Lecture Notes on Testing Monotonicity

Oded Goldreich[*]

April 7, 2016

Summary: For each n, we consider functions from a partially ordered set D_n to a totally ordered set R_n. Such a function $f : D_n \to R_n$ is called monotone if for every $x < y$ in D_n it holds that $f(x) \leq f(y)$, where $<$ denotes the partial order of D_n and $\leq$ refers to the total order in R_n. We shall focus on two special cases:

1. Boolean functions on the Boolean Hypercube: In this case, D_n is the ℓ-dimensional Boolean hypercube (with the natural partial order), where $\ell = \log_2 n$, and $R_n = \{0,1\}$. According to this partial order, $x_1 \cdots x_\ell \leq y_1 \cdots y_\ell$ if and only if $x_i \leq y_i$ for every $i \in [\ell]$.
2. Real functions on the discrete line: In this case, $D_n = [n]$ and $R_n = \mathbb{R}$, both with the natural total order.

We shall later consider also the case of the hypergrid domain $D_n = [m]^\ell$, for any $m, \ell \in \mathbb{N}$ such that $m^\ell = n$, and general ranges R_n. In all these cases, we obtain property testers of complexity $\text{poly}(\epsilon^{-1} \log n)$.

In addition, we briefly survey relatively recent developments as well as known results regarding testing convexity, submodularity, and the Lipschitz property of functions from $[m]^\ell$ to $\mathbb{R}$.

These notes are based on Goldreich *et al.* [11] (for case 1), Ergun *et al.* [7] (for case 2), and Dodis *et al.* [6] (for their "interpoltaion").

Notation: The Hamming weight of a sequence $x \in [m]^\ell$, denoted $\text{wt}(x)$, is the number of locations that hold a non-zero value; that is, $\text{wt}(x) = |\{i \in [|x|] : x_i \neq 0\}|$.

1 Introduction

Leaving the land of algebraic functions behind us, we find the notion of a monotone function most appealing. The definition of this notion presumes a partial order on the domain of the function and a total order on its range. We say that $f : D \to R$ is monotone if, for every $x, y \in D$ if $x < y$ (according to the partial order on D), then $f(x) \leq f(y)$ (according to the order on R).

The most natural partially ordered domains are the total order on the "line" $[n] = \{1, 2, ..., n\}$ and the partial order on the hypercube $\{0,1\}^\ell$. Interpolating these two case, we consider the

[*]Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.

partial order on the hypergrid $[m]^\ell$, where $(x_1, ..., x_m) < (y_1, ..., y_m)$ if $x_i \leq y_i$ for all $i \in [m]$ and $(x_1, ..., x_m) \neq (y_1, ..., y_m)$.

We shall consider testing monotonicity in all these cases, both when the range is Boolean and when it is arbitrary. In all cases, we shall consider pair tests, which are (non-adaptive) two-query proximity-oblivious testers (POTs) that, when given orcale access to $f : D \to R$, select a pair $(x, y) \in D^2$ such that $x < y$ and accept if and only if $f(x) \leq f(y)$. The focus will be on choosing a distribution on these pairs, and analyzing the detection probability of the resulting POT.

Organization. In Section 2 we consider the case of Boolean functions defined on the Boolean Hypercube $\{0,1\}^\ell$. Its core is Section 2.1, which provides a detailed analysis of a simple tester. An alternative tester is reviewed in Section 2.2, but this part is merely an overview of advanced material that is only meant for optional reading. In Section 3 we study the case of multi-valued functions on the discrete line $[n]$; the core of this section is Section 3.1, whereas Section 3.2 presents additional results that are not used elsewhere in this chapter. Lastly, in Section 4, we consider multi-valued functions on the hybpergrid $[m]^\ell$, where generalizes the previous two cases.

2 Boolean functions on the Boolean Hypercube

We consider Boolean functions of the form $f : \{0,1\}^\ell \to \{0,1\}$. Such a function f is called monotone if for every $x < y$ in $\{0,1\}^\ell$ it holds that $f(x) \leq f(y)$, where $x_1 \cdots x_\ell \leq y_1 \cdots y_\ell$ if and only if $x_i \leq y_i$ for every $i \in [\ell]$ (and, indeed, $x_1 \cdots x_\ell = y_1 \cdots y_\ell$ if and only if $x_i = y_i$ for every $i \in [\ell]$).

