + 4\beta/(1 - 2\beta)$. Let i' be the value of i when $\sigma(C(k))$ becomes larger than $2 + 4\beta/(1 - 2\beta)$. Recall that i is decreasing. Therefore, $\sigma_{i'+1}(C(k)) \leq 2 + 4\beta/(1 - 2\beta)$ and $\sigma_{i'}(C(k)) > 2 + 4\beta/(1 - 2\beta)$ (we may also say that $C(i')$ is the right-big pair that causes $\sigma(C(k))$ to reach over $2 + 4\beta/(1 - 2\beta)$). Observe that according to the crediting procedure, every pair $C(j) \in L_{i'}$ satisfies $\sigma_{i'}(C(j)) > 2 + 4\beta/(1 - 2\beta)$. Therefore,

$$|L_{i'}| \cdot (2 + 4\beta/(1 - 2\beta)) < \sum_{C(j) \in L_{i'}} \sigma_{i'}(C(j)) \quad (3.32)$$

Since $C(i')$ is right-big, then $|L_{i'}| \geq (1/2 - \beta)(j_{i'} - i' + 1)$. Therefore,

$$\begin{aligned} j_{i'} - i' + 1 &= (1/2 - \beta)(j_{i'} - i' + 1)(2 + 4\beta/(1 - 2\beta)) \\ &\leq |L_{i'}| \cdot (2 + 4\beta/(1 - 2\beta)) \end{aligned} \quad (3.33)$$

Let $D(i') = \{C(i'), C(i' + 1), \dots, C(j_{i'})\}$. We wish to bound the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$. Observe the following.

- Every $C(j') \in D(i')$ satisfies:
 - Either $C(j') \in RB \cap CT'$, which means that it spreads 1 credit on itself, and does not spread any credit on other pairs. Therefore it contributes at most 1 to the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$.
 - Or $C(j') \in RB \setminus CT'$, which means that it does not spread credit on itself, and that it spreads at most 1 credit on pairs in $D(i')$. Therefore it contributes at most 1 to the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$.
 - Or else, in which case $C(j')$ does not spread credit at all. Therefore it contributes 0 to the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$.

That is, each $C(j') \in D(i')$ contributes at most 1 to the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$.

- For every $C(j') \notin D(i')$:
 - If $j' < i'$ then $C(j')$ spreads credit only after $\mathcal{T}_{i'-1}$. Therefore it has no affect on $\sigma_{i'}$.
 - If $j' > j_{i'}$ then $C(j')$ does not spread credit on pairs in $D(i')$. Therefore it has no affect on $\{\sigma_{i'}(C(i')), \sigma_{i'}(C(i'+1)), \dots, \sigma_{i'}(C(j_{i'}))\}$.

That is, each $C(j') \notin D(i')$ contributes 0 to the value of $\sum_{C(j) \in D(i')} \sigma_{i'}(C(j))$.

Therefore,

$$\sum_{C(j) \in D(i')} \sigma_{i'}(C(j)) \leq |D(i')| = j_{i'} - i' + 1 \quad (3.34)$$

Observe that $L_{i'} \subseteq D(i')$. Therefore

$$\sum_{C(j) \in L_{i'}} \sigma_{i'}(C(j)) \leq \sum_{C(j) \in D(i')} \sigma_{i'}(C(j)) \quad (3.35)$$

Equations (3.32), (3.34), and (3.35) contradict equation (3.33). Therefore, the assumption that leads to Equation (3.32) is not correct. That is, when the crediting procedure ends, all k satisfy $\sigma(C(k)) \leq 2 + 4\beta/(1 - 2\beta)$. ■

■ (Lemma 3.4.1)

3.5 A Distance Approximation Algorithm

The result of this section is presented in the following Lemma.

Lemma 3.5.1 • *At the end of Algorithm 3.5.2, with high probability,*

$$\min\{\epsilon_{con}(f)/25, \epsilon_{con}(f) - \delta\} \leq \hat{\epsilon} \leq \epsilon_{con}(f)$$

- *Let $\alpha = \max(\epsilon_{con}(f), \delta)$. The expected time and query complexity of Algorithm 3.5.2 is*

$$O\left(\frac{\log n}{\alpha} \cdot \log \frac{1}{\alpha} \cdot \log \log \frac{1}{\alpha} \cdot \log \frac{\log n}{\alpha}\right)$$

Proof of Theorem 1.2.4 (see page 8): Observe that

$$\epsilon_{con}(f)/25 - \delta \leq \min\{\epsilon_{con}(f)/25, \epsilon_{con}(f) - \delta\}$$

Theorem 1.2.4 is immediate from Lemma 3.5.1. ■

The algorithms in this section are given a query access to a function $f : \Sigma \mapsto \mathfrak{R}$. Observe that for every i, j , determining whether $C(i) \sim C(j)$ or not requires $O(1)$ computations. Consider the following procedure.

