

# Approximating the Distance to Monotonicity in High Dimensions

Shahar Fattal  
Department of EE – Systems  
Tel-Aviv University  
Ramat Aviv, ISRAEL  
fattalsh@post.tau.ac.il

Dana Ron  
Department of EE – Systems  
Tel-Aviv University  
Ramat Aviv, ISRAEL  
danar@eng.tau.ac.il

## Abstract

In this paper we study the problem of approximating the distance of a function over  $[n]^d$  to monotonicity. Namely, we are interested in randomized sublinear algorithms that approximate the Hamming distance between a given function and the closest monotone function.

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  $d$ . Building on work of Dodis et. al. (*Proceedings of RANDOM*, 1999), 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  $k < d$ . The quality of estimation that this reduction provides is linear in the size of the dimension and logarithmic in the size of the range  $|R|$  (if the range is infinite or just very large, then  $\log |R|$  can be replaced  $d \log n$ ). 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 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.

# 1 Introduction

This paper deals with the following type of approximation problems: For a predetermined property  $P$ , given query access to a function  $f$ , we would like to approximate the distance between  $f$  and a closest function that has a property  $P$ . Distance between functions is defined as the fraction of points on which the functions differ. In other words, we would like to estimate the minimum number of modifications (relative to the size of the domain) that must be made to the function  $f$  so that it obtain the property  $P$ . We refer to this quantity as the *distance to having the property  $P$* . This notion of *distance approximation* was first explicitly studied by Parnas et. al. [PRR06] together with the related notion of *tolerant testing*.<sup>1</sup> Both are natural extensions of (standard) property testing [RS96, GGR98].

We are interested in designing randomized distance approximation algorithms that have low query complexity and running time. In particular we aim for sublinear (e.g., logarithmic) complexity. Our focus in this work is on the property of monotonicity over the  $d$ -dimensional (non-binary) hypercube  $[n]^d$  (where  $[n] = \{1, \dots, n\}$ ).

The property of monotonicity has been studied quite extensively in the context of standard property testing [GGL<sup>+</sup>00, BRW05, EKK<sup>+</sup>00, DGL<sup>+</sup>99, Fis04, FLN<sup>+</sup>02, FN01, HK04, HK03, HK05]. Distance approximation to monotonicity was studied too, but the main focus in previous work [PRR06, ACCL04], was on the one-dimensional case. For higher dimensions, that is, for functions over  $[n]^d$  where  $d > 1$ , Parnas et. al. [PRR06] observed that it is possible to combine their algorithm for one-dimensional functions (or the more efficient algorithm of [ACCL04]), with a *dimension-reduction* lemma [HK05, AC06] to obtain an algorithm that, given a parameter  $\delta$ , outputs an estimate  $\hat{\epsilon}$  such that with probability at least  $2/3$ ,  $\frac{1}{d \cdot 2^{d+1}} \cdot \epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f)$ , where  $\epsilon_{mon}(f)$  is the distance between  $f$  and the closest monotone function. That is, the quality of the estimate degrades *exponentially* with the dimension  $d$ . The query complexity and running time of the algorithm are polynomial in  $\log n$  and  $1/\delta$ .

## 1.1 Our Results

In this work we significantly improve the quality of the estimate as compared to the aforementioned result in terms of the dependence on  $d$ . As we show in detail below, the quality of the estimate we obtain degrades only *linearly* with  $d$  and logarithmically with  $|R|$ . Since the latter dependence is actually on the effective size of the domain, that is, on the number of different function values,  $\log |R|$  can be replaced with  $d \log n$ . Thus, we obtain a better estimate when the range  $R$  of the function satisfies  $|R| < 2^{2^{d-1}}$ , or alternatively, when  $n < 2^{2^{d/2-1}}$ . That is, we improve the previous result whenever the dimension  $d$  is non-negligible with respect to either  $|R|$  or  $n$ . We note that for the special case of  $n = |R| = 2$  (that is, Boolean functions over the binary hypercube), the work of Goldreich et. al. [GGL<sup>+</sup>00] implicitly implies a distance approximation algorithm where the quality of the estimate degrades linearly with  $d$ .

In order to state our results, we introduce the following notation. We say that an algorithm is an  $\eta$ -*approximation algorithm with an additive error* for monotonicity if, given an additive error parameter  $\delta$  as input, it outputs, with success probability at least  $2/3$ , an estimate  $\hat{\epsilon}$  that sat-

---

<sup>1</sup>A tolerant property testing algorithm is required (with high probability), to accept objects that are  $\epsilon_1$ -close to having a given property  $P$  and reject objects that are  $\epsilon_2$ -far from having property  $P$ , for some parameters  $0 \leq \epsilon_1 < \epsilon_2 \leq 1$ . Standard property testing refers to the special case of  $\epsilon_1 = 0$ .

isfies:  $\frac{1}{\eta}\epsilon_{mon}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{mon}(f)$ .<sup>2</sup> For the sake of succinctness, we shall use the shorthand  $\eta$ -approximation algorithm to mean  $\eta$ -approximation algorithm with an additive error.

All our results are obtained by combining a dimension-reduction algorithm with approximation algorithms for low-dimensional functions. Theorem 1 is applicable to any range  $R$  (where, as noted previously,  $\log |R|$  can be replaced by  $d \log n$ ).

**Theorem 1 (Distance approximation to monotonicity for  $[n]^d \rightarrow R$  functions)** *There is a  $(5 \cdot d \log |R|)$ -approximation algorithm for monotonicity of  $[n]^d \rightarrow R$  functions. The query complexity and running time of the algorithm are  $\tilde{O}\left(\frac{\log n}{\delta^4}\right)$ .*

When dealing with Boolean functions we obtain the following results, which give a trade-off between the quality of the estimate and the complexity of the algorithm. Note that as opposed to Theorem 1, in the following results the complexity of the algorithm does not depend on  $n$ .

**Theorem 2 (Distance approximation to monotonicity for  $[n]^d \rightarrow \{0, 1\}$  functions)** *For distance approximation to monotonicity of Boolean functions over  $[n]^d$  we have the following results:*

1. A  $\frac{4d}{k}$ -approximation algorithm for any given parameter  $k \in [d]$  such that  $d/k$  is an integer, whose query complexity and running time are  $\tilde{O}\left(\frac{k}{\delta}\right)^{2k+6}$ .
2. A  $d$ -approximation algorithm whose query complexity and running time are  $\tilde{O}\left(\frac{1}{\delta^6}\right)$ .
3. A  $2d$ -approximation algorithm whose query complexity and running time are  $\tilde{O}\left(\frac{1}{\delta^4}\right)$ .

We note that it is possible to reduce the complexity of the last algorithm from  $\tilde{O}\left(\frac{1}{\delta^4}\right)$  to  $\tilde{O}\left(\frac{1}{\delta^3}\right)$  at the cost of a factor of 2 (or, more generally, any constant greater than 1) in the multiplicative factor of the estimate (for details see [Fat06]).

## 1.2 Techniques

We build on both the *dimension reduction* technique presented in [GGL<sup>+</sup>00] and [DGL<sup>+</sup>99] and on the *range reduction* technique in [DGL<sup>+</sup>99]. Dimension reduction, which was mentioned in the previous paragraphs, means that we are interested in the relation between  $\epsilon_{mon}(f)$  and  $\epsilon_{mon}(f')$  for functions  $f'$  that correspond to projections of  $f$  to lower dimensional hypercubes  $[n]^k$ . We first extend a result from [DGL<sup>+</sup>99] that applies to lines ( $k = 1$ ), to higher dimensional hypercubes. Specifically, we bound the distance of  $f$  to monotonicity in terms of the average distance to monotonicity of its projections to  $[n]^k$ , for  $k \geq 1$ . The quality of the bound improves as  $k$  increases. This result 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  $R$ , to the quality of such estimates in the case of  $|R| = 2$ . In particular we show that this relation holds for the estimate based on average distance to monotonicity of projections to lower dimensional hypercubes. Here we adapt a technique of [DGL<sup>+</sup>99] to our needs, and present an analysis that builds on the “violation graph”

---

<sup>2</sup>We note that if one does not allow an additive error (that is,  $\delta = 0$ ), then a dependence on  $1/\epsilon_{mon}(f)$  must be allowed.

of the function. We note that while the dimension reduction translates into an algorithmic tool, the range reduction is only an analysis tool.

We stress that though we build on [DGL<sup>+</sup>99], we cannot simply use their results as a black box. One reason is that we are interested in a dimension reduction that works for any  $k \geq 1$ . The second reason is, that even if we restrict ourselves to reducing the dimension to  $k = 1$ , the one-dimensional procedure applied in [DGL<sup>+</sup>99] is useful for testing but not for distance approximation.

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 by using the algorithm of [ACCL04] for one-dimensional functions as a subroutine. For the case  $R = \{0, 1\}$  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. Perhaps the most interesting algorithm for low-dimensional functions works by applying a new approach of approximating the distance of the function to its “sorted version”. This approximation is performed by selecting random points in the domain and constructing the value of the sorted function on these points (without, of course, constructing all of the sorted function).

### 1.3 Other Related Work

As noted previously, tolerant property testing and distance approximation were first explicitly studied by Parnas et. al. [PRR06]. 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 [PRR06, ACCL04, FF05, GR05], ensembles of points [PRR06], and graphs [FN05, MR06]. In recent work [FR07] we study the related property of convexity, and give a constant factor approximation of the distance to convexity of functions  $f : [n] \rightarrow \mathbb{R}$ .

## 2 Preliminaries

For  $d \in \mathcal{Z}$  let  $[d] = \{1, 2, \dots, d\}$ . For two strings  $x, y \in [n]^d$ ,  $x = x_1x_2 \dots x_d$  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  $x_i < y_i$ .

The additive and multiplicative Chernoff bounds which we use throughout the paper can be found in Appendix A.

### 2.1 Approximating the Distance to Properties

A *property*  $P$  of functions from domain  $D$  to range  $R$  is simply a subset of these functions. Therefore, we use the term “function  $f$  has property  $P$ ” and “ $f \in P$ ” interchangeably. Here we always assume that the range  $D$  is finite. The *distance* between two functions  $f, g : D \rightarrow R$  is the relative Hamming distance between the two, that is,

$$\text{dist}(f, g) \stackrel{\text{def}}{=} \frac{1}{|D|} \left| \left\{ x \in D : f(x) \neq g(x) \right\} \right|. \quad (1)$$

The distance of a function  $f$  to (having) a property  $P$  is the minimum distance between  $f$  and a function  $g$  that has property  $P$ . We denote this distance by  $\epsilon_P(f)$ . Namely,

$$\epsilon_P(f) \stackrel{\text{def}}{=} \min_{g \in P} \{\text{dist}(f, g)\}. \quad (2)$$

We say that  $f$  is  $\epsilon$ -far from having property  $P$  if  $\epsilon_P(f) > \epsilon$ , otherwise it is  $\epsilon$ -close.

**Definition 1 (Distance Approximation)** *An algorithm for approximating the distance to a property  $P$  (a distance approximation algorithm) is given query access to a function  $f$ , and outputs an estimate  $\hat{\epsilon}$  of  $\epsilon_P(f)$ . We say that  $\hat{\epsilon}$  is an  $(\eta, \delta)$ -estimate of  $\epsilon_P(f)$  for  $\eta \geq 1$  and  $0 \leq \delta \leq 1$  if<sup>3</sup>*

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

If for a fixed  $\eta$  and any given additive error parameter  $\delta$  and failure probability  $\gamma$ , given access to any function  $f$ , an algorithm outputs an  $(\eta, \delta)$ -estimate of  $\epsilon_P(f)$  with probability at least  $1 - \gamma$ , then we say it is an  $\eta$ -approximation algorithm for property  $P$ . If  $\eta = 1$  then we say it is a purely-additive approximation algorithm. If  $\gamma$  is not stated explicitly, then we assume that  $\gamma = 1/3$ .

## 2.2 Monotonicity

**Definition 2 (Monotonicity)** *For a partially ordered domain  $D$  and a fully ordered range  $R$ , we say that a function  $f : D \rightarrow R$  is monotone if  $f(x) \leq f(y)$  for every  $x, y \in D$  such that  $x \leq y$ .*

We denote the distance of a function  $f$  to monotonicity by  $\epsilon_{\text{mon}}(f)$ .

The notion of a *violation graph*, defined below, has played a role in several papers on testing and distance approximation for monotonicity (e.g., [EKK<sup>+</sup>00, PRR06, HK04, ACCL04]).

**Definition 3 (Violation Graph)** *For a function  $f : D \rightarrow R$  we say that a pair  $x, y \in D$  violate monotonicity with respect to  $f$  if  $x < y$  but  $f(x) > f(y)$  or  $x > y$  and  $f(x) < f(y)$ . Let the Violation Graph of  $f$ , denoted  $G_{\text{viol}}(f) = (D, E_{\text{viol}}(f))$ , be an undirected graph over the domain  $D$  where  $(x, y) \in E_{\text{viol}}(f)$  if and only if  $x, y$  violate monotonicity with respect to  $f$ .*

Lemma 2.1, stated next, follows from [HK04].