It is instructive to think of $\{0,1\}^\ell$ (with the above partial order) as a directed version of the Boolean hypercube. The Boolean hypercube of dimension ℓ is a graph with vertex set $\{0,1\}^\ell$ and edge set $\{\{u, v\} : \text{wt}(v \oplus u) = 1\}$; that is, u is adjacent to v if and only if they differ on a single bit. In the directed version, which we consider, the edge $\{u, v\}$ is directed from the vertex of smaller Hamming weight to the vertex with higher (by 1) weight.

2.1 The edge test

We show that the natural algorithm that selects uniformly an edge of the Boolean hypercube and compares the values of the function at its end-points constitutes a relatively good proximity-oblivious tester. Such an edge corresponds to a pair (x, y) such that $x < y$ and x differs from y in a single bit (i.e, $\text{wt}(x \oplus y) = 1$), and the algorithm accepts if and only if $f(x) \leq f(y)$. Specifically, we refer to the following algorithm.

Algorithm 1 (testing whether $f : \{0,1\}^\ell \to \{0,1\}$ is monotone): *Select uniformly $v \in \{0,1\}^\ell$ and $i \in [\ell]$, query f at v and $v \oplus 0^{i-1}10^{\ell-i}$, and accept if and only if f is monotone on this pair; that is, letting $\{x, y\} = \{v, v \oplus 0^{i-1}10^{\ell-i}\}$ such that $x < y$, the algorithm accepts if and only if $f(x) \leq f(y)$.*

Let Π_n denote the set of monotone of Boolean functions over $\{0,1\}^\ell$, where $n = 2^\ell$.

Theorem 2 (Algorithm 1 is a POT for monotonicity): *Algorithm 1 is a* (one-sided error) *proximity oblivious tester for Π_n with detection probability δ/ℓ, where δ denotes the distance of the given function from being monotone.*

We comment that this analysis of Algorithm 1 is asymptotically tight in a strong sense: for every $\alpha \in (\exp(-\Omega(\ell)), 0.5)$, there exists a function $f : \{0,1\}^\ell \to \{0,1\}$ that is at distance $\delta \in [\alpha, 2\alpha]$ from being monotone such that Algorithm 1 rejects f with probability $2\delta/\ell$ (see [11, Prop. 4, Part 1]).

Proof: Algorithm 1 accepts each monotone function with probability 1, since the set of all possible executions check conditions that are a subsets of the local conditions used in the definition of monotonicity (i.e., the edges (u, v) are a subset of the set of all pairs (x, y) such that $x < y$).[1] The point, however, is showing that if $f : \{0,1\}^\ell \to \{0,1\}$ is at distance δ from being monotone, then it is rejected with probability at least δ/ℓ. We shall prove the counter-positive. That is, assuming that f is accepted with probability $1 - \rho$, we shall show that f is $(\ell \cdot \rho)$-close to being monotone.

We shall show that f can be made monotone by modifying its values on at most $\rho\ell \cdot 2^\ell$ points. We shall proceed in iterations such that in the i^{th} iteration we make f "monotone in the i^{th} direction", while preserving its monotonicity in the prior directions.

Definition 2.1 (monotonicity in direction i): *Let $f : \{0,1\}^\ell \to \{0,1\}$ and $i \in [\ell]$. We say that f* is monotone in direction i *if for every $v' \in \{0,1\}^{i-1}$ and $v'' \in \{0,1\}^{\ell-i}$ it holds that $f(v'0v'') \le f(v'1v'')$.*

We make f monotone in direction i by applying a corresponding "switching operator", denoted S_i, which maps Boolean functions to Boolean functions as follows.

Definition 2.2 (switch in direction i): *For every $i \in [\ell]$, the switch operator S_i is defined such that for every function $f : \{0,1\}^\ell \to \{0,1\}$ the function $S_i(f) : \{0,1\}^\ell \to \{0,1\}$ is monotone in direction i and satisfies $\{S_i(f)(v'0v''), S_i(f)(v'1v'')\} = \{f(v'0v''), f(v'1v'')\}$ for every $v' \in \{0,1\}^{i-1}$ and $v'' \in \{0,1\}^{\ell-i}$.* (Indeed, $S_i(f)(x)$ denotes the value of the function $S_i(f)$ at the point x.)