Procedure β -big-test₂(i, j, β, γ)

$i, j \leq n - 1$ satisfy $i \neq j$. $0 < \gamma, \beta \leq 1$.

- Let $m = \Theta(\frac{1}{\beta^2} \log(\frac{1}{\gamma}))$, and $TH = (1/2 - 3\beta/4)m$.
- – If ($j > i$) let $M = \{k_q\}_{q=1}^m$ be a set of uniformly independently selected elements from the set $\{i + 1, \dots, j\}$.
– If ($j < i$) let $M = \{k_q\}_{q=1}^m$ be a set of uniformly independently selected elements from the set $\{j, \dots, i - 1\}$.
- For every $q \in [m]$, let $X_q = 1$ if $C(i) \not\sim C(k_q)$ and $X_q = 0$ otherwise.
- Let $R = \sum_{q=1}^m X_q$. If $R \geq TH$ return 1. Else return 0.

Lemma 3.5.2 *The following is correct for every $i, j \in \Sigma$ such that $i \neq j$.*

1. *If $C(i)$ is $\frac{\beta}{2}$ -big with respect to j then β -big-test₂(i, j, β, γ) returns 1 with probability at least $1 - \gamma$.*
2. *If $C(i)$ is not β -big with respect to j then β -big-test₂(i, j, β, γ) returns 0 with probability at least $1 - \gamma$.*

Proof: Assume without loss of generality that $j > i$.

Item 1. If $C(i)$ is $\frac{\beta}{2}$ -big with respect to j then

$$\left| \left\{ i < k \leq j \mid C(i) \not\sim C(k) \right\} \right| \geq (1/2 - \beta/2)(j - i + 1) \quad (3.36)$$

and the additive chernoff bound implies that with probability at least $1 - \gamma$

$$R = \frac{1}{m} \sum_{q=1}^m X_q \geq (1/2 - \beta/2) - \beta/4 = TH \quad (3.37)$$

Item 2. If $C(i)$ is not β -big with respect to j then

$$\left| \left\{ i < k \leq j \mid C(i) \not\sim C(k) \right\} \right| < (1/2 - \beta)(j - i + 1) \quad (3.38)$$

and the additive chernoff bound implies that with probability at least $1 - \gamma$

$$R = \frac{1}{m} \sum_{q=1}^m X_q < (1/2 - \beta) + \beta/4 = TH \quad (3.39)$$

The lemma follows. ■

Lemma 3.5.3 Let $1 \leq i \leq n - 1$. For every $0 < \beta \leq 1/2$

1. If $C(i)$ is 0-big then there exists $k \in \mathcal{Z}$ such that $C(i)$ is $\frac{\beta}{2}$ -big with respect to $j = i \pm \lfloor \left(1 + \frac{\beta}{4}\right)^k \rfloor$.
2. If $C(i)$ is not β -big then there is no j such that $C(i)$ is β -big with respect to j .

Proof: The second item directly follows from the definition of β -big.

We turn to the first item. Let j_i be the witness to $C(i)$'s 0-bigness. Assume that $j_i > i$. There exists a $k \in \mathcal{Z}$ such that:

$$i + \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor \geq j_i \geq i + \left(1 + \frac{\beta}{4}\right)^{k-1} \quad (3.40)$$

Let $j = i + \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor$ (if $i + \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor > n$ then let $j = n$). Therefore

$$\left((j_i - i)\left(1 + \frac{\beta}{4}\right) + 1\right) \geq (j - i + 1) \quad (3.41)$$

The fact that $C(i)$ is 0-big with respect to j_i implies that

$$\left| \left\{ i < t \leq j_i \mid C(t) \not\sim C(i) \right\} \right| \geq \frac{j_i - i + 1}{2} \quad (3.42)$$