**Lemma 2.1** *For any function  $f : D \rightarrow R$ ,  $\epsilon_{\text{mon}}(f) = |VC(G_{\text{viol}}(f))|$ , where  $VC(G_{\text{viol}}(f))$  denotes a minimum size vertex cover in  $G_{\text{viol}}(f)$ .*

We consider functions over the domain  $D = [n]^d$ . For any  $i \in [d]$  we say that a function  $f$  over the domain  $[n]^d$  is *monotone in dimension  $i$* , if for every  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{d-i}$ , the one-dimensional function  $f_{i, \alpha, \beta}^1$  is monotone. For a set  $T \subseteq [d]$  we say that  $f$  is *monotone in dimensions  $T$* , if  $f$  is monotone in dimension  $i$  for every  $i \in T$ ,

---

<sup>3</sup>We have chosen a non-symmetric definition which allows the algorithm to underestimate  $\epsilon_P(f)$  but not to overestimate it. It is of course possible to slightly modify the algorithm so as to fit variants of this definition that allow for both an additive and a multiplicative error.

### 3 Main Ingredients

The main lemma on which our algorithms are based, is Lemma 3.1, given below. In order to state it we shall need the following notation.

For a function  $f : [n]^d \rightarrow R$ ,  $1 \leq i \leq j \leq d$ , and  $\alpha \in [n]^{i-1}$   $\beta \in [n]^{d-j}$ , we define the function  $f_{i,j,\alpha,\beta} : [n]^{j-i+1} \rightarrow R$  as follows:  $f_{i,j,\alpha,\beta}(x) = f(\alpha x \beta)$ . To simplify our notation, for a fixed choice of  $k \in \mathcal{Z}$  such that  $d$  is divisible by  $k$  and for  $q \in [d/k]$ ,  $\alpha \in [n]^{(q-1) \cdot k}$ , and  $\beta \in [n]^{d-q \cdot k}$ , let  $f_{q,\alpha,\beta}^k = f_{(q-1) \cdot k+1, q \cdot k, \alpha, \beta}$ . That is, each  $f_{q,\alpha,\beta}^k$  is a projection of  $f$  to a  $k$ -dimensional sub-cube of  $[n]^d$ , determined by  $\alpha$  and  $\beta$ .

**Lemma 3.1** *Let  $f : [n]^d \rightarrow R$ , and let  $k \in \mathcal{Z}$  be a fixed integer such that  $\frac{d}{k} \in \mathcal{Z}$ . Then*

1.  $\epsilon_{\text{mon}}(f) \geq \text{Exp}_{q,\alpha,\beta}[ \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) ]$ .
2.  $\epsilon_{\text{mon}}(f) \leq \log |R| \cdot \frac{2d}{k} \cdot \text{Exp}_{q,\alpha,\beta}[ \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) ]$

where the expectation  $\text{Exp}_{q,\alpha,\beta}[ \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) ]$  is taken over all  $q \in [d/k]$  and  $\alpha \in [n]^{(q-1) \cdot k}$ ,  $\beta \in [n]^{d-q \cdot k}$ .

We note that if  $|R| > |D| (= n^d)$ , and in particular if  $R$  is not finite, then  $\log |R|$  can be replaced by  $\log |D| = d \log n$ .

Based on Lemma 3.1 we can reduce the problem of distance approximation for  $d$ -dimensional functions, to the problem of distance approximation for  $k$ -dimensional functions over the same range, for  $k \in \mathcal{Z}$  s.t.  $\frac{d}{k} \in \mathcal{Z}$ . That is, assume we have an  $\eta$ -approximation algorithm for monotonicity of functions from the domain  $[n]^k$  to a range  $R$ . We denote this algorithm by  $\text{App}^k$ . Consider the algorithm in Figure 1.

**Algorithm 1** (*Distance Approximation by Dimension Reduction*)

- for  $j = 1$  to  $m = 18 \cdot \delta^{-2}$ :
  - Randomly and uniformly select  $q(j) \in \{1, \dots, \frac{d}{k}\}$ ,  $\alpha(j) \in [n]^{(q(j)-1) \cdot k}$ , and  $\beta(j) \in [n]^{d-q(j) \cdot k}$ .
  - Let  $\hat{\epsilon}_j$  be the output of algorithm  $\text{App}^k$  given an additive error parameter  $\frac{\delta}{3}$ , an allowed failure probability  $\frac{1}{6m}$  and query access to  $f_{q(j),\alpha(j),\beta(j)}^k$ .
- return  $\hat{\epsilon} = \frac{1}{m} \sum_{j=1}^m \hat{\epsilon}_j - \frac{\delta}{3}$ .

Figure 1: Algorithm 1. Distance approximation by dimension reduction.

**Lemma 3.2** *If  $\text{App}^k$  is an  $\eta$ -approximation algorithm for monotonicity of functions from  $[n]^k$  to  $R$ , then Algorithm 1 is an  $\eta \cdot \frac{2d \log |R|}{k}$ -approximation algorithm for functions from  $[n]^d$  to  $R$ . The query complexity of Algorithm 1 is  $\frac{1}{\delta^2} \cdot Q_{\text{App}^k}(\delta/3, \delta^2/C)$ , where  $C = 108$  and  $Q_{\text{App}^k}(\delta/2, \delta^2/C)$  is the query complexity of  $\text{App}^k$  when executed with an additive error parameter  $\delta/3$  and allowed failure probability  $\delta^2/C$ .*

**Proof:** Let  $p = \text{Exp}_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k)]$ , and for each  $j = 1, \dots, m$ , let  $\epsilon_j = \epsilon_{\text{mon}}(f_{q(j),\alpha(j),\beta(j)}^k)$ . Thus, each  $\epsilon_j$  is a random variable whose expectation is  $p$ , where the different  $\epsilon_j$ 's are independent. Let  $\tilde{\epsilon} = \frac{1}{m} \sum_{j=1}^m \epsilon_j$ . By the additive Chernoff bound (and our choice of  $m$ ):

$$\Pr \left[ |\tilde{\epsilon} - p| > \frac{\delta}{3} \right] < 2 \exp(-2m(\delta/3)^2) < 1/6. \quad (3)$$

By definition of  $\text{App}^k$ , for each  $j$ , with probability at least  $1 - \frac{1}{6m}$ ,

$$\frac{1}{\eta} \cdot \epsilon_j - \frac{\delta}{3} \leq \hat{\epsilon}_j \leq \epsilon_j, \quad (4)$$

and by the union bound, Equation (4) holds for all  $j = 1, \dots, m$  with probability at least  $5/6$ . By definition of  $\tilde{\epsilon}$  and  $\hat{\epsilon}$ ,

$$\Pr \left[ \frac{1}{\eta} \cdot \tilde{\epsilon} - \frac{2\delta}{3} \leq \hat{\epsilon} \leq \tilde{\epsilon} - \frac{\delta}{3} \right] \geq 5/6. \quad (5)$$

By combining Equations (3) and (5) we get that with probability at least  $2/3$ ,

$$\frac{1}{\eta} \cdot p - \delta \leq \hat{\epsilon} \leq p. \quad (6)$$

Finally, applying Lemma 3.1 we have that with probability at least  $2/3$ ,

$$\frac{k}{2d \log |R| \eta} \epsilon_{\text{mon}}(f) - \delta \leq \hat{\epsilon} \leq \epsilon_{\text{mon}}(f). \quad (7)$$

■

For  $k = 1$  let  $\text{App}^1$  be (a slight variant of) the algorithm of Ailon et. al. [ACCL04] for distance approximation to monotonicity of 1-dimensional functions. For any given constant  $\lambda > 0$ , this variant outputs a  $(2 + \lambda, \delta)$ -estimate (with high probability) and has query complexity and running time  $\tilde{O}\left(\frac{\log n}{\delta^2}\right)$ . By setting  $\lambda = 0.5$  and applying Lemma 3.1, we obtain Theorem 1.

The proof of the first item in Lemma 3.1 is simple, and we give the details below for completeness. In the next sections we: (1) prove the second item in Lemma 3.1 for the special case of  $R = \{0, 1\}$ ; (2) show how to extend the proof to a general range; (3) present several approximation algorithms for low dimensions and apply Lemma 3.2 to obtain our final results.

**Proof of Item 1 in Lemma 3.1.** Since for any choice of  $q \in [d/k]$ , the functions  $f_{q,\alpha,\beta}^k$  (for  $\alpha \in [n]^{(q-1) \cdot k}$  and  $\beta \in [n]^{d-q \cdot k}$ ) are over disjoint domains, we have that

$$\epsilon_{\text{mon}}(f) \cdot n^d \geq \sum_{\alpha \in [n]^{(q-1) \cdot k}, \beta \in [n]^{d-q \cdot k}} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) \cdot n^k. \quad (8)$$

Therefore,

$$\epsilon_{\text{mon}}(f) \cdot n^d \geq \max_{q \in \{1, \dots, \frac{d}{k}\}} \sum_{\alpha, \beta} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) \cdot n^k$$

$$\begin{aligned}
&\geq \frac{k}{d} \cdot \sum_{q=1}^{d/k} \sum_{\alpha \in [n]^{(q-1) \cdot k}, \beta \in [n]^{d-q \cdot k}} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) \cdot n^k \\
&= \frac{k \cdot n^k}{d} \sum_{q \in [d/k], \alpha \in [n]^{(q-1) \cdot k}, \beta \in [n]^{d-q \cdot k}} \epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k) \\
&= \frac{k \cdot n^k}{d} \cdot \frac{d}{k} \cdot n^{d-k} \cdot \text{Exp}_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k)] \\
&= \text{Exp}_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta}^k)] \cdot n^d
\end{aligned} \tag{9}$$

■

## 4 Proving Item 2 in Lemma 3.1 for $R = \{0, 1\}$

The next definition, which appeared in [GGL<sup>+</sup>00], is central to this section, and is also used by one of our algorithms for low-dimensional functions.

**Definition 4 (Sorting operator)** *Let  $f : [n]^d \rightarrow R$ , where  $R$  is a fully ordered set. For each  $i \in [d]$ , the function  $S_i[f] : [n]^d \rightarrow R$  is defined as follows: for each  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{d-i}$ , 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 one-dimensional projections to dimension  $i$ . For  $j \geq i$  let the multi-dimensional sorting operator  $S_{i,j}$  be defined as follows:  $S_{i,j}[f] = S_j[S_{j-1}[\dots[S_i[f]\dots]]$ .*

The next two lemmas are used in order to prove Item 2 in Lemma 3.1 for  $R = \{0, 1\}$ . Lemma 4.1 is a generalization of [GGL<sup>+</sup>00, Lemma 8].

**Lemma 4.1** *Let  $h : [n]^d \rightarrow \{0, 1\}$ . For every  $1 \leq i \leq j \leq d$  and  $\ell \notin \{i, \dots, j\}$ ,*

$$\sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{\text{mon}}(S_\ell[h]_{i,j,\alpha,\beta}) \leq \sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{\text{mon}}(h_{i,j,\alpha,\beta})$$

A simple but important corollary of Lemma 4.1 is:

**Corollary 4.2** *Let  $h : [n]^d \rightarrow \{0, 1\}$ . Then  $S_{1,d}[h]$  is monotone.*

**Proof:** By Lemma 4.1, if a function  $h$  is monotone in dimensions  $\{1, \dots, \ell - 1\}$  then  $S_\ell[h]$  is monotone in dimensions  $\{1, \dots, \ell\}$ . Therefore,  $S_{1,d}[h]$  is monotone in all dimensions. ■

**Lemma 4.3** *Let  $h : [n]^k \rightarrow \{0, 1\}$ . Then,*

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

We prove the two lemmas subsequently, and first prove the first item in Lemma 3.1 for the case of a binary range.

**Proof of Item 2 in Lemma 3.1 for  $R = \{0, 1\}$ .** Let  $f_{(q)} = S_{1,q \cdot k}[f]$ , where  $f_{(0)} = f$ . By Corollary 4.1,  $f_{(d/k)} = S_{1,d}[f]$  is monotone. Therefore,

$$\epsilon_{mon}(f) \leq \text{dist}(f, f_{(d/k)}) \leq \sum_{q=0}^{d/k-1} \text{dist}(f_{(q)}, f_{(q+1)}), \quad (10)$$

where the second inequality follows from the triangle inequality.