That is, for every $v' \in \{0,1\}^{i-1}$ and $v'' \in \{0,1\}^{\ell-i}$ if $f(v'0v'') \le f(v'1v'')$ then $(S_i(f)(v'0v''), S_i(f)(v'1v'')) = (f(v'0v''), f(v'1v''))$, and otherwise $(S_i(f)(v'0v''), S_i(f)(v'1v'')) = (f(v'1v''), f(v'0v''))$. Either way, it holds that $S_i(f)(v'0v'') \le S_i(f)(v'1v'')$.

Now, assuming that f is accepted with probability $1 - \rho$, we shall consider the sequence of functions $f_0, ..., f_\ell$ such that $f_0 = f$ and $f_i = S_i(f_{i-1})$ for every $i \in [\ell]$. We shall show that f_ℓ is monotone and that $\sum_{i \in [\ell]} \delta(f_i, f_{i-1}) \le \ell \cdot \rho$, where $\delta(g, h) = \mathbf{Pr}_x[g(x) \ne h(x)]$ is the standard distance between functions. The fact that $f_i = S_i(f_{i-1})$ is monotone in direction i follows by the definition of the switch operator, whereas the fact that f_i preserves the monotonicity of f_{i-1} in each direction $j < i$ needs to be proved. This will follow as a special case of a general claim that will also allow us to establish $\sum_{i \in [\ell]} \delta(f_i, f_{i-1}) \le \ell \cdot \rho$. Towards this claim, we shall need the following definition.

Definition 2.3 (violation in direction i): *Let $g : \{0,1\}^\ell \to \{0,1\}$ and $i \in [\ell]$. For $v' \in \{0,1\}^{i-1}$ and $v'' \in \{0,1\}^{\ell-i}$, the directed edge $(v'0v'', v'1v'')$* is a violating edge of g in direction i *if $g(v'0v'') > g(v'1v'')$. We denote by $V_i(g)$ the set of violating edges of g in direction i.*

Clearly, g is monotone in direction i if and only if it has no violating edges in direction i (i.e., $V_i(g) = \emptyset$). We are now ready to state our main claim.

[1]**Advanced comment:** Actually, the only functions that are accepted by Algorithm 1 with probability 1 are monotone, since the subset of local conditions checked by the algorithm impose all the local conditions in the definition. To see this, consider, for every $x < y$, a (shortest) directed path (in the hypercube), denoted $x^{(0)} = x, x^{(1)}, ..., x^{(t)} = y$, leading from x to y, and use $f(x^{(0)}) \le f(x^{(1)}) \le \cdots \le f(x^{(t)})$. Indeed, w.l.o.g., $x^{(i)} = x_1 \cdots x_j y_{j+1} \cdots y_\ell$, where j is the location of the i^{th} non-zero bit in $x \oplus y$. (See the related Exercise 1.)

Claim 2.4 (the effect of the switch operator on the set of violations): *Let $g : \{0,1\}^\ell \to \{0,1\}$ and $i, j \in [\ell]$. Then, $|V_j(S_i(g))| \le |V_j(g)|$.*

It follows that if g is monotone in direction j, then so is $S_i(g)$. The fact that $\sum_{i\in[\ell]} \delta(f_i, f_{i-1}) \le \ell\rho$ will follow by using two additional observations (see Facts 1 and 2 below).[2]

Proof: The case of $i = j$ is trivial (since $V_i(S_i(g)) = \emptyset$), and so, we consider $i \ne j$. For sake of notational simplicity and without loss of generality, we may consider the case of $i = 1$ and $j = 2$. The key observation is that the analysis of the effect of S_i on the violations of g in direction j can be reduced to the effects on the violations of the functions obtained (from g) by all possible restrictions of the other $\ell - 2$ coordinates. That is, for every $u \in \{0,1\}^{\ell-2}$, we consider the residual function $g_u(\sigma\tau) = g(\sigma\tau u)$, and observe that $V_2(g) = \cup_{u\in\{0,1\}^{\ell-2}} V_2(g_u)$ (and $V_2(S_1(g)) = \cup_{u\in\{0,1\}^{\ell-2}} V_2(S_1(g_u))$).[3] Hence, it suffices to prove that $|V_2(S_1(g_u))| \le |V_2(g_u)|$ *holds for every u.* This can be verified by a case analysis, but it is instructive to make a picture.