Also observe that

$$\frac{j_i - i + 1}{2} \geq \left(1/2 - \beta/2\right) \left((j_i - i)\left(1 + \frac{\beta}{4}\right) + 1\right) \quad (3.43)$$

(it is equivalent to $\frac{j_i - i}{4}\beta^2 + \left(1 + \frac{3}{4}(j_i - i)\right)\beta \geq 0$ which is correct for every $0 < \beta \leq 1/2$). Equations (3.41), (3.42) and (3.43) imply that

$$\left| \left\{ i < t \leq j \mid C(t) \not\sim C(i) \right\} \right| \geq (1/2 - \beta/2)(j - i + 1) \quad (3.44)$$

The proof for the case of $j_i < i$ is similar. ■

Consider some pair $C(i)$. Lemma 3.5.3 implies that by going over all k and checking whether $j = i \pm \lfloor \left(1 + \frac{\beta}{4}\right)^k \rfloor$ is a witness to $C(i)$'s $\frac{\beta}{2}$ -bigness, we could tell if $C(i)$ is not 0-big, and if $C(i)$ is β -big. Therefore, we do the following.

Procedure β -big-test₁(i, β, γ)

γ and β satisfy $0 < \gamma \leq 1$ and $0 < \beta \leq 1/2$.

- For $k = 1$ to $\lceil \log_{1+\frac{\beta}{4}}(n - i) \rceil$
 - Let $j = \min \left\{ n, i + \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor \right\}$.
 - If β -big-test₂($i, j, \beta, \frac{\beta}{8 \log n} \gamma$) = 1 return 1.
- For $k = 1$ to $\lceil \log_{1+\frac{\beta}{4}}(i - 1) \rceil$
 - Let $j = \max \left\{ 1, i - \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor \right\}$.
 - If β -big-test₂($i, j, \beta, \frac{\beta}{8 \log n} \gamma$) = 1 return 1.
- If the algorithm has not returned 1 so far, then return 0.

Observe that the time and query complexity of β -big-test₁(i, β, γ) is $O\left(\frac{\log n}{\beta^3} \log\left(\frac{\log n}{\beta \cdot \gamma}\right)\right)$.

Lemma 3.5.4 *Let $1 \leq i \leq n - 1$.*

1. *If $C(i)$ is 0-big then $\Pr[\beta$ -big-test₁(i, β, γ) = 1] $\geq 1 - \gamma$.*
2. *If $C(i)$ is not β -big then $\Pr[\beta$ -big-test₁(i, β, γ) = 0] $\geq 1 - \gamma$.*

Proof:

Item 1. If $C(i)$ is 0-big then the first item of Lemma 3.5.3 implies that there exists $k \in \mathcal{Z}$ such that $C(i)$ is $\frac{\beta}{2}$ -big with respect to $j = i \pm \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor$. The first item of Lemma 3.5.2 implies that β -big-test₂($i, j, \beta, \frac{\beta}{8 \log n} \gamma$) returns 1 with probability at least $1 - \frac{\beta}{8 \log n} \gamma > 1 - \gamma$. Therefore, β -big-test₁(i, β, γ) returns 1 with probability at least $1 - \gamma$.

Item 2. If $C(i)$ is not β -big, the second item of Lemma 3.5.3 and the second item of Lemma 3.5.2 imply that for every k and $j = i \pm \left\lfloor \left(1 + \frac{\beta}{4}\right)^k \right\rfloor$, the probability that β -big-test₂($i, j, \beta, \frac{\beta}{8 \log n} \gamma$) returns 1 is at most $\frac{\beta}{8 \log n} \gamma$. Observe that the number of times that β -big-test₂ is called in β -big-test₁ is at most

$$\log_{1+\frac{\beta}{4}}(n-i) + \log_{1+\frac{\beta}{4}}(i-1) + 2 \leq \frac{8 \log n}{\beta} \quad (3.45)$$

The second part of the lemma follows from this and the union bound. \blacksquare

Using β -big-test₁, we can now distinguish between the case that there are many 0-big pairs and the case that there are not too many β -big pairs. This and Lemma 3.4.1 (see page 56) provide the following tolerant testing algorithm.