We next bound  $\text{dist}(f_{(q)}, f_{(q+1)})$  for any  $q \in \{1, \dots, d/k\}$ . For  $\alpha \in [n]^{q \cdot k}$ ,  $\beta \in [n]^{d-k(q-1)}$ , and  $x \in [n]^k$ , let  $g_{q,\alpha,\beta}(x) = f_{(q)}(\alpha x \beta)$ . For every  $q \in [d/k]$ :

$$\begin{aligned} \text{dist}(f_{(q)}, f_{(q+1)}) &= n^{-d} \cdot \sum_{\alpha \in [n]^{q \cdot k}, \beta \in [n]^{d-k(q-1)}} \left| \{x \in [n]^k : f_{(q)}(\alpha x \beta) \neq f_{(q+1)}(\alpha x \beta)\} \right| \\ &= n^{-d} \cdot \sum_{\alpha \in [n]^{q \cdot k}, \beta \in [n]^{d-k(q-1)}} \left| \{x \in [n]^k : g_{q,\alpha,\beta}(x) \neq S_{1,k}[g_{q,\alpha,\beta}](x)\} \right| \\ &\leq n^{-d} \cdot \sum_{\alpha \in [n]^{q \cdot k}, \beta \in [n]^{d-k(q-1)}} 2\epsilon_{mon}(g_{q,\alpha,\beta}) \cdot n^k \\ &= 2 \cdot \text{Exp}_{\alpha,\beta}[\epsilon_{mon}(g_{q,\alpha,\beta})] \end{aligned} \quad (11)$$

$$\leq 2 \cdot \text{Exp}_{\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^k)], \quad (12)$$

where Equation (11) follows from Lemma 4.3, and Equation (12) follows from Lemma 4.1.

By combining Equations (10) and (12) we get

$$\begin{aligned} \epsilon_{mon}(f) &\leq \sum_{q=0}^{d/k-1} \text{dist}(f_{(q)}, f_{(q+1)}) \\ &\leq \sum_{q=0}^{d/k-1} 2 \cdot \text{Exp}_{\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^k)] \\ &= \frac{2d}{k} \cdot \text{Exp}_{q,\alpha,\beta}[\epsilon_{mon}(f_{q,\alpha,\beta}^k)] \end{aligned} \quad (13)$$

and the claim follows.  $\blacksquare$

In order to prove Lemma 4.1 we first prove the following claim.

**Claim 4.4** For every  $1 \leq i \leq j \leq d$ ,  $\ell \notin \{i, \dots, j\}$ ,  $b \in [n-1]$  and function  $h : [n]^{\ell-1} \times \{b, b+1\} \times [n]^{d-\ell} \rightarrow \{0, 1\}$ ,

$$\sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{mon}(S_\ell[h]_{i,j,\alpha,\beta}) \leq \sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{mon}(h_{i,j,\alpha,\beta})$$

**Proof:** Let  $\tilde{h} = S_\ell[h]$  and let  $h^M : [n]^{\ell-1} \times \{b, b+1\} \times [n]^{d-\ell} \rightarrow \{0, 1\}$  be a function such that for every  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{d-j}$ , the function  $h_{i,j,\alpha,\beta}^M$  is monotone and

$$\text{dist}(h_{i,j,\alpha,\beta}^M, h_{i,j,\alpha,\beta}) = \epsilon_{mon}(f_{i,j,\alpha,\beta}) \quad (14)$$

In order to prove the claim, it suffices to show that there exists a function  $\tilde{h}^M : [n]^{\ell-1} \times \{b, b+1\} \times [n]^{d-\ell} \rightarrow \{0, 1\}$ , such that for every  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{d-j}$ , the function  $\tilde{h}_{i,j,\alpha,\beta}^M$  is monotone and

$$\sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \text{dist}(\tilde{h}_{i,j,\alpha,\beta}, \tilde{h}_{i,j,\alpha,\beta}^M) \leq \sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \text{dist}(h_{i,j,\alpha,\beta}, h_{i,j,\alpha,\beta}^M) \quad (15)$$

Before we do so, we introduce some more notation. For simplicity, and without loss of generality, assume that  $\ell = d$ ,  $i = 1$  and  $j < d$ . Let  $r \in \{b, b+1\}$  and  $\psi \in [n]^{d-j-1}$ . For every  $x \in [n]^j$  let

$$g_{\psi,r}(x) = h(x\psi r), \quad g_{\psi,r}^M(x) = h^M(x\psi r), \quad \tilde{g}_{\psi,r}(x) = \tilde{h}(x\psi r), \quad \tilde{g}_{\psi,r}^M(x) = \tilde{h}^M(x\psi r)$$

We now define  $\tilde{g}^M$  (and hence  $\tilde{h}^M$ ). For every  $x \in [n]^j$

$$\begin{aligned} \tilde{g}_{\psi,b}^M(x) &= \min\{g_{\psi,b}^M(x), g_{\psi,b+1}^M(x)\} \\ \tilde{g}_{\psi,b+1}^M(x) &= \max\{g_{\psi,b}^M(x), g_{\psi,b+1}^M(x)\} \end{aligned} \quad (16)$$

First we verify that  $\tilde{g}_{\psi,b}^M$  and  $\tilde{g}_{\psi,b+1}^M$  are monotone functions. Indeed, for every  $y > x$ ,

$$\begin{aligned} \tilde{g}_{\psi,b}^M(y) &= \min\{g_{\psi,b}^M(y), g_{\psi,b+1}^M(y)\} \geq \min\{g_{\psi,b}^M(x), g_{\psi,b+1}^M(x)\} = \tilde{g}_{\psi,b}^M(x) \\ \tilde{g}_{\psi,b+1}^M(y) &= \max\{g_{\psi,b}^M(y), g_{\psi,b+1}^M(y)\} \geq \max\{g_{\psi,b}^M(x), g_{\psi,b+1}^M(x)\} = \tilde{g}_{\psi,b+1}^M(x) \end{aligned} \quad (17)$$

It remains to show that,

$$\sum_{\psi} \sum_{r \in \{b, b+1\}} \text{dist}(\tilde{g}_{\psi,r}, \tilde{g}_{\psi,r}^M) \leq \sum_{\psi} \sum_{r \in \{b, b+1\}} \text{dist}(g_{\psi,r}, g_{\psi,r}^M) \quad (18)$$

We claim that for every  $\psi$ ,

$$\sum_{r \in \{b, b+1\}} \text{dist}(\tilde{g}_{\psi,r}, \tilde{g}_{\psi,r}^M) \leq \sum_{r \in \{b, b+1\}} \text{dist}(g_{\psi,r}, g_{\psi,r}^M) \quad (19)$$

Proving Equation (19) is a technical process of going over the 16 possible values for the combinations of  $g_{\psi,r}$  and  $g_{\psi,r}^M$  for  $r \in \{b, b+1\}$ . It is not hard to verify that for every  $x$  and  $\psi$  such that

$$\begin{aligned} &g_{\psi,b}(x) = 0, \quad g_{\psi,b+1}(x) = 1, \quad g_{\psi,b}^M(x) = 1, \quad g_{\psi,b+1}^M(x) = 0 \\ &OR \\ &g_{\psi,b}(x) = 1, \quad g_{\psi,b+1}(x) = 0, \quad g_{\psi,b}^M(x) = 0, \quad g_{\psi,b+1}^M(x) = 1 \end{aligned} \quad (20)$$

we have that

$$(\tilde{g}_{\psi,b}(x) \oplus \tilde{g}_{\psi,b}^M(x)) + (\tilde{g}_{\psi,b+1}(x) \oplus \tilde{g}_{\psi,b+1}^M(x)) = (g_{\psi,b}(x) \oplus g_{\psi,b}^M(x)) + (g_{\psi,b+1}(x) \oplus g_{\psi,b+1}^M(x)) - 2 \quad (21)$$

(so that Equation (19) holds with a strict inequality), and otherwise

$$(\tilde{g}_{\psi,b}(x) \oplus \tilde{g}_{\psi,b}^M(x)) + (\tilde{g}_{\psi,b+1}(x) \oplus \tilde{g}_{\psi,b+1}^M(x)) = (g_{\psi,b}(x) \oplus g_{\psi,b}^M(x)) + (g_{\psi,b+1}(x) \oplus g_{\psi,b+1}^M(x)). \quad (22)$$

The claim follows.  $\blacksquare$

**Proof of Lemma 4.1.** For any given  $\ell$ ,  $\alpha \in [n]^{\ell-1}$  and  $\beta \in [n]^{d-\ell}$ , we view  $h_{\ell,\ell,\alpha,\beta}$  as an  $n$ -dimensional 0/1 valued vector. Recall that the sorting operator  $S_\ell$  works by sorting each  $h_{\ell,\ell,\alpha,\beta}$  separately. In particular, it will be convenient to think of the sorting operator as applying the Bubble-Sort algorithm. Specifically, for each  $\alpha \in [n]^{\ell-1}$  and  $\beta \in [n]^{d-\ell}$ , sorting proceeds in  $\binom{n}{2}$  iterations. In each iteration, for some  $b \in [n-1]$ , if  $h_{\ell,\ell,\alpha,\beta}(b) > h_{\ell,\ell,\alpha,\beta}(b+1)$  (that is,  $h_{\ell,\ell,\alpha,\beta}(b) = 1$  while  $h_{\ell,\ell,\alpha,\beta}(b+1) = 0$ ) then a swap is performed (that is,  $h_{\ell,\ell,\alpha,\beta}(b)$  is set to 0 and  $h_{\ell,\ell,\alpha,\beta}(b+1) = 1$  is set to 1). The choice of  $b$  can be according to a pre-specified order (i.e.,  $b$  runs from 1 to  $n-1$ , and then from 2 to  $n-1$  and so on), or  $b$  can be selected arbitrarily conditioned on  $h_{\ell,\ell,\alpha,\beta}(b) > h_{\ell,\ell,\alpha,\beta}(b+1)$  (as long as such a  $b$  exists).

Let  $h^{(t)}$  denote the resulting function after  $t$  such steps, and let  $b^{(t)}$  denote the ‘ $b$ ’ that was considered in step  $t$ . Assume without loss of generality that  $\ell > j$ . Claim 4.4 implies that every  $t$  satisfies

$$\sum_{\alpha,\beta_1,\beta_2} \left( \sum_{r \in \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t)}) \right) \leq \sum_{\alpha,\beta_1,\beta_2} \left( \sum_{r \in \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t-1)}) \right) \quad (23)$$

where the sum is over  $\alpha \in [n]^{i-1}$ ,  $\beta_1 \in [n]^{\ell-1-j}$ , and  $\beta_2 \in [n]^{d-\ell}$ . Note that for  $r \in [n] \setminus \{b^{(t)}, b^{(t)+1}\}$  we have that  $h_{i,j,\alpha,\beta_1 r \beta_2}^{(t)} = h_{i,j,\alpha,\beta_1 r \beta_2}^{(t-1)}$ . Therefore,

$$\begin{aligned} & \sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{mon}(h_{i,j,\alpha,\beta}^{(t)}) \\ &= \sum_{\substack{\alpha \in [n]^{i-1} \\ \beta_1 \in [n]^{\ell-1-j}, \beta_2 \in [n]^{d-\ell}}} \left( \sum_{r \in [n] \setminus \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t)}) + \sum_{r \in \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t)}) \right) \\ &\leq \sum_{\substack{\alpha \in [n]^{i-1} \\ \beta_1 \in [n]^{\ell-1-j}, \beta_2 \in [n]^{d-\ell}}} \left( \sum_{r \in [n] \setminus \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t-1)}) + \sum_{r \in \{b^{(t)}, b^{(t)+1}\}} \epsilon_{mon}(h_{i,j,\alpha,\beta_1 r \beta_2}^{(t-1)}) \right) \\ &= \sum_{\alpha \in [n]^{i-1}, \beta \in [n]^{d-j}} \epsilon_{mon}(h_{i,j,\alpha,\beta}^{t-1}) \end{aligned} \quad (24)$$

which inductively proves the second part of Lemma 4.1.  $\blacksquare$

**Proof of Lemma 4.3.** Let  $g = S_{1,k}[h]$ . By Corollary 4.2, the function  $g$  is monotone. For  $\sigma, \sigma' \in \{0, 1\}$ , Let

$$X^\sigma = \{x \in [n]^k : g(x) = \sigma\}, \quad \text{and} \quad X^{\sigma,\sigma'} = \{x \in [n]^k : g(x) = \sigma, \text{ and } h(x) = \sigma'\}.$$

Since  $g$  is obtained from  $h$  by applying a (multi-dimensional) sorting operator,  $|X^{0,1}| = |X^{1,0}|$ . By definition of the distance between functions,  $\text{dist}(f, g) = n^{-k} \cdot (|X^{0,1}| + |X^{1,0}|) = 2 \cdot n^{-k} \cdot |X^{0,1}|$ . We shall show that there exists a matching of size  $|X^{0,1}|$  in  $G_{viol}(h)$ , which implies that  $\epsilon_{mon}(h) \geq n^{-k} \cdot |X^{0,1}|$ , and the lemma follows.