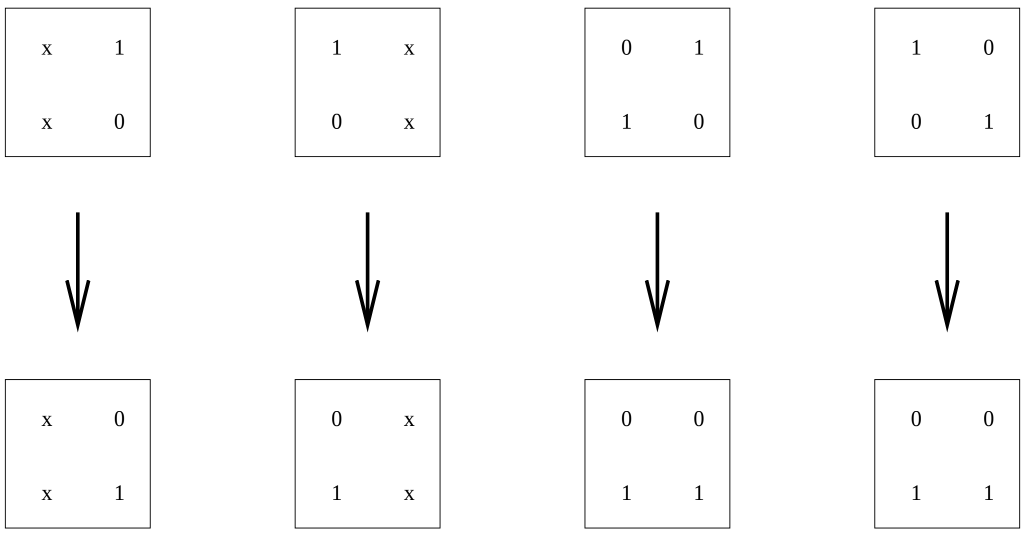

Figure 1: The remaining four cases in the proof of Claim 2.4. In the first two (leftmost) cases the sorted column equals $(xx)^\top$, whereas in the other two cases the sorted column is $(01)^\top$.

Pictorially, consider a 2-by-2 Boolean matrix M such that the (σ, τ)-entry corresponds to $g_u(\sigma, \tau)$. The foregoing claim (i.e., $|V_2(S_1(g_u))| \le |V_2(g_u)|$) asserts that *if we sort the columns*

[2]Specifically, we shall show that $\sum_{i\in[\ell]} |V_i(f)| = \rho \cdot \ell \cdot 2^{\ell-1}$ (Fact 1) and $\delta(f_i, f_{i-1}) = 2^{-(\ell-1)} \cdot |V_i(f_{i-1})|$ (Fact 2). Combining these facts with Claim 2.4, it will follows that

$$\begin{aligned}\sum_{i\in[\ell]} \delta(f_i, f_{i-1}) &= 2^{-(\ell-1)} \cdot \sum_{i\in[\ell]} |V_i(f_{i-1})| \\ &\le 2^{-(\ell-1)} \cdot \sum_{i\in[\ell]} |V_i(f)| \\ &= \rho \cdot \ell\end{aligned}$$

as claimed.

[3]This is because the question of whether the edge $(\sigma 0u, \sigma 1u)$ is violating depends only on the values at its endpoints, whereas S_1 satisfies $\{S_1(g)(0\tau u), S_1(g)(1\tau u)\} = \{g(0\tau u), g(1\tau u)\}$. Hence, the contribution of the edges $(00u, 01u)$ and $(10u, 11u)$ to $V_2(g)$ and $V_2(S_1(g))$ depend only on the values of g and $S_1(g)$ on $00u, 01u, 10u$ and $11u$.

of M, then the number of unsorted rows may only decrease. The only cases worthy of consideration are those in which at least one of the columns of M is unsorted, since otherwise sorting the columns has no effect. Now, if both columns are unsorted, then they are both equal to the vector $(10)^\top$, and sorting the columns only means permuting the rows, which means that the number of violations (which is zero) is preserved. We are left with four cases, depicted in Figure 1, in which exactly one column is sorted. In the first two cases (where the sorted column is monochromatic), sorting the columns means permuting the rows (which again preserves the number of violations). In the other two cases (where the sorted column is $(01)^\top$), sorting the columns means eliminating all violations (since the resulting columns will both equal $(01)^\top$). ■