Algorithm 3.5.1 *Tolerant Testing Convexity*

Given query access to a function $f : \Sigma \mapsto \mathfrak{R}$ and parameters $\epsilon, 0 < \beta \leq 1/6, 0 < \gamma \leq 1$:

- Let $m = \Theta\left(\frac{1}{\epsilon \cdot \beta^2} \log \frac{1}{\gamma}\right)$ and $M = \{i_q\}_{q=1}^m$ be a set of uniformly independently selected elements in $\{1, \dots, n-1\}$.
- For every $q \in [m]$ let $\hat{X}_q = \beta$ -big-test₁($i_q, \beta, \frac{\gamma}{2m}$).
- If $\frac{1}{m} \sum_{q=1}^m \hat{X}_q > (1 - \beta)\epsilon$ return 0. Else return 1.

Observe that the time and query complexity of Algorithm 3.5.1 is

$$\Theta\left(\frac{\log n}{\epsilon \cdot \beta^5} \log \frac{1}{\gamma} \cdot \log\left(\frac{\log n}{\epsilon \cdot \beta \cdot \gamma}\right)\right) \quad (3.46)$$

Lemma 3.5.5 1. *If $\epsilon_{\text{con}}(f) > 2\epsilon$ then Algorithm 3.5.1 returns 0 with probability at least $1 - \gamma$.*

2. *If $\epsilon_{\text{con}}(f) \leq \frac{\epsilon}{12}(1 - 6\beta)$ then Algorithm 3.5.1 returns 1 with probability at least $1 - \gamma$.*

Proof:

Item 1. Assume that $\epsilon_{con}(f) > 2\epsilon$. Lemma 3.4.1 implies that $|B_0(f)|/n > \epsilon$. For every $q \in [m]$ let $X_q = 1$ if $C(i_q)$ is 0-big, and $X_q = 0$ otherwise. The multiplicative chernoff bound implies that with probability at least $1 - \gamma/2$

$$\frac{1}{m} \sum_{q=1}^m X_q > (1 - \beta)\epsilon \quad (3.47)$$

Lemma 3.5.4 implies that for every $q \in [m]$, if $X_q = 1$ then $\hat{X}_q = 1$ with probability at least $1 - \frac{\gamma}{2m}$. Therefore, by the union bound, with probability at least $1 - \gamma/2$, for every $q \in [m]$, if $X_q = 1$ then $\hat{X}_q = 1$. Hence,

$$\frac{1}{m} \sum_{q=1}^m \hat{X}_q \geq \frac{1}{m} \sum_{q=1}^m X_q \quad (3.48)$$

Summing up the probabilities that Equations (3.47) and (3.48) are not correct, we have that with probability at least $1 - \gamma$ Algorithm 3.5.1 returns 0.

Item 2. Assume that $\epsilon_{con}(f) \leq \frac{\epsilon}{12}(1 - 6\beta)$. Lemma 3.4.1 and the fact that $\beta \leq 1/6$ imply that

$$\begin{aligned} |B_\beta(f)| &\leq 12\epsilon_{con}(f) \cdot \left(1 + \frac{2\beta}{1 - 2\beta}\right) \cdot n \\ &< \epsilon \cdot (1 - 6\beta) \cdot (1 + 4\beta) \cdot n \\ &< \epsilon(1 - 2\beta) \cdot n \end{aligned} \quad (3.49)$$

Now, for every $q \in [m]$ let $X_q = 1$ if $C(i_q)$ is β -big, and $X_q = 0$ otherwise. Observe that $(1 + \beta)(1 - 2\beta) < 1 - \beta$. Therefore

$$\begin{aligned} &\Pr\left[\frac{1}{m} \sum_{q=1}^m X_q > (1 - \beta)\epsilon \mid |B_\beta(f)| < \epsilon(1 - 2\beta) \cdot n\right] \\ &\leq \Pr\left[\frac{1}{m} \sum_{q=1}^m X_q > (1 + \beta)(1 - 2\beta)\epsilon \mid |B_\beta(f)| < \epsilon(1 - 2\beta) \cdot n\right] \\ &\leq \Pr\left[\frac{1}{m} \sum_{q=1}^m X_q > (1 + \beta)(1 - 2\beta)\epsilon \mid |B_\beta(f)| = \epsilon(1 - 2\beta) \cdot n\right] \\ &= \Pr\left[\frac{1}{m} \sum_{q=1}^m X_q > (1 + \beta) \frac{|B_\beta(f)|}{n} \mid |B_\beta(f)| = \epsilon(1 - 2\beta) \cdot n\right] \end{aligned} \quad (3.50)$$