We view the multidimensional sorting operator  $S_{1,k}$  as *relocating* the labels of the function  $h$ . Namely, each 0/1 label  $h(x)$  is initially associated with the point  $x$ . Each application of the one-dimensional sorting operator  $S_i$  is viewed as swapping between 0 labels and 1 labels. In this manner, if a 1 label was initially associated with a point  $x$  (i.e.,  $h(x) = 1$ ), then, at the end of the

sorting process it is associated with a point  $y$ , so that  $g(y) = 1$ , where,  $y > x$ . Note that it is not necessarily the case that  $h(y) = 0$  (for  $k > 1$ ). However, we shall show that for each  $x \in X^{0,1}$  there is a unique  $z > x$  in  $X^{1,0}$ , thus obtaining the desired matching.

For a set of points  $Z$ , let  $L(Z)$  denote the set of points that are larger than points in  $Z$ . Namely,

$$L(Z) = \{y : \exists z \in Z \text{ s.t. } z < y\} . \quad (25)$$

For  $\sigma, \sigma' \in \{0, 1\}$  let

$$L^\sigma(Z) = L(Z) \cap X^\sigma \quad \text{and} \quad L^{\sigma, \sigma'}(Z) = L(Z) \cap X^{\sigma, \sigma'} . \quad (26)$$

To simplify the notation, let  $X = X^{0,1}$ . By definition of  $X$ , for every  $x \in X$ , the label of  $x$  is relocated by the sorting operator to some  $y \in L^1(X)$ . For each such point  $y$ , either  $y \in L^{1,0}(X)$ , or  $y \in L^{1,1}(X)$  (i.e.,  $h(y) = 1$ ). In the latter case the label of  $y$  was relocated to some  $y' > y$ , where necessarily  $y' \in L^1(X)$  as well. In turn, the label of  $y'$  was relocated to some  $y'' \in L^1(X)$ , where either  $y'' \in L^{1,0}(X)$  or  $y'' \in L^{1,1}(X)$ . Thus, for each  $x \in X$  there is a ‘‘relocation path’’ in  $L^1(X)$  that reaches some  $w \in L^{1,0}(X)$ . Since the paths are (vertex-)disjoint, and in particular their end-points are disjoint, we obtain a matching of size  $|X^{0,1}|$  in  $G_{viol}(h)$ , as claimed.

For an illustration of the argument in this proof, see Figure 2. ■

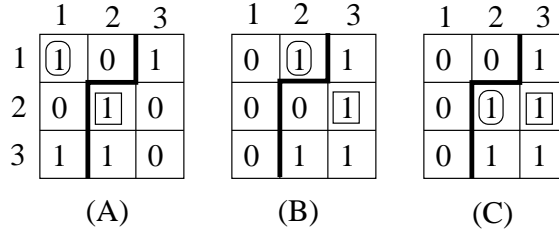


Figure 2: An illustration for the proof of Lemma 4.3. (A) depicts a function  $h : [3]^2 \rightarrow \{0, 1\}$ , (B) depicts  $S_1[h]$  and (C) depicts the monotone function  $g = S_{1,2}[h]$ . In all three functions, the bold line signifies the boundary between  $X^0$  (the points labeled 0 by  $g$ ) and  $X_1$  (the points labeled 1 by  $g$ ). The label of the point  $x = 11$  (where  $x \in X^{0,1}$ ) is marked by an ellipse. This label is relocated to  $x' = 12$  after  $S_1$  is applied, and then to  $y = 22$ , after  $S_2$  is applied. The label of  $y = 22$  (where  $y \in X^{1,1}$ ) is relocated to  $y' = 23$ , where  $y' \in X^{1,0}$ . Thus,  $x = 11$  is matched to  $y' = 23$ .

## 5 Proving Item 2 in Lemma 3.1 for general $R$

For a function  $f : [n]^d \rightarrow R$  and  $k \in \mathcal{Z}$  such that  $d/k \in \mathcal{Z}$ , let

$$\Gamma^k(f) = \frac{2d}{k} \cdot \text{Exp}_{q, \alpha, \beta}[\epsilon_{mon}(f_{q, \alpha, \beta}^k)] \quad (27)$$

In the previous section we showed that if  $R = \{0, 1\}$  then  $\epsilon_{mon}(f) \leq \Gamma^k(f)$ . In this section our goal is to extend this upper bound to any  $R$  and show that  $\epsilon_{mon}(f) \leq \log |R| \cdot \Gamma^k(f)$ , thus establishing Item 2 in Lemma 3.1. To this end we prove the following. Let  $\mathcal{F}$  be the family of all functions from some partially ordered domain  $V$  to a fully ordered finite range  $R$ . Then for every operator  $\Gamma$  defined over  $\mathcal{F}$  that has certain properties, if  $\epsilon_{mon}(f) \leq \Gamma(f)$  whenever  $f$  has a binary range, then

$\epsilon_{mon}(f) \leq \lceil \log |R| \rceil \cdot \Gamma(f)$  for all functions  $f$ . We then show that  $\Gamma^k$  as defined in Equation (27) satisfies these properties.

In order to define the properties that the operator  $\Gamma$  should have and prove that they suffice for our purposes, we introduce certain operations on the violation graph  $G_{viol}(f) = (V, E_{viol}(f))$ , based on which we can relate  $\epsilon_{mon}(f)$  to the distance to monotonicity of functions with a smaller range than  $f$ . As noted in the introduction, our proof is based on [DGL<sup>+</sup>99]. In that paper a particular operator  $\Gamma$  was considered. While that operator was useful for *testing* monotonicity (i.e., obtaining evidence that a function is not monotone), it cannot be used for the task of estimating the distance to monotonicity.

## 5.1 The High Level Idea

Recall that by Lemma 2.1,  $\epsilon_{mon}(f)$  equals the size of a minimum vertex cover in  $G_{viol}(f)$ . Let us name the elements in  $R$  by  $\{0, \dots, r-1\}$ , where  $r = 2^{\lceil \log |R| \rceil}$ . 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. Given this partition of  $V$  we can define a Boolean function, denoted  $f'$ , for which  $E'$  are the violating edges of  $G_{viol}(f')$ .

Next consider the operation of 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 3). Following this transformation we obtain a new function (denoted  $g$ ) for which we can show that there are no edges in the violation graph of  $g$  that cross 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, and define two corresponding functions,  $g^L$ , and  $g^H$ , each having a range of size  $r/2$ . A more formal description follows (where all missing proofs can be found in the appendix).

## 5.2 The Functions $f'$ and $g$

Let  $f : V \rightarrow \{0, \dots, r-1\}$ , where  $V$  is a partially ordered finite domain, and  $\{0, \dots, r-1\}$  is a fully ordered range of size  $r$ , where  $r$  is a power of 2. Recall that  $G_{viol}(f) = (V, E_{viol}(f))$  denotes 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,  $V^L(f) = \{x : f(x) \leq \frac{r}{2} - 1\}$ . Similarly, let  $V^H(f)$  be the “high value” subset of  $V$  with respect to  $f$ . That is,  $V^H(f) = \{x : f(x) \geq \frac{r}{2}\}$ . Let  $f' : V \rightarrow \{0, 1\}$  be defined as follows:

$$\begin{aligned} \text{if } x \in V^L(f) \text{ then } f'(x) &= 0 \\ \text{if } x \in V^H(f) \text{ then } f'(x) &= 1 \end{aligned} \tag{28}$$

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  $g : V \rightarrow \{0, \dots, r-1\}$  be defined as follows:

$$\begin{aligned} \text{if } x \in VC(G_{viol}(f')) \text{ and } x \in V^H(f) \text{ then } g(x) &= \frac{r}{2} - 1 \\ \text{if } x \in VC(G_{viol}(f')) \text{ and } x \in V^L(f) \text{ then } g(x) &= \frac{r}{2} \end{aligned}$$

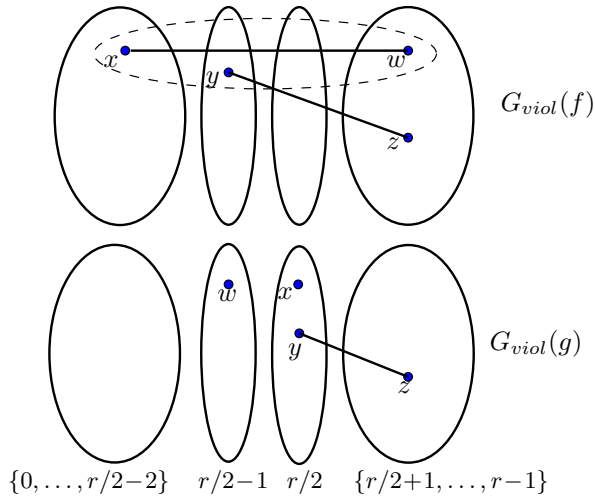


Figure 3: An illustration of the violation graph of the given function  $f$ , and the generated function  $g$ . The domain is partitioned into 4 subsets, according to the labels of the function. The location of elements  $x, y, w$  and  $z$  corresponds to their label according to  $f$  and  $g$  respectively. That is,  $f(x) \leq \frac{r}{2} - 2$ ,  $f(y) = \frac{r}{2} - 1$ ,  $f(w) \geq \frac{r}{2} + 1$ , and  $f(z) \geq \frac{r}{2} + 1$ , while  $g(x) = \frac{r}{2}$ ,  $g(y) = \frac{r}{2}$ ,  $g(w) = \frac{r}{2} - 1$ , and  $g(z) = f(z) \geq \frac{r}{2} + 1$ . The dotted ellipse denotes a minimum vertex cover  $VC(G_{viol}(f))$  in  $G_{viol}(f)$  (i.e.,  $x, y, w \in VC(G_{viol}(f))$ ). Recall that edges in the violation graph denote violations of monotonicity (e.g.  $x > w$  while  $f(x) < f(w)$ ).

$$\text{if } x \notin VC(G_{viol}(f')) \text{ then } g(x) = f(x) \quad (29)$$

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

**Lemma 5.1** *For the function  $g$  as defined in Equation (29),*

1. *There are no edges in the violation graph of  $g$ , connecting vertices in  $V^L(g)$  to vertices in  $V^H(g)$ . That is,  $\{(x, y) \in E_{viol}(g) : x \in V^L(g) \text{ and } y \in V^H(g)\} = \emptyset$ .*
2.  $E_{viol}(g) \subseteq E_{viol}(f)$

**Proof:**

Part 1. Let  $x, y$  be a pair of elements such that  $x < y$ . We show that either  $(x, y) \notin E_{viol}(g)$ , or  $x$  and  $y$  both belong to either  $V^L(g)$  or to  $V^H(g)$ . We proceed with a case analysis.

Consider first the case that both  $x$  and  $y$  belong to  $V^L(f)$ . The case that  $x, y \in V^H(f)$  is analyzed analogously. We first observe that if  $y \in VC(G_{viol}(f'))$  then  $x \in VC(G_{viol}(f'))$  as well. Assume, contrary to the claim, that  $x \notin VC(G_{viol}(f'))$ . If  $y \in VC(G_{viol}(f'))$  then there must be some  $z < y$  such that  $z \notin VC(G_{viol}(f'))$  and  $z \in V^H(f)$ . But then, since  $x < y < z$  and  $x \in V^L(f)$ , the pair  $(x, z)$  is an edge in  $E_{viol}(f')$  that is not covered by  $VC(G_{viol}(f'))$ , and we reach a contradiction. Hence there are four subcases to consider.

1. If  $x, y \notin VC(G_{viol}(f'))$  then by definition of  $g$ ,  $g(x) = f(x)$  and  $g(y) = f(y)$  so that  $x, y \in V^L(g)$ .
2. If  $x, y \in VC(G_{viol}(f'))$ , then  $g(x) = g(y) = \frac{r}{2}$  and so  $(x, y) \notin E_{viol}(g)$ .

3. If  $x \in VC(G_{viol}(f'))$  and  $y \notin VC(G_{viol}(f'))$ , then  $g(x) = \frac{r}{2}$  while  $g(y) = f(y) > \frac{r}{2}$  and so  $(x, y) \notin E_{viol}(g)$ .

We turn to the case that  $x \in V^L(f)$  and  $y \in V^H(f)$ . In this case,  $(x, y) \in E_{viol}(f')$  and so either  $x$  or  $y$  (or both) belong to  $VC(G_{viol}(f'))$ , and there are three subcases to consider.

1. If  $x \in VC(G_{viol}(f'))$  and  $y \notin VC(G_{viol}(f'))$  then  $x, y \in V^H(g)$ .
2. If  $x \notin VC(G_{viol}(f'))$  and  $y \in VC(G_{viol}(f'))$  then  $x, y \in V^L(g)$ .
3. If  $x, y \in VC(G_{viol}(f'))$  then  $g(x) > g(y)$  and so  $(x, y) \notin E_{viol}(g)$ .

Finally we deal with the case that  $x \in V^H(f)$  and  $y \in V^L(f)$ . In this case we claim that it is not possible that both  $x$  and  $y$  belong to  $VC(G_{viol}(f'))$ . Indeed, if  $y \in VC(G_{viol}(f'))$  then there exists a  $z < y$ ,  $z \notin VC(G_{viol}(f'))$  and  $z \in V^H(f)$ . Similarly, if  $x \in VC(G_{viol}(f'))$  then there exists a  $w > x$ ,  $w \notin VC(G_{viol}(f'))$  and  $w \in V^L(f)$ . But then the edge  $(w, z) \in E_{viol}(f')$  is not covered by  $VC(G_{viol}(f'))$ , and we reach a contradiction. Therefore, we need to consider three subcases.