By repeated applications of Claim 2.4, we obtain

Corollary 2.5 (on the sets of violations in the sequence of f_i's):

1. *For every $i \in [\ell]$, the function f_i is monotone in each direction $j \le i$. In particular, f_ℓ is monotone.*
2. *For every $i, j \in [\ell]$, it holds that $|V_j(f_i)| \le |V_j(f)|$.*

Proof: Recalling that $f_i = S_i(f_{i-1})$ and applying Claim 2.4 (with $g = f_{i-1}$), we get that, for every $i, j \in [\ell]$, it holds that $|V_j(f_i)| \le |V_j(f_{i-1})|$. Hence, for every $j \in [\ell]$ and $0 \le i_1 < i_2 \le \ell$, it holds that $|V_j(f_{i_2})| \le |V_j(f_{i_1})|$. Now, Item 1 follows because $|V_j(f_i)| \le |V_j(f_j)| = 0$ (for every $j \le i$), whereas Item 2 follows because $|V_j(f_i)| \le |V_j(f_0)|$ (for every $i > 0$).[4] ■

We now establish the two facts mentioned above (i.e., right after Claim 2.4):[5]

Fact 1: $2 \cdot \sum_{i\in[\ell]} |V_i(f)| = \rho \cdot \ell 2^\ell$.

This follows since the choice (v, i) makes Algorithm 1 rejects f if and only if the (directed version of the) edge $\{v, v \oplus 0^{i-1}10^{\ell-i}\}$ is violating (for f in direction i).[6] (Indeed, each such violating edge $(v'0v'', v'1v'')$ contributes to two choices (i.e., to $(v'0v'', |v'| + 1)$ and $(v'1v'', |v'| + 1)$).)

Fact 2: $2^\ell \cdot \delta(f_i, f_{i-1}) = 2 \cdot |V_i(f_{i-1})|$.

This is a special case of $2^\ell \cdot \delta(S_i(g), g) = 2 \cdot |V_i(g)|$, which holds because $S_i(g)(x) \ne g(x)$ if and only if the (directed version of the) edge $\{x, x \oplus 0^{i-1}10^{\ell-i}\}$ is violating (for g in direction i).[7] (Indeed, each such violating edge contributes twice to $\delta(f_i, f_{i-1})$.)

[4] Recall that $f_0 = f$.

[5] See also Footnote 2, which also provides a preview of their use.

[6] Formally,

$$\begin{aligned}\rho &= \mathbf{Pr}_{(v,i)\in\{0,1\}^\ell\times[\ell]}\left[f(v_{[i-1]}0v_{[i+1,\ell]}) > f(v_{[i-1]}1v_{[i+1,\ell]})\right]\\ &= \frac{1}{\ell}\cdot\sum_{i\in[\ell]}\mathbf{Pr}_{(v',v'')\in\{0,1\}^{i-1}\times\{0,1\}^{\ell-i}}\left[f(v'0v'') > f(v'1v'')\right]\\ &= \frac{1}{\ell}\cdot\sum_{i\in[\ell]}\frac{|V_i(f)|}{2^{\ell-1}}\end{aligned}$$

where the second equality uses the fact that the value of the bit v_i is irrelevant to the event being analyzed.

[7] Formally, $\delta(S_i(g), g) = \mathbf{Pr}_{x\in\{0,1\}^\ell}[S_i(g)(x) \ne g(x)]$, which equals the probability that x is an endpoint of an edge in $V_i(g)$, which in tuen equals $2 \cdot |V_i(g)|/2^\ell$.