Equations (3.49), (3.50) and the multiplicative chernoff bound imply that with probability at least $1 - \gamma/2$.

$$\frac{1}{m} \sum_{q=1}^m X_q \leq (1 - \beta)\epsilon \quad (3.51)$$

Lemma 3.5.4 implies that for every $q \in [m]$, if $X_q = 0$ then $\hat{X}_q = 0$ with probability at least $1 - \frac{\gamma}{2m}$. Therefore, by the union bound, with probability at least $1 - \gamma/2$, for every $q \in [m]$, if $X_q = 0$ then $\hat{X}_q = 0$. Hence,

$$\frac{1}{m} \sum_{q=1}^m \hat{X}_q \leq \frac{1}{m} \sum_{q=1}^m X_q \quad (3.52)$$

Summing up the probabilities that Equations (3.51) and (3.52) are not correct, we have that with probability at least $1 - \gamma$ Algorithm 3.5.1 returns 1. ■

For simplicity we set $\beta = \frac{1}{294}$, which means that $\frac{1-6\beta}{12} = \frac{1}{12.25}$ (see Lemma 3.5.5 for the meaning of that value). Also, the time and query complexity of Algorithm 3.5.1 in this case is

$$Q'_{3.5.1}(\epsilon, \gamma) = \Theta\left(\frac{\log n}{\epsilon} \log \frac{1}{\gamma} \cdot \log\left(\frac{\log n}{\epsilon \cdot \gamma}\right)\right) \quad (3.53)$$

Algorithm 3.5.2 *Distance Approximation to Convexity*

Given query access to a function $f : \Sigma \mapsto \mathfrak{R}$ and a parameter $0 \leq \delta \leq 1$:

First define the following procedure:

Estimate-Iter(L, H, δ, ν) {

- If $H < 25L$ or $H - L < \delta$ then return L .
- Run Algorithm 3.5.1 with $\epsilon = \frac{H+L}{2+1/12.25}$ and $\gamma = \frac{1}{6\nu^2}$.
If it accepts return **Estimate-Iter**($L, \frac{L+1223H}{1224}, \delta, \nu + 1$).
Else return **Estimate-Iter**($\frac{1223L+H}{1224}, H, \delta, \nu + 1$) }

Return $\hat{\epsilon} = \text{Estimate-Iter}(0, 1, \delta, 1)$

Proof of Lemma 3.5.1 (see page 60): The probability that Algorithm 3.5.1 gives a wrong answer depends on ν at the time it is called and is $\frac{1}{6\nu^2}$. Note that ν increases each time. Therefore, by the union bound, the probability that Algorithm 3.5.1 gives a wrong answer at least once is at most

$$\sum_{\nu=1}^{\infty} \frac{1}{6\nu^2} < \frac{1}{3} \quad (3.54)$$

Now assume that Algorithm 3.5.1 gives a correct answer each time it is called. Let $\epsilon_\nu, L_\nu, H_\nu$ be the value of ϵ , L and H respectively in iteration ν . We show by induction that for every ν we have $L_\nu \leq \epsilon_{\text{con}}(f) \leq H_\nu$. Obviously this is correct for $\nu = 1$. Assume that for some ν , $L_\nu \leq \epsilon_{\text{con}}(f) \leq H_\nu$. Recall that $25L_\nu < H_\nu$. Which means that

$$\begin{aligned} \frac{1}{12.25}\epsilon_\nu &= \frac{1}{12.25} \cdot \frac{L_\nu + H_\nu}{2 + 1/12.25} > \frac{1223L_\nu + H_\nu}{1224} \\ 2\epsilon_\nu &= \frac{2(L_\nu + H_\nu)}{2 + 1/12.25} < \frac{L_\nu + 1223H_\nu}{1224} \end{aligned} \quad (3.55)$$

There are three cases for the relation between $\epsilon_{\text{con}}(f)$ and ϵ_ν :