1. If  $x, y \notin VC(G_{viol}(f'))$  then  $g(x) = f(x)$  and  $g(y) = f(y)$  so that  $(x, y) \notin E_{viol}(g)$ .
2. If  $x \in VC(G_{viol}(f'))$  and  $y \notin VC(G_{viol}(f'))$ , then  $x, y \in V^L(g)$ .
3. If  $x \notin VC(G_{viol}(f'))$  and  $y \in VC(G_{viol}(f'))$ , then  $x, y \in V^H(g)$ .

**Part 2.** Consider an edge  $(x, y) \in E_{viol}(g)$  where  $g(x) < g(y)$  (so that  $x > y$ ). By the first part of the lemma, either  $x$  and  $y$  are both in  $V^L(g)$ , or they are both in  $V^H(g)$ . Assume that  $x$  and  $y$  are both in  $V^L(g)$ . The case that  $x, y \in V^H(g)$  is analyzed analogously.

First we show that  $x$  must be in  $V^L(f)$  and not in  $VC(G_{viol}(f'))$ . Assume otherwise, i.e.,  $x \in V^H(f)$  or  $x \in VC(G_{viol}(f'))$ . In fact, since  $x \in V^L(g)$  then (under the counter-assumption), it must be the case that  $x \in V^H(f)$  and  $x \in VC(G_{viol}(f'))$ . In this case,  $g(x) = \frac{r}{2} - 1$ , and since  $y \in V^L(g)$  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. If  $y \in V^L(f)$  and  $y \notin VC(G_{viol}(f'))$ , then  $f(y) = g(y)$  and  $(x, y) \in E_{viol}(f)$ .
2. If  $y \in V^H(f)$  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. ■

### 5.3 The Functions $g^L$ and $g^H$

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

$$\begin{aligned} \text{if } x \in V^L(g) & \quad \text{then} & \quad g^L(x) = g(x) \\ \text{if } x \in V^H(g) & \quad \text{then} & \quad g^L(x) = \frac{r}{2} - 1 \end{aligned} \tag{30}$$

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

$$\begin{aligned} \text{if } x \in V^H(g) & \quad \text{then} & \quad g^H(x) = g(x) \\ \text{if } x \in V^L(g) & \quad \text{then} & \quad g^H(x) = \frac{r}{2} \end{aligned} \tag{31}$$

**Lemma 5.2** For  $g^L$  and  $g^H$  as defined in Equations (30) and (31), 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 5.1 implies that  $E_{viol}(g^L) \cup E_{viol}(g^H) = E_{viol}(g)$ . Part 2 of Lemma 5.1 implies that  $E_{viol}(g) \subseteq E_{viol}(f)$  and Part 1 of this lemma follows. Part 2 of this lemma follows from the definition of  $f'$ .

It remains to prove Part 3. Part 1 of Lemma 5.1 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  $g^{mon} : V \rightarrow \{0, \dots, r-1\}$  be a monotone function such that  $\text{dist}(g, g_{mon}) = \epsilon_{mon}(g)$ . Also recall that  $\epsilon_{mon}(f') = \text{dist}(f, g)$ . Therefore,

$$\begin{aligned} \epsilon_{mon}(f) &\leq \text{dist}(f, g_{mon}) \\ &\leq \text{dist}(f, g) + \text{dist}(g, g_{mon}) \\ &= \epsilon_{mon}(f') + \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L) \end{aligned} \tag{32}$$

and Part 3 of the lemma follows. ■

## 5.4 Properties of the Operator $\Gamma$

**Lemma 5.3** Let  $\mathcal{F}$  be the set of all functions that map  $V$  to a finite range. Let  $\Gamma : \mathcal{F} \rightarrow \mathbb{R}$  be an operator that satisfies the following properties for every  $f \in \mathcal{F}$  (where  $f'$ ,  $g^L$ , and  $g^H$  are as defined above with respect to  $f$ ):

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

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

$$\epsilon_{mon}(f) \leq \Gamma(f) \tag{33}$$

then for every  $f : V \rightarrow R$ ,

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

**Proof:** Let  $s = \lceil \log |R| \rceil$ . We prove the lemma by induction on  $s$ . The base of the induction,  $s = 1$  follows directly from Equation (33). Now assume the lemma holds for  $s-1$ , and let  $f : V \rightarrow R$ . Therefore,

$$\epsilon_{mon}(f) \leq \epsilon_{mon}(f') + \epsilon_{mon}(g^H) + \epsilon_{mon}(g^L) \tag{34}$$

$$\leq \Gamma(f') + (s-1) \cdot \Gamma(g^H) + (s-1) \cdot \Gamma(g^L) \tag{35}$$

$$\leq \Gamma(f) + (s-1) \cdot \Gamma(f) \tag{36}$$

$$= s \cdot \Gamma(f) \tag{37}$$

Equation (34) follows from Part 3 of Lemma 5.2. Equation (35) 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  $2^{s-1}$ . Equation (36) is based on the properties of  $\Gamma$ . ■

## 5.5 Proof of Item 2 in Lemma 3.1

Let  $V = [n]^d$  and let  $\Gamma(f) = \Gamma^k(f) = \frac{2d}{k} \cdot \text{Exp}_{q,\alpha,\beta}[\epsilon_{\text{mon}}(f_{q,\alpha,\beta})]$ . We need to show that  $\Gamma$  has the properties stated in Lemma 5.3.

For every  $q, \alpha, \beta$ , let  $E_{\text{viol}}(f_{q,\alpha,\beta})$  denote the set of edges in the violation graph of  $f_{q,\alpha,\beta}$  (i.e.  $G_{\text{viol}}(f_{q,\alpha,\beta}) = (V, E_{\text{viol}}(f_{q,\alpha,\beta}))$ ). Part 2 of Lemma 5.2 implies that  $E_{\text{viol}}(f') \subseteq E_{\text{viol}}(f)$ . Therefore, for every  $q, \alpha, \beta$ , we have that  $E_{\text{viol}}(f'_{q,\alpha,\beta}) \subseteq E_{\text{viol}}(f_{q,\alpha,\beta})$ . which means that  $\epsilon_{\text{mon}}(f'_{q,\alpha,\beta}) \leq \epsilon_{\text{mon}}(f_{q,\alpha,\beta})$ . Therefore Property 1 holds (i.e.,  $\Gamma(f') \leq \Gamma(f)$ ).

Part 1 of Lemma 5.2, and the fact that  $V^H(g) \cap V^L(g) = \emptyset$ , and  $E_{\text{viol}}(g^H) \cap E_{\text{viol}}(g^L) = \emptyset$  imply that for every  $q, \alpha, \beta$ ,

$$|VC(G_{\text{viol}}(g_{q,\alpha,\beta}))| \leq |VC(G_{\text{viol}}(f_{q,\alpha,\beta}))| \quad (38)$$

and

$$\begin{aligned} VC(G_{\text{viol}}(g_{q,\alpha,\beta}^H)) \cup VC(G_{\text{viol}}(g_{q,\alpha,\beta}^L)) &= VC(G_{\text{viol}}(g_{q,\alpha,\beta})) \\ VC(G_{\text{viol}}(g_{q,\alpha,\beta}^H)) \cap VC(G_{\text{viol}}(g_{q,\alpha,\beta}^L)) &= \emptyset. \end{aligned} \quad (39)$$

Hence,

$$\begin{aligned} |VC(G_{\text{viol}}(g_{q,\alpha,\beta}^H))| + |VC(G_{\text{viol}}(g_{q,\alpha,\beta}^L))| &= |VC(G_{\text{viol}}(g_{q,\alpha,\beta}))| \\ &\leq |VC(G_{\text{viol}}(f_{q,\alpha,\beta}))| \end{aligned} \quad (40)$$

Lemma 2.1 implies that for every  $q, \alpha, \beta$  we have

$$\begin{aligned} \epsilon_{\text{mon}}(g_{q,\alpha,\beta}^H) + \epsilon_{\text{mon}}(g_{q,\alpha,\beta}^L) &\leq \epsilon_{\text{mon}}(g_{q,\alpha,\beta}) \\ &\leq \epsilon_{\text{mon}}(f_{q,\alpha,\beta}) \end{aligned} \quad (41)$$

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

The correctness of Item 2 in Lemma 3.1 for the case  $R = \{0, 1\}$  (established in the previous section) implies that for every  $f : [n]^d \rightarrow \{0, 1\}$ ,  $\epsilon_{\text{mon}}(f) \leq \Gamma(f)$ , and the second item in Lemma 3.1 follows.  $\blacksquare$ .

## 6 Algorithms for Low Dimension $k$ and $R = \{0, 1\}$

In this section we describe several algorithms for low dimensions and a Boolean range where each is based on a different idea. Combining each of these algorithms with Algorithm 1 we obtain the different items in Theorem 2. Specifically, the first algorithm is a 2-approximation algorithm that works for any given  $k$  and has exponential dependence on  $k$  (but no dependence on  $n$ ). The second algorithm is a purely additive approximation algorithm for  $k = 2$ , and the third algorithm is a simple purely additive approximation algorithm for  $k = 1$ . What is common to the second and third algorithms is that they essentially *learn* a good approximation of the low-dimensional function. When combined with Algorithm 1 the first algorithm gives the best approximation ratio for sufficiently large  $k$ , and the third has the smallest complexity.

### 6.1 A 2-Approximation Algorithm for any $k \geq 1$

In this subsection we present Algorithm 2 (see Figure 4). Algorithm 2 is given parameters  $0 < \delta \leq 1$ ,  $0 < \gamma \leq 1$  and query access to a function  $h : [n]^k \rightarrow \{0, 1\}$ . It returns a value  $\hat{\epsilon}$  such that:

**Lemma 6.1** *At the end of Algorithm 2, with probability at least  $1 - \gamma$*

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

*The query complexity and running time of the algorithm are*

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

**Proof of Item 1 of Theorem 2.** If we run Algorithm 2 with the additive error parameter set to  $\frac{\delta}{(k+2)}$  and output  $\frac{1}{2} \cdot (\hat{\epsilon} - \delta)$ , then with probability at least  $1 - \gamma$  we obtain a  $(2, \delta)$ -estimate of  $\epsilon_{\text{mon}}(h)$ .

The query complexity and running time are  $\left(O\left(\left(\frac{k}{\delta}\right)^2 \log\left(\frac{k}{\delta}\right)\right)\right)^{k+2} \log(1/\gamma)$ . By Lemma 3.2, if we use this algorithm as a subroutine in Algorithm 1, then we obtain a  $\frac{k}{4d}$ -approximation algorithm for monotonicity of functions  $f : [n]^d \rightarrow \{0, 1\}$ . The query complexity and running time of the combined algorithm are

$$\left(\tilde{O}\left(\frac{k}{\delta}\right)\right)^{2k+6} \quad (42)$$

■

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

**The high-level idea of the algorithm.** Recall that by Corollary 4.2, if  $h : [n]^k \rightarrow \{0, 1\}$ , then  $S_{1,k}[h]$  is monotone. Therefore,  $\text{dist}(h, S_{1,k}[h]) \geq \epsilon_{\text{mon}}(h)$ . On the other hand, by Lemma 4.3,  $\text{dist}(h, S_{1,k}[h]) \leq 2\epsilon_{\text{mon}}(h)$ . Algorithm 2 estimates  $\text{dist}(h, S_{1,k}[h])$  to within a small additive error. It does so by reconstructing the values of  $S_{1,k}[h]$  (with a small margin of error), on uniformly selected points given query access to  $h$ . Details follow (missing details can be found in the appendix).

### 6.1.1 The family $\{h_i : [n]^k \rightarrow \{0, 1, *\}\}_{i=0}^k$

In order to estimate the distance between  $h$  and  $S_{1,k}[h]$  we define a family of functions  $\{h_i : [n]^k \rightarrow \{0, 1, *\}\}_{i=0}^k$  such that for every  $i$ ,  $h_i$  is very close to  $S_{1,i}[h]$ . We show that for every  $i$  and  $y \in [n]^k$ , if  $h_i(y) \neq *$  then  $h_i(y) = S_{1,i}[h](y)$ . We also show that the part of the domain for which the labels of  $h_i$  equal  $*$ , is very small. The order on the set  $\{0, 1, *\}$  is  $0 < * < 1$ .

In what follows, when referring to a triplet  $(i, \alpha, \beta)$  such that  $i \in [k]$ ,  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{k-i}$ , if  $i = 1$  this actually means a pair  $(1, \beta)$  such that  $\beta \in [n]^{k-1}$ , and if  $i = k$  this actually means a pair  $(k, \alpha)$  such that  $\alpha \in [n]^{k-1}$ .