By combining these two facts with Item 2 of Corollary 2.5 we get

$$\begin{aligned}\delta(f,f_\ell) &\leq \sum_{i\in[\ell]}\delta(f_{i-1},f_i)\\ &= \sum_{i\in[\ell]}2^{-(\ell-1)}\cdot|V_i(f_{i-1})|\\ &\leq 2^{-(\ell-1)}\cdot\sum_{i\in[\ell]}|V_i(f)|\\ &= \ell\cdot\rho\end{aligned}$$

where the first equality follows by Fact 2, the second inequality follows by Item 2 of Corollary 2.5, and the second equality follows by Fact 1. Recalling that (by Item 1 of Corollary 2.5) the function f_ℓ is monotone, we conclude that f is $\ell\rho$-close to monotone. The theorem follows. ■

Digest. The proof of Theorem 2 shows that the *absolute* distance of f from being monotone, denoted $\Delta_{\mathtt{M}}(f)$, is upper-bounded by twice the number of violating edges (of f). Denoting the latter set by $V(f)$, it is tempting to think that $\Delta_{\mathtt{M}}(f) \leq |V(f)|$, since each violation can be corrected by modifying one endpoint of the edge, but this ignores the possibility that the correction of one violation may cause other violations. Indeed, in the proof of Theorem 2, we performed modifications with more care: We proceed in iterations such that in the i^{th} iteration, we eliminate a subset of violations in f_i, denoted $V_i(f_i)$, while making sure that the number of violations in the resulting function, denoted f_{i+1}, does not exceed $|V(f_i)\setminus V_i(f_i)|$. We stress that $V(f_{i+1})$ is not necessarily a subset of $V(f_i)\setminus V_i(f_i)$, yet $|V(f_{i+1})| \leq |V(f_i)\setminus V_i(f_i)|$. Recall that *the set of violations $V_i(f_i)$, which constitutes a matching, is not eliminated by modifying f_i at one endpoint of each edge, but rather by switching the pair of values at the endpoints of each edge.* (Thus, $|\{x : f_i(x) \neq f_{i+1}\}| = 2\cdot|V_i(f_i)|$, rather than half this amount.) This "wasteful" method of modifying f_i enables proving that $|V(f_{i+1})| \leq |V(f_i)\setminus V_i(f_i)|$, and it follows that $\Delta_{\mathtt{M}}(f) \leq 2\cdot|V(f)|$.

On the tightness of the analysis. Recall that the *relative* distance of f from being monotone, denoted $\delta_{\mathtt{M}}(f)$, equals $\Delta_{\mathtt{M}}(f)/2^\ell$, whereas the probability that Algorithm 1 rejects f, denoted $\rho(f)$, equals $\frac{|V(f)|}{\ell\cdot 2^\ell/2}$. Hence, $\Delta_{\mathtt{M}}(f) \leq 2\cdot|V(f)|$ translates to $\delta_{\mathtt{M}}(f) \leq 2^{-\ell+1}\cdot|V(f)| = \ell\cdot\rho(f)$. As stated upfront, the upper bound $\delta_{\mathtt{M}}(f) = O(\ell\cdot\rho(f))$ is tight: For every $\alpha \in (\exp(-\Omega(\ell)), 0.5)$, there exists a function $f : \{0,1\}^\ell \to \{0,1\}$ such that $\delta_{\mathtt{M}}(f) \in [\alpha, 2\alpha]$ and $\delta_{\mathtt{M}}(f) = \Omega(\ell\cdot\rho(f))$ (see [11, Prop. 4, Part 1]). For example, for $f(x) = 1 - x_1$ it holds that $\delta_{\mathtt{M}}(f) = 0.5$ and $\rho(f) = 1/\ell$.

On the other hand, $\delta_{\mathtt{M}}(f) = \Omega(\ell\cdot\rho(f))$ does not hold for all f's: For every $\alpha \in (\exp(-\Omega(\ell)), 0.5)$, there exists a function $f : \{0,1\}^\ell \to \{0,1\}$ such that $\delta_{\mathtt{M}}(f) \in [\alpha, 2\alpha]$ and $\delta_{\mathtt{M}}(f) = \Theta(\rho(f))$ (see [11, Prop. 4, Part 2]). For example, for $f(x) = \mathrm{wt}(x) \bmod 2$ it holds that $\delta_{\mathtt{M}}(f) \approx 0.5$ and $\rho(f) \approx 0.5$ (see Exercise 2).