- $\frac{1}{12.25}\epsilon_\nu \leq \epsilon_{\text{con}}(f) \leq 2\epsilon_\nu$. Therefore $\epsilon_{\text{con}}(f)$ satisfies,

$$\begin{aligned} L_{\nu+1} &\leq \frac{1223L_\nu + H_\nu}{1224} \leq \frac{1}{12.25} \cdot \frac{L_\nu + H_\nu}{2 + 1/12.25} = \frac{\epsilon_\nu}{12.25} \\ &\leq \epsilon_{\text{con}}(f) \\ &\leq 2\epsilon_\nu = \frac{2(L_\nu + H_\nu)}{2 + 1/12.25} \leq \frac{L_\nu + 1223H_\nu}{1224} \leq H_{\nu+1} \end{aligned}$$

- $\epsilon_{con}(f) \geq 2\epsilon_\nu$, and then $L_{\nu+1} = \frac{1223L_\nu + H_\nu}{1224}$ and $H_{\nu+1} = H_\nu$ and so $\epsilon_{con}(f)$ satisfies,

$$L_{\nu+1} = \frac{1223L_\nu + H_\nu}{1224} < 2\epsilon_\nu \leq \epsilon_{con}(f) \leq H_\nu = H_{\nu+1} \quad (3.56)$$

- $\epsilon_{con}(f) \leq \frac{1}{12.25}\epsilon_\nu$, and then $L_{\nu+1} = L_\nu$ and $H_{\nu+1} = \frac{L_\nu + 1223H_\nu}{1224}$ and so $\epsilon_{con}(f)$ satisfies,

$$L_{\nu+1} = L_\nu \leq \epsilon_{con}(f) \leq \frac{1}{12.25}\epsilon_\nu \leq \frac{L_\nu + 1223H_\nu}{1224} = H_{\nu+1} \quad (3.57)$$

Therefore for every ν

$$L_\nu \leq \epsilon_{con}(f) \leq H_\nu \quad (3.58)$$

Assume that Algorithm 3.5.2 performs ν' steps. That is, $\hat{\epsilon} = L_{\nu'}$.

- If $H_{\nu'} - L_{\nu'} < \delta$ then $L_{\nu'}$ satisfies

$$\epsilon_{con}(f) - \delta \leq H_{\nu'} - \delta \leq L_{\nu'} \leq \epsilon_{con}(f) \quad (3.59)$$

- Else if $H_{\nu'} < 25L_{\nu'}$ then $L_{\nu'}$ satisfies

$$\frac{\epsilon_{con}(f)}{25} \leq \frac{H_{\nu'}}{25} \leq L_{\nu'} \leq \epsilon_{con}(f) \quad (3.60)$$

Therefore at the end of Algorithm 3.5.2 we have

$$\min\{\epsilon_{con}(f)/25, \epsilon_{con}(f) - \delta\} \leq \hat{\epsilon} \leq \epsilon_{con}(f) \quad (3.61)$$

The first part of the lemma follows.

For the second part of the lemma, we start by analyzing the complexity (time and query) in the case that Algorithm 3.5.1 gives a correct answer each time it is called. We need to upper bound $\frac{1}{H_\nu - L_\nu}$ and $\frac{1}{\epsilon_\nu}$.

- By the definition of δ , it is always smaller than the size of $H_\nu - L_\nu$. Also, the stopping condition of Algorithm 3.5.2 implies that $H_\nu > 25L_\nu$, which means that $H_\nu - L_\nu > \frac{24}{25}H_\nu \geq \frac{24}{25}\epsilon_{con}(f)$. And so $\frac{1}{H_\nu - L_\nu} = O(1/\alpha)$.
- Equation (3.58) implies that $\epsilon_\nu = \frac{L_\nu + H_\nu}{2+1/12.25} > \frac{H_\nu}{3} \geq \frac{\epsilon_{con}(f)}{3}$. Also, $\epsilon_\nu = \frac{L_\nu + H_\nu}{2+1/12.25} \geq \frac{H_\nu - L_\nu}{2+1/12.25} \geq \frac{\delta}{2+1/12.25}$. Therefore, $\frac{1}{\epsilon_\nu} = O(1/\alpha)$.