**Definition 5** *For a function  $g : [n]^k \rightarrow \{0, 1\}$ , let*

$$\begin{aligned} O_{i,\alpha,\beta}(g) &= |\{\hat{y} \in [n] : g(\alpha\hat{y}\beta) = 1\}| \\ Z_{i,\alpha,\beta}(g) &= |\{\hat{y} \in [n] : g(\alpha\hat{y}\beta) = 0\}| \end{aligned} \quad (43)$$

*(where  $O$  stands for “Ones” and  $Z$  stands for “Zeros”). Given a function  $h : [n]^k \rightarrow \{0, 1\}$ , let the set of functions  $\{h_i : [n]^k \rightarrow \{0, 1, *\}\}_{i=0}^k$  be defined recursively as follows. For  $i = 0$ , let  $h_0 = h$ .*

For every  $1 \leq i \leq k$ , and for every  $\alpha \in [n]^{i-1}$ ,  $\beta \in [n]^{k-i}$  and  $\acute{y} \in [n]$ ,

$$\begin{aligned}
& \text{if } \acute{y} \leq Z_{i,\alpha,\beta}(h_{i-1}) - \frac{\delta n}{2} \text{ then } h_i(\alpha\acute{y}\beta) = 0 \\
& \text{if } \acute{y} \geq n - O_{i,\alpha,\beta}(h_{i-1}) + 1 + \frac{\delta n}{2} \text{ then } h_i(\alpha\acute{y}\beta) = 1 \\
& \text{else } h_i(\alpha\acute{y}\beta) = *
\end{aligned} \tag{44}$$

The next claim follows directly from Definition 5.

**Claim 6.2** For every  $i \in [k]$ :

1.  $h_i$  is monotone in dimension  $i$ . That is,  $S_i[h_i] = h_i$ .
2. For every  $y \in [n]^k$

$$\text{if } h_i(y) = 0 \text{ then } S_i[h_{i-1}](y) = 0 \tag{45}$$

$$\text{if } h_i(y) = 1 \text{ then } S_i[h_{i-1}](y) = 1 \tag{46}$$

3. For every  $\alpha \in [n]^{i-1}$  and  $\beta \in [n]^{k-i}$

$$|\{\acute{y} \in [n] : h_i(\alpha\acute{y}\beta) = *\}| \leq |\{\acute{y} \in [n] : h_{i-1}(\alpha\acute{y}\beta) = *\}| + \delta n \tag{47}$$

**Lemma 6.3** Consider the family  $\{h_i : [n]^k \rightarrow \{0, 1, *\}\}_{i=0}^k$  introduced in Definition 5.

1. For every  $y \in [n]^k$

$$\begin{aligned}
& \text{if } h_k(y) = 0 \text{ then } S_{1,k}[h](y) = 0 \\
& \text{if } h_k(y) = 1 \text{ then } S_{1,k}[h](y) = 1
\end{aligned} \tag{48}$$

2. For every  $0 \leq i \leq k$ , let  $X_*(h_i) = \{y \in [n]^k : h_i(y) = *\}$ .

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

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

**Induction Base:** Equations (45) and (46) imply that for every  $y \in [n]^k$

$$\begin{aligned}
& \text{if } h_1(y) = 0 \text{ then } S_{1,1}[h](y) = S_1[h_0](y) = 0 \\
& \text{if } h_1(y) = 1 \text{ then } S_{1,1}[h](y) = S_1[h_0](y) = 1
\end{aligned} \tag{49}$$

**Induction Step:** Assume that for some  $1 \leq i \leq k-1$  we have that for every  $y \in [n]^k$

$$\begin{aligned}
& \text{if } h_i(y) = 0 \text{ then } S_{1,i}[h](y) = 0 \\
& \text{if } h_i(y) = 1 \text{ then } S_{1,i}[h](y) = 1
\end{aligned} \tag{50}$$

This means that

$$\begin{aligned}
& \text{if } S_{i+1}[h_i](y) = 0 \text{ then } S_{i+1}[S_{1,i}[h]](y) = S_{1,i+1}[h](y) = 0 \\
& \text{if } S_{i+1}[h_i](y) = 1 \text{ then } S_{i+1}[S_{1,i}[h]](y) = S_{1,i+1}[h](y) = 1
\end{aligned} \tag{51}$$

Equations (45) and (46) imply that

$$\begin{aligned} \text{if } h_{i+1}(y) = 0 & \text{ then } S_{i+1}[h_i](y) = 0 \\ \text{if } h_{i+1}(y) = 1 & \text{ then } S_{i+1}[h_i](y) = 1 \end{aligned} \quad (52)$$

And together we have that for every  $y \in [n]^k$

$$\begin{aligned} \text{if } h_{i+1}(y) = 0 & \text{ then } S_{1,i+1}[h](y) = 0 \\ \text{if } h_{i+1}(y) = 1 & \text{ then } S_{1,i+1}[h](y) = 1 \end{aligned} \quad (53)$$

The first item of the lemma follows.

For the second item of the lemma, for every  $i \in [k]$ , Equation (47) implies that

$$|X_*(h_i)| \leq |X_*(h_{i-1})| + \delta \cdot n^k \quad (54)$$

Recall that  $|X_*(h_0)| = 0$  and therefore for every  $i \in [k]$

$$|X_*(h_i)| \leq \delta \cdot i \cdot n^k \quad (55)$$

and the second item of the lemma follows.  $\blacksquare$

### 6.1.2 The Algorithm

Algorithm 2 appears in Figure 4. As we prove subsequently, the algorithm estimates the distance between  $h_k$  and  $h$ .

**Lemma 6.4** *For every  $y \in [n]^k$ , if  $h_k(y) \neq *$  then with probability at least  $1 - \frac{\delta}{8}$ ,  $\text{Estimate}(k, y) = h_k(y)$ .*

Before we prove Lemma 6.4, we use it to prove that Algorithm 2 is indeed a distance approximation algorithm. Namely, we prove Lemma 6.1.

**Proof of Lemma 6.1.** The query complexity and running time of Algorithm 2 are

$$s^k \cdot m = \left( O\left(\frac{1}{\delta^2} \log(1/\delta)\right) \right)^{k+2} \log(1/\gamma) \quad (56)$$

We turn to prove its correctness.

Let

$$Y_1 = \{y \in [n]^k : h(y) \neq S_{1,k}[h](y)\} \quad (57)$$

Lemma 4.3 implies that

$$\epsilon_{\text{mon}}(h) \cdot n^k \leq |Y_1| \leq 2\epsilon_{\text{mon}}(h) \cdot n^k \quad (58)$$

Let

$$Y_2 = \{y \in [n]^k \setminus X_*(h_k) : h(y) \neq S_{1,k}[h](y)\} \quad (59)$$

**Algorithm 2** (*Distance Approximation for  $k$  Dimensions and  $\{0, 1\}$  Range*)

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 and independently selected elements in  $[n]^k$  (including repetitions).
2. For every  $1 \leq j \leq m$  let  $s_j = \text{Estimate}(k, x^j)$  where the procedure `Estimate` is presented below.
3. return  $\hat{\epsilon} = \frac{1}{m} |\{j \in [m] : s_j \neq h(x^j)\}|$

**Estimate**( $i, y$ ) {

1. if  $i = 0$  return  $h(y)$
2. Let  $s = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\delta}))$ , and let  $S = \{y_j\}_{j=1}^s$  be a multi-set of  $s$  uniformly and independently selected elements in  $[n]$  (including repetitions). For every  $j \in [s]$ , let  $r_j = \text{Estimate}(i - 1, y_1 \cdots y_{i-1} y_j y_{i+1} \cdots y_k)$ .
3. Let  $\tilde{Z} = \frac{n}{s} |\{j \in [s] : r_j = 0\}|$
4. if  $y_i \leq \tilde{Z}$  then return 0.
5. if  $y_i \geq \tilde{Z} + 1$  then return 1.

}

Figure 4: Algorithm 2. The algorithm estimates the distance to monotonicity of a function  $h : [n]^k \rightarrow \{0, 1\}$  by estimating its distance to  $S_{1,k}[h]$ .

Item 2 of Lemma 6.3 implies that

$$|Y_1| - k \cdot \delta n^k \leq |Y_2| \leq |Y_1| \quad (60)$$

Let

$$\hat{Y}_2 = \{x^j \in S \setminus X_*(h_k) : h(x^j) \neq S_{1,k}[h](x^j)\} \quad (61)$$

In all that follows, when we apply the additive Chernoff bound we assume that for both the sample sizes  $s$  and  $m$  defined in the algorithm, the constants in the  $\Theta(\cdot)$  notation are sufficiently large. In particular, By the additive Chernoff bound, with probability at least  $1 - \frac{\gamma}{4}$

$$\left| \frac{|Y_2|}{n^k} - \frac{|\hat{Y}_2|}{m} \right| \leq \frac{\delta}{2} \quad (62)$$

Assume that this is the case. Item 1 of Lemma 6.3 implies also that

$$\hat{Y}_2 = \{x^j \in M \setminus X_*(h_k) : h(x^j) \neq h_k(x^j)\} \quad (63)$$

Lemma 6.4 and the additive Chernoff bound imply that with probability at least  $1 - \frac{\gamma}{2}$ ,

$$|\{j \in [m] : h_k(x^j) \neq * \text{ and } \text{Estimate}(k, x^j) \neq h_k(x^j)\}| \leq \frac{\delta}{4}m \quad (64)$$

Assume that this is the case. Let

$$\hat{Y}_3 = \{x^j \in M \setminus X_*(h_k) : h(x^j) \neq \text{Estimate}(k, x^j)\} \quad (65)$$

Equation (64) implies that

$$|\hat{Y}_2| - |\hat{Y}_3| \leq \frac{\delta}{4}m \quad (66)$$

Let

$$\hat{X} = \{x^j \in M : h_k(x^j) = *\} \quad (67)$$

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

$$\left| \frac{|\hat{X}|}{m} - \frac{|X_*(h_k)|}{n^k} \right| \leq \delta \quad (68)$$

Assume that this is indeed the case. Equation (68) and Item 2 of Lemma 6.3 imply that

$$\frac{|\hat{X}|}{m} \leq (k+1)\delta \quad (69)$$

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

$$\hat{Y}_3 = \{x^j \in M \setminus \hat{X} : h(x^j) \neq \text{Estimate}(k, x^j)\} \quad (70)$$

Now recall that

$$\hat{\epsilon} = \frac{1}{m} |\{x^j \in M : h(x^j) \neq \text{Estimate}(k, x^j)\}| \quad (71)$$

Obviously  $\frac{|\hat{Y}_3|}{m} \leq \hat{\epsilon}$ . This and Equations (69) and (70) imply that

$$\frac{|\hat{Y}_3|}{m} \leq \hat{\epsilon} \leq \frac{|\hat{Y}_3|}{m} + \frac{|\hat{X}|}{m} \leq \frac{|\hat{Y}_3|}{m} + (k+1) \cdot \delta \quad (72)$$

Summing up the probabilities that Equation (62), Equation (64) or Equation (68) are not correct, and using Equations (58), (60), (62), (66) and (72) we have that with probability at least  $1 - \gamma$

$$\epsilon_{mon}(h) - (k+1)\delta \leq \hat{\epsilon} \leq 2\epsilon_{mon}(h) + (k+2)\delta \quad (73)$$

■

**Proof of Lemma 6.4.** We show by induction on  $i$  that for every  $y \in [n]^k$ , if  $h_i(y) \neq *$  then with probability at least  $1 - \frac{\delta}{8}$ ,  $\text{Estimate}(i, y) = h_i(y)$ .

Induction Base: Observe that for  $i = 0$ ,  $\text{Estimate}(0, y) = h(y) = h_0(y)$  for every  $y \in [n]^k$ .

Induction Step: Consider any  $y = y_1 \cdots y_k \in [n]^k$  and an iteration of  $\text{Estimate}(i, y)$ . Denote  $\alpha(y, i) = y_1 \cdots y_{i-1}$  and  $\beta(y, i) = y_{i+1} \cdots y_k$ .