2.2 Path tests

The fact that the analysis of the rejection probability of Algorithm 1 is tight (i.e., there are non-monotone functions f that this algorithm rejects with probability $O(\delta_{\mathtt{M}}(f)/\ell)$), does not mean that one cannot do better, even when using two-query tests. Algorithm 1 checks the values at the endpoints of a uniformly selected edge of the hypercube, which seems the most natural thing to do. Indeed, this is the best choice for tests that examines the values at the endpoints of an edge selected according to any distribution.[8]

Of course, there is no reason to restrict two-query testers to examine the values at the endpoints of an edge of the hypercube. Indeed, without loss of generality, the two queries made by the test must be comparable (or else it makes no sense to compare the answers), but these two queries may reside on the endpoints of a path of (almost) arbitrary length. Also, for the purpose of ϵ-testing, little is lost when restricting the random path to having both endpoints be strings of Hamming weight in $[(\ell/2) \pm O(\sqrt{\ell \log(1/\epsilon)})]$, since vertices with Hamming weight that deviates from this interval occupy at most a 0.1ϵ fraction of the hyper-cube. (Also, little is lost by restricting the tester to be non-adaptive.)[9]

To see the benefit of this generalization, consider the function $f(x) = 1 - x_1$, which is 0.5-far from being monotone. While the edge test rejects this function with probability $1/\ell$, a tester that examines the endpoints of a random path of length $\sqrt{\ell}$ (which starts at a uniformly distributed vertex) rejects this function with probability $1/\sqrt{\ell}$. It turns out that rejection probability $\widetilde{\Theta}(1/\sqrt{\ell})$ is achievable and is optimal for two-query $\Omega(1)$-testers. On the other hand, the rejection probability of such optimal two-query testers cannot be linear in the distance of the function from monotonicity; actually, if, for some function $F : \mathbb{N} \to \mathbb{N}$, a two-query proximity-oblivious tester (with one-sided error) rejects f with probability $\delta_{\mathtt{M}}(f)/F(\ell)$, then $F(\ell) = \Omega(\ell/\log \ell)$ (cf. [1]).

Following is a description of a generic "path tester": In light of the foregoing, this tester selects a "random path" (i.e., a pair of comparable vertices) such that each of its endpoints is almost uniformly distributed. This is done by selecting the first vertex, denoted u, uniformly, and selecting the second vertex, denoted v, uniformly among all vertices that are at distance d from u, where the distance is selected according to some distribution, denoted $\mathcal{D}_\ell$. One specific suggestion that works well (see Theorem 4) is to have $\mathcal{D}_\ell$ be uniform over the set $\{2^i : i \in \{0, 1, ..., \lfloor \log \ell \rfloor\}\}$.

Algorithm 3 (the generic path test, parameterized by a distribution $\mathcal{D}_\ell$ over $[\ell]$):

1. *Select comparable $u, v \in \{0,1\}^\ell$ by the following process. First, select u uniformly in $\{0,1\}^\ell$, and then select $d \leftarrow \mathcal{D}_\ell$ and $j \in \{-1, +1\}$ uniformly. Now, select v uniformly among all ℓ-bit long strings of weight $\mathrm{wt}(u) + j \cdot d$ that are comparable to u (i.e., either $u < v$ or $u > v$, depending on j).[10] Specifically, if $j = 1$, then v is selected uniformly in $\{z > u : \mathrm{wt}(z) = \mathrm{wt}(u) + d\}$, else v is selected uniformly in $\{z < u : \mathrm{wt}(z) = \mathrm{wt}(u) - d\}$. Indeed, if $\mathrm{wt}(u) + j \cdot d \notin \{0, 1, ..., \ell\}$, then no vertex v is selected and the algorithm halts accepting.*

2. *Query f at u and v and accept if and only if f is monotone on this pair; that is, letting $\{x, y\} = \{v, u\}$ such that $x < y$, the algorithm accepts if and only if $f(x) \le f(y)$.*

Indeed, having $\mathcal{D}_\ell \equiv 1$ corresponds to the edge test of Algorithm 1. We now consider two alternative choices for the distribution $\mathcal{D}_\ell$:

[8]Consider such an algorithm and let $i \in [\ell]$ denote the direction that is selected with the lowest probability, where the direction of an edge $\{u, v\}$ is the coordinate on which u and v differ. Then, the function $f(x) = 1 - x_i$ is rejected with probability at most $1/\ell$, while it is 0.5-far from being monotone.

[9]See Exercise **??**.

[10]Indeed, in this case the vertex v is not uniformly distributed among u's neighbors, altghough typically (i.e., when $\mathrm{wt}(u) \approx \ell/2$) this is approximately the case. To have v be distributed uniformly among u's neighbors, one should select $j = -1$ with probility $\mathrm{wt}(u)/\ell$.