Therefore, the query and time complexity in case Algorithm 3.5.1 is always correct is at most

$$\begin{aligned} Q'(\delta) &= O\left(\log \frac{1}{\alpha} \cdot Q'_{3.5.1}\left(\alpha, \frac{1}{\log^2 \frac{1}{\alpha}}\right)\right) = O\left(\log \frac{1}{\alpha} \cdot \frac{\log n}{\alpha} \cdot \log \log \frac{1}{\alpha} \cdot \log \frac{\log n}{\alpha}\right) \\ &= \tilde{O}\left(\frac{\log n}{\alpha}\right) \end{aligned}$$

Now, assume Algorithm 3.5.1 may give an incorrect answer. As long as $L_\nu \leq \epsilon_{con}(f) \leq H_\nu$ the above analysis still holds. Assume this is not the case. First we observe that

$$\text{if } \nu_1 > \nu_2 \text{ then } L_{\nu_2} \leq L_{\nu_1} < H_{\nu_1} \leq H_{\nu_2} \quad (3.62)$$

Therefore, if for some ν' , $\epsilon_{con}(f) \leq L_{\nu'+1}$ (i.e. ν' is the smallest such one), then for every $\nu > \nu'$,

$$\epsilon_{con}(f) \leq L_\nu \leq H_\nu \quad (3.63)$$

and the above complexity analysis still holds.

The second case is that for some ν' , $\epsilon_{con}(f) \geq H_{\nu'+1}$ (i.e. ν' is the smallest such one). Therefore, for every $\nu \leq \nu'$ we have $\epsilon_{con}(f) \leq H_\nu$, and for every $\nu > \nu'$ we have $\epsilon_{con}(f) \geq H_\nu$. Let ν'' be the smallest ν such that $\nu > \nu'$ and for which Algorithm 3.5.1 gives a correct answer. Since $\epsilon_{con}(f) \geq H_{\nu''} \geq \epsilon_{\nu''}$, then

$$L_{\nu''+1} = \frac{1223L_{\nu''} + H_{\nu''}}{1224} \geq \frac{H_{\nu''}}{1224} \quad \text{and} \quad H_{\nu''+1} = H_{\nu''} \quad (3.64)$$

Recall that for every $\nu' \leq \nu < \nu''$, Algorithm 3.5.1 gives a wrong answer. Therefore, for every $\nu' \leq \nu < \nu''$, we have that

$$H_{\nu+1} = \frac{L_\nu + 1223H_\nu}{1224} \geq \frac{1223H_\nu}{1224} \quad (3.65)$$

Therefore, for every $\nu \geq \nu'' + 1$, ϵ_ν satisfies

$$\begin{aligned} \epsilon_\nu &\geq L_\nu \geq L_{\nu''+1} \geq \frac{1}{1224}H_{\nu''} \geq \frac{1}{1224} \cdot \left(\frac{1223}{1224}\right)^{\nu''-\nu'} H_{\nu'} \geq \frac{1}{1224} \cdot \left(\frac{1223}{1224}\right)^{\nu''-\nu'} \epsilon_{con}(f) \\ \epsilon_\nu &= \frac{H_\nu + L_\nu}{2 + 1/12.25} \geq \frac{\delta}{2 + 1/12.25} \end{aligned} \quad (3.66)$$

That is, $\frac{1}{\epsilon_\nu} = O\left(\left(\frac{1224}{1223}\right)^{\nu''-\nu'} \frac{1}{\alpha}\right)$. Also, the stopping condition of Algorithm 3.5.2 implies that

$$\begin{aligned} H_\nu - L_\nu &\geq \frac{24}{25}H_\nu > \frac{24}{25}L_\nu \geq \frac{24}{25} \cdot \frac{1}{1224} \cdot \left(\frac{1223}{1224}\right)^{\nu''-\nu'} \cdot \epsilon_{con}(f) \\ H_\nu - L_\nu &\geq \delta \end{aligned} \quad (3.67)$$

That is, $\frac{1}{H_\nu - L_\nu} = O\left(\left(\frac{1224}{1223}\right)^{\nu''-\nu'} \frac{1}{\alpha}\right)$. Equations (3.66) and (3.67) imply that the query and time complexity in this case is at most

$$O((1224/1223)^{\nu''-\nu'} \cdot Poly(\nu'' - \nu') \cdot Q'(\delta)) \quad (3.68)$$

The probability that for all ν such that $\nu' \leq \nu < \nu''$ Algorithm 3.5.1 indeed gives a wrong answer is at most

$$\left(\frac{1}{6\nu'^2}\right)^{\nu''-\nu'} < (1222/1224)^{\nu''-\nu'} \quad (3.69)$$