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

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

$$\begin{aligned} & \frac{n}{s} \cdot \left| \left\{ j \in [s] : h_{i-1}(\alpha(y, i) \acute{y}_j \beta(y, i)) \neq * \right. \right. \\ & \quad \left. \left. \text{and } \text{Estimate}(i-1, \alpha(y, i) \acute{y}_j \beta(y, i)) \neq h_{i-1}(\alpha(y, i) \acute{y}_j \beta(y, i)) \right\} \right| \leq \frac{\delta}{4} \cdot n \end{aligned} \quad (74)$$

Assume that this is indeed the case. Recall that

$$\tilde{Z} = \frac{n}{s} \cdot |\{j \in [s] : \text{Estimate}(i-1, \alpha(y, i) \acute{y}_j \beta(y, i)) = 0\}|$$

and let

$$\begin{aligned} \tilde{O} &= n - \tilde{Z} \\ &= \frac{n}{s} \cdot |\{j \in [s] : \text{Estimate}(i-1, \alpha(y, i) \acute{y}_j \beta(y, i)) = 1\}| \end{aligned} \quad (75)$$

Let

$$\begin{aligned} \hat{O} &= \frac{n}{s} \cdot |\{j \in [s] : h_{i-1}(\alpha(y, i) \acute{y}_j \beta(y, i)) = 1\}| \\ \hat{Z} &= \frac{n}{s} \cdot |\{j \in [s] : h_{i-1}(\alpha(y, i) \acute{y}_j \beta(y, i)) = 0\}| \end{aligned} \quad (76)$$

Equation (74) implies that

$$\begin{aligned} \tilde{O} &\geq \hat{O} - \frac{\delta}{4}n \\ \tilde{Z} &\geq \hat{Z} - \frac{\delta}{4}n \end{aligned} \quad (77)$$

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

$$\begin{aligned} \hat{O} &\geq O_{i, \alpha(y, i), \beta(y, i)}(h_{i-1}) - \frac{\delta n}{4} \\ \hat{Z} &\geq Z_{i, \alpha(y, i), \beta(y, i)}(h_{i-1}) - \frac{\delta n}{4} \end{aligned} \quad (78)$$

Observe that by the definition of  $h_i$

$$\begin{aligned} O_{i, \alpha(y, i), \beta(y, i)}(h_{i-1}) - \frac{\delta n}{2} &= |\{\acute{y} \in [n] : h_i(\alpha(y, i) \acute{y} \beta(y, i)) = 1\}| \\ Z_{i, \alpha(y, i), \beta(y, i)}(h_{i-1}) - \frac{\delta n}{2} &= |\{\acute{y} \in [n] : h_i(\alpha(y, i) \acute{y} \beta(y, i)) = 0\}| \end{aligned} \quad (79)$$

Equations (77), (78) and (79) imply that

$$\begin{aligned} \tilde{O} &\geq |\{\acute{y} \in [n] : h_i(\alpha(y, i) \acute{y} \beta(y, i)) = 1\}| \\ \tilde{Z} &\geq |\{\acute{y} \in [n] : h_i(\alpha(y, i) \acute{y} \beta(y, i)) = 0\}| \end{aligned} \quad (80)$$

Summing up the probabilities that Equation (74) or Equation (78) are not correct, and using Equation (80) and the fact that  $h_i$  is sorted in the  $i^{\text{th}}$  dimension, we have that

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

and the lemma follows. ■

## 6.2 A Purely-Additive Approximation Algorithm for $k = 2$

In this subsection we present Algorithm 3 (see Figure 7). It is given query access to a function  $h : [n]^2 \rightarrow \{0, 1\}$  and parameters  $0 < \delta \leq 1$ ,  $0 < \gamma \leq 1$ , and returns  $\hat{\epsilon}$  such that:

**Lemma 6.5** *At the end of Algorithm 3, with probability at least  $1 - \gamma$*

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

*The query complexity and running time of Algorithm 3 are  $O\left(\frac{1}{\delta^4} \log\left(\frac{1}{\delta\gamma}\right)\right)$ .*

Unlike Algorithm 2 (see Figure 4), Algorithm 3 provides a distance approximation with no multiplicative error. Furthermore, it is more efficient than Algorithm 2 (when run with  $k = 2$ ).

**Proof of Item 2 of Theorem 2.** If we run Algorithm 3 with the additive error parameter set to  $\delta/4$  and return  $\hat{\epsilon}' = \hat{\epsilon} - \delta$ , then we obtain a purely additive approximation algorithm as defined in Definition 1. By Lemma 3.2, if we use this slight variant of Algorithm 3 as a subroutine in Algorithm 1 (see Figure 1), then we get a  $d$ -approximation algorithm for functions  $f : [n]^d \rightarrow \{0, 1\}$ . The query complexity and running time of the combined algorithm are  $O\left(\frac{1}{\delta^6} \log\left(\frac{1}{\delta}\right)\right)$ . ■

**The high-level idea of the algorithm.** Since we consider functions over the domain  $[n]^2$ , it is convenient to view them as  $n \times n$  matrices. We first observe that every such function can be approximated by a “coarser” version of the function. Namely, the matrix representing the function can be partitioned into blocks of size  $(\delta n) \times (\delta n)$ . The coarser version, which we refer to as a  $\delta$ -block function, assigns the same value to points that belong to the same block. Thus it is defined by  $(1/\delta)^2$  values. The algorithm tries to find a monotone  $\delta$ -block function that is closest to  $h$ .

### 6.2.1 $\delta$ -block Functions

In this subsection, functions are always over the domain  $[n]^2$  and range  $\{0, 1\}$ , even if we do not specifically state it.

**Definition 6** *Let  $\delta \in (0, 1]$ .*

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

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

- A  $\delta$ -block function  $h$  is a function such that for every  $i_1, i_2, j_1, j_2$ , if there exists a pair  $(i, j)$  such that  $(i_1, j_1) \in \text{block}_\delta(i, j)$  and  $(i_2, j_2) \in \text{block}_\delta(i, j)$ , then  $h(i_1, j_1) = h(i_2, j_2)$ . That is, for each block,  $h$  has the same value on all points in the block.

	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	1	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 5: An illustration of a monotone function  $h : [n]^2 \rightarrow \{0, 1\}$  and a  $\delta$ -block monotone function  $g$ . In this case,  $[n] = \{1, \dots, 15\}$  and  $\delta = 1/5$ . The 0/1 labels in the matrix are the values of  $h$  on the points in the domain. For example,  $h(5, 9) = 0$  and  $h(5, 10) = 1$ . The bold dashed line represents the boundary between the 0 label area and the 1 label area of  $h$ . The bold solid line represents the boundary between the 0 label area and the 1 label area of the monotone  $\delta$ -block function  $g$  (where the two boundaries sometimes overlap).

We show that for every monotone function  $h : [n]^2 \rightarrow \{0, 1\}$ , there exists a monotone  $\delta$ -block function  $g : [n]^2 \rightarrow \{0, 1\}$  such that  $\text{dist}(h, g) \leq 2\delta$ . This implies that for every  $h : [n]^2 \rightarrow \{0, 1\}$  (which is not necessarily monotone), there exists a monotone  $\delta$ -block function  $g$  such that  $\epsilon_{\text{mon}}(h) \leq \text{dist}(h, g) \leq \epsilon_{\text{mon}}(h) + 2\delta$ . We then show how to estimate  $\text{dist}(h, g)$ .

**Lemma 6.6** *Let  $h : [n]^2 \rightarrow \{0, 1\}$  be a monotone function. Define  $g : [n]^2 \rightarrow \{0, 1\}$  as follows. For every  $i, j \in \{1, \dots, \frac{1}{\delta}\}$ :*

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

*The function  $g$  has the following properties:*

1. *Function  $g$  as defined above is a monotone  $\delta$ -block function.*
2.  *$\text{dist}(h, g) \leq 2\delta$ .*

For an illustration of a function  $g$  as define in Lemma 6.6, see Figure 5.

**Proof:**

**Part 1:** Obviously,  $g$  is a  $\delta$ -block function. We need to show that it is also monotone. Consider any pair  $i_1, j_1$  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  $g(i_2, j_2) = g(i_1, j_1) = 1$ .

- Otherwise,  $(i_2, j_2)$  is in a block (name it  $block_2$ ) whose points are all larger then the points in the block of  $(i_1, j_1)$  (name it  $block_1$ ). This means that for every point in  $block_1$ , there is a point in  $block_2$  that is larger. Recall that by the definition of  $g$ , there must be a point in  $block_1$  for which the label of  $h$  is 1. This means that there must be an point in  $block_2$  for which the label of  $h$  is also 1. This means that the labels of  $g$  in  $block_2$  are 1, and therefore  $g(i_2, j_2) = 1$ .

The first part of the lemma follows.

Part 2: It is easy to verify that for every  $i, j$ , if all labels of  $h$  in  $block_\delta(i, j)$  are 0, then all labels of  $g$  in  $block_\delta(i, j)$  are 0, and if all labels of  $h$  in  $block_\delta(i, j)$  are 1, then all labels of  $g$  in  $block_\delta(i, j)$  are 1. Therefore, we need to bound the number of blocks in which the labels of  $h$  are not all 0 and not all 1.

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

- For every  $1 \leq j \leq j_\ell(i)$ , the labels of  $block_\delta(i, j)$  of  $h$  are all 0 (if  $j_\ell(i) = 0$  then there is no such  $j$ ).
- For every  $j_q(i) \leq j \leq 1/\delta$ , the labels of  $block_\delta(i, j)$  of  $h$  are all 1 (if  $j_q(i) = 1/\delta + 1$  then there is no such  $j$ ).
- For every  $j_\ell(i) < j < j_q(i)$  the labels of  $block_\delta(i, j)$  of  $h$  are not all 0 and not all 1 (if  $j_\ell(i) = j_q(i) - 1$  then there is no such  $j$ ).

Since  $h$  is monotone,  $j_\ell$  and  $j_q$  are well defined. For example, in Figure 5,  $j_\ell(2) = 3$  and  $j_q(2) = 6$ . Also, the fact that  $h$  is monotone implies that for every  $i < 1/\delta$  we have that  $j_q(i + 1) \leq j_\ell(i) + 2$ . Therefore, the number of blocks in which the labels of  $h$  are not all 0 and not all 1 is at most

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

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

### 6.2.2 The Algorithm

Consider the dynamic programming procedure BestBF (“Best  $\delta$ -Block Function”) in Figure 6. It is given a  $t \times t$  matrix  $A$  where  $t = 1/\delta$ . For every  $i, j$ ,  $A[i, j]$  holds the number of 1 labels and 0 labels in  $block_\delta(i, j)$  of a function  $\tilde{h} : [n]^2 \rightarrow \{0, 1\}$  (that is close to  $h$ ). It returns the distance between  $\tilde{h}$  and a  $\delta$ -block monotone function that is closest to  $\tilde{h}$ .

It is based on the following recursive principle. For  $t = \frac{1}{\delta}$  and  $i, j \in \{1, \dots, t\}$ :

- Let  $\tilde{h}^{i,j} : \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\} \rightarrow \{0, 1\}$  be such that for every  $x \in \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\}$ ,  $\tilde{h}^{i,j}(x) = \tilde{h}(x)$ .
- Let  $g^{i,j} : \{(i-1) \cdot \delta n + 1, \dots, n\} \times \{1, \dots, j \cdot \delta n\} \rightarrow \{0, 1\}$  be a  $\delta$ -block monotone function that is closest to  $\tilde{h}^{i,j}$ .

For simplicity of notation, let  $|\tilde{h}^{i,j}|$  denote the size of the domain of  $\tilde{h}^{i,j}$ . Consider the following recursive equation (its correctness is explained right after the equation):

$$\begin{aligned} \text{dist}(\tilde{h}^{i,j}, g^{i,j}) \cdot |\tilde{h}^{i,j}| = \\ \min \left\{ \begin{aligned} & \text{dist}(\tilde{h}^{i+1,j}, g^{i+1,j}) \cdot |\tilde{h}^{i+1,j}| + \sum_{q=1}^j A[i, q].\text{ones}, \\ & \text{dist}(\tilde{h}^{i,j-1}, g^{i,j-1}) \cdot |\tilde{h}^{i,j-1}| + \sum_{p=i}^k A[p, j].\text{zeros} \end{aligned} \right\} \end{aligned} \quad (81)$$

**An explanation:**

- If  $g^{i,j}$  assigns 0 to all points in  $block_\delta(i, j)$  then
  - For every  $q \leq j$ ,  $g^{i,j}$  assigns 0 to all points in  $block_\delta(i, q)$ . This implies that the number of labels in  $\bigcup_{q=1}^j block_\delta(i, q)$  on which  $g^{i,j}$  and  $\tilde{h}^{i,j}$  differ is exactly  $\sum_{q=1}^j A[i, q].\text{ones}$ .
  - For every  $p > i$  and  $q \leq j$ ,  $g^{i,j}$  assigns either 0 or 1 to all points in  $block_\delta(p, q)$ . This implies that the labels of  $g^{i,j}$  on these blocks (which are optimal for minimizing  $\text{dist}(\tilde{h}^{i,j}, g^{i,j})$  s.t.  $g$  is  $\delta$ -block and monotone), are equal to the labels of  $g^{i+1,j}$  on these blocks.
- If  $g^{i,j}$  assigns 1 to all points in  $block_\delta(i, j)$  then
  - For every  $p \geq i$ ,  $g^{i,j}$  assign 1 to all points in  $block_\delta(p, j)$ . This implies that the number of points in  $\bigcup_{p=i}^k block_\delta(p, j)$  on which  $g^{i,j}$  and  $\tilde{h}^{i,j}$  differ is exactly  $\sum_{p=i}^k A[p, j].\text{zeros}$ .
  - For every  $p \geq i$  and  $q < j$ ,  $g^{i,j}$  may assign either 0 or 1 to the points in  $block_\delta(p, q)$ . This implies that the labels of  $g^{i,j}$  on these blocks (which are optimal for minimizing  $\text{dist}(\tilde{h}^{i,j}, g^{i,j})$  s.t.  $g$  is  $\delta$ -block and monotone), are equal to the labels of  $g^{i,j-1}$  on these blocks.