Therefore, the average complexity is at most

$$Q'(\delta) + \sum_{n=0}^{\infty} \left(\frac{1222}{1224}\right)^n \cdot \left(\left(\frac{1224}{1223}\right)^n \cdot Poly(n) \cdot Q'(\delta)\right) = Q'(\delta) \quad (3.70)$$

The lemma follows. \blacksquare

Bibliography

- [AC05] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimensions. In *Proceedings of the Twenty-Second Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 434–447, 2005.
- [ACCL04] N. Ailon, B. Chazelle, S. Comandur, and D. Liue. Estimating the distance to a monotone function. In *Proceedings of the Eight International Workshop on Randomization and Computation (RANDOM)*, pages 229–236, 2004.
- [BRW05] T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.
- [DGL⁺99] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of the Third International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 97–108, 1999.
- [EKK⁺00] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spott-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.
- [FF05] E. Fischer and L. Fortnow. Tolerant versus intolerant testing for boolean properties. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, 2005.
- [Fis01] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.
- [Fis04] E. Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1), 2004.
- [FLN⁺02] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 474–483, 2002.
- [FN01] E. Fischer and I. Newman. Testing of matrix properties. In *Proceedings of the Thirty-Second Annual ACM Symposium on the Theory of Computing (STOC)*, pages 286–295, 2001.
- [FN05] E. Fischer and I. Newman. Testing versus estimation of graph properties. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 138–146, 2005.
- [GGL⁺00] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samordinsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

- [GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- [Gol98] O. Goldreich. Combinatorial property testing - a survey. In *Randomization Methods in Algorithm Design*, pages 45–60, 1998.
- [GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, pages 302–343, 2002.
- [GR05] V. Guruswami and A. Rudra. Tolerant locally testable codes. In *Proceedings of the Ninth International Workshop on Randomization and Computation (RANDOM)*, pages 306–317, 2005.
- [HK03] S. Halevy and E. Kushilevitz. Distribution-free property testing. In *Proceedings of the Seventh International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 341–353, 2003.
- [HK04] S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. In *Automata, Languages and Programming: Thirty-First International Colloquium (ICALP)*, pages 721–732, 2004.
- [HK05] S. Halevy and E. Kushilevitz. A lower bound for distribution-free monotonicity testing. In *RANDOM-APPROX*, pages 330–341, 2005.
- [MR06] S. Marko and D. Ron. Distance approximation in bounded-degree and general sparse graphs. In *Proceedings of the Tenth International Workshop on Randomization and Computation (RANDOM)*, 2006.
- [PRR03] M. Parnas, D. Ron, and R. Rubinfeld. On testing convexity and submodularity. *SIAM Journal on Computing*, 32(5):1158–1184, 2003.
- [PRR04] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. ECC Report TR04-010, 2004, to appear in JCSS, 2004.
- [Ron01] D. Ron. Property testing. In *Handbook on Randomization, Volume II*, pages 597–649, 2001. Editors: S. Rajasekaran, P. M. Pardalos, J. H. Reif and J. D. P. Rolim.
- [RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

Appendix A

Chernoff Bounds

Let χ_1, \dots, χ_m be m independent random variables where $\chi_i \in [0, 1]$ for every $1 \leq i \leq m$. Let $p \stackrel{\text{def}}{=} \frac{1}{m} \sum_i \text{Exp}[\chi_i]$. (A special case, which occurs quite often, is when $\chi_i \in \{0, 1\}$ (*Bernoulli* random variables), and $\Pr[\chi_i = 1] = p$ for every i (i.e., the random variables are equally distributed).) Then, for every $\gamma \in (0, 1]$, the following bounds hold:

- (Additive Form)

$$\Pr \left[\frac{1}{m} \cdot \sum_{i=1}^m \chi_i > p + \gamma \right] < \exp(-2\gamma^2 m)$$

and

$$\Pr \left[\frac{1}{m} \cdot \sum_{i=1}^m \chi_i < p - \gamma \right] < \exp(-2\gamma^2 m)$$

- (Multiplicative Form)

$$\Pr \left[\frac{1}{m} \cdot \sum_{i=1}^m \chi_i > (1 + \gamma)p \right] < \exp(-\gamma^2 pm/3)$$

and

$$\Pr \left[\frac{1}{m} \cdot \sum_{i=1}^m \chi_i < (1 - \gamma)p \right] < \exp(-\gamma^2 pm/2)$$