Procedure BestSB calculates  $D[i, j] = \text{dist}(\tilde{h}^{i,j}, g^{i,j}) \cdot |\tilde{h}^{i,j}|$  starting from the last row to the first and from the first column to the last. Observe that  $\tilde{h}^{1,t} = \tilde{h}$  and  $\tilde{g}^{1,t} = \tilde{g}$ , and so  $D[i, j] = \text{dist}(\tilde{h}, \tilde{g}) \cdot n^2$ . The time complexity of BestBF is  $O(t^2) = O(\frac{1}{\delta^2})$ . Algorithm 3, which appears in Figure 7, uses the procedure as a subroutine.

**Proof of Lemma 6.5.** Let  $p_{ij}$  be the fraction of points in  $block_\delta(i, j)$  that are labeled 1 by  $h$ . Note that  $p_{ij} \cdot \delta^2 \cdot n^2$  is the exact number of points labeled 1 in  $block_\delta(i, j)$ . For every  $i, j \in \{1, \dots, \frac{1}{\delta}\}$ , by the additive Chernoff bound, with probability at least  $1 - \gamma \cdot \delta^2$ ,

$$\left| \frac{1}{m} \sum_{r=1}^m X_r^{i,j} - p_{ij} \right| \leq \delta \quad (82)$$

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

$$\sum_{i=1}^{1/\delta} \sum_{j=1}^{1/\delta} \left| \tilde{A}[i, j].\text{ones} - p_{ij} \delta^2 n^2 \right| \leq \delta n^2. \quad (83)$$

**Procedure BestBF**( $A, \delta, n$ )

For  $t = 1/\delta$ ,  $A$  is a  $t \times t$  matrix such that for some function  $\tilde{h} : [n]^2 \rightarrow \{0, 1\}$ ,

- $A[i, j].ones$  = the number of ones in  $block_\delta(i, j)$  of  $\tilde{h}$ .
- $A[i, j].zeros$  = the number of zeros in  $block_\delta(i, j)$  of  $\tilde{h}$ .

1. Let  $O$  and  $Z$  be  $t \times t$  matrices such that

- For every  $i$  and  $j$ ,  $O[i, j] = \sum_{q=1}^j A[i, q].ones$
- For every  $i$  and  $j$ ,  $Z[i, j] = \sum_{q=i}^t A[q, j].zeros$

2. Let  $D$  be a  $(t+1) \times (t+1)$  matrix whose rows are labeled from 1 to  $t+1$  and whose columns are labeled from 0 to  $t$ . The initial value of each entry in  $D$  is 0.

3. for  $i = t$  down-to 1 and for  $j = 1$  to  $t$

- $D[i, j] = \min\{D[i+1, j] + O[i, j], D[i, j-1] + Z[i, j]\}$

4. Let  $\hat{\epsilon} = D[1, t]/n^2$ . Return  $\hat{\epsilon}$ .

Figure 6: Procedure BestBF. The procedure is given as input a  $t \times t$  matrix  $A$  where  $t = 1/\delta$  and  $A[i, j]$  holds the number of zeros and the number of ones in  $block_\delta(i, j)$  of some function  $\tilde{h} : [n]^2 \rightarrow \{0, 1\}$ . It returns the distance between  $\tilde{h}$  and a monotone  $\delta$ -block function that is closest to  $\tilde{h}$ .

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

Let  $\tilde{h} : [n]^2 \rightarrow \{0, 1\}$  be a function that is closest to  $h$  such that for every  $i, j \in \{1, \dots, 1/\delta\}$ , the number of 1 labels in  $block_\delta(i, j)$  of  $\tilde{h}$  is exactly  $\tilde{A}[i, j].ones$ , and the number of 0 labels is  $\tilde{A}[i, j].zeros$ . Therefore,  $\text{dist}(h, \tilde{h}) \leq \delta$ . Let  $\tilde{g}$  be a  $\delta$ -block monotone function such that  $\text{dist}(\tilde{h}, \tilde{g}) = \text{BestBF}(\tilde{A}) = \hat{\epsilon}$ . Then,

$$\begin{aligned}
\epsilon_{mon}(h) &\leq \text{dist}(f, \tilde{g}) \\
&\leq \text{dist}(f, \tilde{h}) + \text{dist}(\tilde{h}, \tilde{g}) \\
&\leq \delta + \text{dist}(\tilde{h}, \tilde{g}) \\
&= \delta + \hat{\epsilon}
\end{aligned} \tag{84}$$

Let  $\tilde{h}_{mon}$  be a monotone function that is closest to  $\tilde{h}$ . That is,  $\epsilon_{mon}(\tilde{h}) = \text{dist}(\tilde{h}, \tilde{h}_{mon})$ . Lemma 6.6 implies that there is a  $\delta$ -block monotone function, name it  $\tilde{g}_{mon}$ , such that  $\text{dist}(\tilde{h}_{mon}, \tilde{g}_{mon}) \leq 2\delta$ . Recall that  $\tilde{g}$  is a  $\delta$ -block monotone function that is closest to  $\tilde{h}$ . Therefore,

$$\begin{aligned}
\text{dist}(\tilde{h}, \tilde{g}) &\leq \text{dist}(\tilde{h}, \tilde{g}_{mon}) \\
&\leq \text{dist}(\tilde{h}, \tilde{h}_{mon}) + \text{dist}(\tilde{h}_{mon}, \tilde{g}_{mon}) \\
&\leq \epsilon_{mon}(\tilde{h}) + 2\delta
\end{aligned} \tag{85}$$

**Algorithm 3** (*Distance Approximation for Two Dimensions and  $\{0, 1\}$  Range*)

1. Let  $t = \frac{1}{\delta}$  and  $m = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\delta^2 \gamma}))$
2. for  $i = 1$  to  $t$  and  $j = 1$  to  $t$ 
  - (a) uniformly and independently select  $m$  points in  $\text{block}_\delta(i, j)$ . Let  $X_r^{i,j} = h(i', j')$ , where  $i', j'$  are the coordinates of the  $r^{\text{th}}$  selected point in the sample.
  - (b)  $\tilde{A}[i, j].\text{ones} = \frac{1}{m} \cdot \delta^2 n^2 \cdot \sum_{r=1}^m X_r^{i,j}$ .
  - (c)  $\tilde{A}[i, j].\text{zeros} = \delta^2 n^2 - \tilde{A}[i, j].\text{ones}$
3. Let  $\hat{\epsilon} = \text{BestBF}(\tilde{A})$
4. return  $\hat{\epsilon}$

Figure 7: Algorithm 3 for approximating the distance to monotonicity of two-dimensional Boolean functions. The algorithm is based on the dynamic programming procedure BestBF.

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

$$\hat{\epsilon} = \text{dist}(\tilde{h}, \tilde{g}) \leq \delta + \epsilon_{\text{mon}}(h) + 2\delta \quad (86)$$

and finally, equations (84) and (86) imply that

$$\epsilon_{\text{mon}}(h) - \delta \leq \hat{\epsilon} \leq \epsilon_{\text{mon}}(h) + 3\delta \quad (87)$$

■

### 6.3 A Purely-Additive Approximation Algorithm for $k = 1$

For the special case of  $k = 1$ , that is, for functions  $h : [n] \rightarrow \{0, 1\}$ , Algorithm 2 is a  $(2, \delta)$ -estimate algorithm with query complexity and running time 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 a better result.

When the function is one-dimensional, if the function is monotone, then there exists a *switch-point*  $i \in [n]$  such that for every  $j < i$ ,  $h(j) = 0$ , and for every  $j \geq i$ ,  $h(j) = 1$ . This means that for any given function  $h$ , there is an index (switch-point)  $1 \leq i \leq n + 1$  such that

$$\epsilon_{\text{mon}}(h) \cdot n = |\{j : (j < i \text{ and } h(j) = 1) \text{ or } (j \geq i \text{ and } h(j) = 0)\}|. \quad (88)$$

The algorithm in this subsection is based on the idea of estimating this switch-point.

Algorithm 4 is given query access to a function  $h : [n] \rightarrow \{0, 1\}$  and parameters  $0 < \delta \leq 1$ ,  $0 < \gamma \leq 1$ , and returns  $\hat{\epsilon}$  such that:

**Lemma 6.7** *At the end of Algorithm 4, with probability larger than  $1 - \gamma$ ,*

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

*The query complexity and running time of Algorithm 4 are  $\Theta(\frac{1}{\delta^2} \log(\frac{1}{\gamma \delta}))$ .*

**Algorithm 4** (*Distance Approximation for one-dimensional Boolean functions*)

1. Uniformly and independently select  $m = \Theta(\frac{1}{\delta^2} \log(\frac{1}{\gamma \cdot \delta}))$  points in  $[n]$ .
2. 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  $\delta n$ .
3. Let  $a_j$  be the  $j^{\text{th}}$  sample point. For each  $i \in \text{switch-points}$  let  $X_{i,j} = 1$  if  $a_j < i$  and  $h(a_j) = 1$  or  $a_j \geq i$  and  $h(a_j) = 0$ . Otherwise let  $X_{i,j} = 0$ . Let  $\hat{b}_i = \frac{1}{m} \sum_{j=1}^m X_{i,j}$ . It is not hard to verify that it is possible to compute all  $\hat{b}_i$  in time linear in  $m$ .
4. Let  $t \in \text{switch-points}$  satisfy  $\hat{b}_t = \min_i \{\hat{b}_i\}$
5. Let  $\hat{\epsilon} = \hat{b}_t - \frac{\delta}{2}$ . Return  $\hat{\epsilon}$ .

Figure 8: Algorithm 4 for approximating the distance to monotonicity of one-dimensional Boolean functions.

**Proof of Item 3 of Theorem 2.** If we run Algorithm 4 with the additive error parameter set to  $\delta/2$  and return  $\hat{\epsilon}' = \hat{\epsilon} - \delta/2$ , then we obtain a purely additive approximation algorithm as defined in Definition 1. By Lemma 3.2, if we use this slight variant of Algorithm 4 as a subroutine in Algorithm 1, then we get a  $2d$ -approximation algorithm for functions  $f : [n]^d \rightarrow \{0, 1\}$ . The query complexity and running time of the algorithm are  $O(\frac{1}{\delta^4} \log(\frac{1}{\delta}))$ . ■

**Proof of Lemma 6.7.** For every  $i$  and  $\ell$  let  $Y_{i,\ell} = 1$  if  $\ell < i$  and  $h(\ell) = 1$  or  $\ell \geq i$  and  $h(\ell) = 0$ . 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  $h$  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  $1 - \gamma \cdot \delta$

$$b_i - \frac{\delta}{4} \leq \hat{b}_i \leq b_i + \frac{\delta}{4} \quad (89)$$

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

$$s - \frac{\delta \cdot n}{2} \leq i \leq s + \frac{\delta \cdot n}{2} \quad (90)$$

and therefore

$$b_s \leq b_i \leq b_s + \frac{\delta}{2} \quad (91)$$

We also know that  $\hat{b}_k \leq \hat{b}_i$ . Therefore Equation (89) implies that  $b_k \leq b_i + \frac{\delta}{2}$ . Using Equation (91) and the fact that  $b_s \leq b_k$ , we get that

$$b_s \leq b_k \leq b_s + \delta \quad (92)$$

Using Equation (89) again

$$b_s - \frac{\delta}{4} \leq \hat{b}_k \leq b_s + \frac{5\delta}{4} \quad (93)$$

and therefore

$$\epsilon_{\text{mon}}(h) - \delta \leq \hat{\epsilon} \leq \epsilon_{\text{mon}}(h) + \delta \tag{94}$$

■

## References

- [AC06] N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimensions. *Information and Computation*, 204:1704–1717, 2006.
- [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<sup>+</sup>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<sup>+</sup>00] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000.
- [Fat06] S. Fattal. Approximating the distance to monotonicity and convexity. Master’s thesis, School of Electrical Engineering, 2006.
- [FF05] E. Fischer and L. Fortnow. Tolerant versus intolerant testing for boolean properties. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, pages 135–140, 2005.
- [Fis04] E. Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1), 2004.
- [FLN<sup>+</sup>02] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samrodnitsky. 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.

- [FR07] S. Fattal and D. Ron. Approximating the distance to convexity. Available from [www.eng.tau.ac.il/~danar/papers.html](http://www.eng.tau.ac.il/~danar/papers.html), 2007.
- [GGL<sup>+</sup>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.
- [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 *Proceedings of the Ninth International Workshop on Randomization and Computation (RANDOM)*, 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)*, pages 475–486, 2006.
- [PRR06] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.
- [RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

## A Chernoff Bounds

Let  $\chi_1, \dots, \chi_m$  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

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