The path tester: One natural choice is to have $\mathcal{D}_\ell$ be distributed as $|\text{wt}(U_\ell) - (\ell/2)|$, where U_ℓ is uniformly distributed in $\{0,1\}^\ell$. That is, $\mathcal{D}_\ell$ represents the devaition from $\ell/2$ of the Hamming weight of a uniformly distributed ℓ-bit long string.[11] In this case, $\mathcal{D}_\ell$ resides in $[\Theta(\sqrt{\ell})]$ with constant probability, and equals 1 with probability $\Theta(1/\sqrt{\ell})$. Hence, the corresponding tester (which typically uses long paths) is called the pure path tester.

The combined path and edge tester: In contrast, letting $\mathcal{D}_\ell$ be uniform over the set $\{2^i : i \in \{0,1,...,\lfloor \log \ell \rfloor\}\}$ yields a distribution in which both the values 1 and $2^{0.5\lfloor \log \ell \rfloor} \approx \sqrt{\ell}$ occur with probability $1/\log \ell$. Hence, the corresponding tester (i.e., that uses this "skewed" $\mathcal{D}_\ell$) is called the combined path and edge tester.

While an exact analysis of the pure path test is still unknown, an almost exact analysis of the combined path and edge test is known.

Theorem 4 (the combined edge and path test [13]): *Algorithm 3, with $\mathcal{D}_\ell$ that is uniform over $\{2^i : i \in \{0,1,...,\lfloor \log \ell \rfloor\}\}$, constitutes a* (one-sided error) *proximity oblivious tester with detection probability $\widetilde{\Omega}(\delta^2/\sqrt{\ell})$, where δ denotes the distance of the given function from being monotone.*

Note that the detection probability bound provided by Theorem 4 is quadratic in δ and linear in $1/\sqrt{\ell}$, whereas the bound in Theorem 2 is linear in both δ and $1/\ell$. Indeed, the point of Theorem 4 is obtaining an improved performance in terms of ℓ; in fact, this improved performance is optimal (up to polylogarithmic factors).[12] It is conjectured that the pure path test also achieves the bound stated in Theorem 4. More generally, we pose the following question.

Open Problem 5 (which path testers are best for constant δ?): *Which choices of the distribution $\mathcal{D}_\ell$ preserve the result of Theorem 4?*

3 Multi-valued functions on the discrete line

Here we consider multi-valued functions of the form $f : [n] \to R_n$, where R_n is an arbitrary totally ordered set (e.g., any subset of the real numbers). Such a function f is called monotone if for every $x < y$ in $[n]$ it holds that $f(x) \leq f(y)$. Recall that a special case of this problem, where $R_n = \{0,1\}$, was presented in the first lecture.

3.1 A tester based on binary search

It will be instructive to view the values of $f : [n] \to R_n$ as residing in an array of n cells and to assume that all values of f are distinct (i.e., $|\{f(i) : i \in [n]\}| = n$). Consider the following tester for monotonicity that selects $i \in [n]$ uniformly at random, and then tries to find the value $f(i)$ in the said array by conducting a binary search. If f is indeed (strictly) monotone, then this search will succeed in finding $f(i)$ in location i. Hence, this (binary-search) tester performs $1 + \lceil \log_2 n \rceil$ queries, and accepts if and only if $f(i)$ is found in this binary search. (In order to waive the requirement that f has distinct values, we augment $f(i)$ to $((f(i), i)$ when comparing values of f, while using the lexicographic order on pairs.)[13]

[11]**Advanced comment:** A related alternative is to have $\mathcal{D}_\ell$ be uniform over $[O(\sqrt{\ell \log(1/\epsilon)})]$.

[12]**Advanced comment:** More generally, if a pair tester has detection probability $\Omega(\delta^b/\ell^a)$, then $2a + b \geq 3$ (see [13]). Hence, both Theorems 2 and 4 meet this lower bound, at $(a,b) = (1,1)$ and $(a,b) = (0.5, 2)$, respectively.

[13]That is, instead of comparing $f(i)$ to $f(j)$, we compare $(f(i), i)$ to $(f(j), j)$ and say that $(f(i), i)$ is (strictly) smaller than $(f(j), j)$ if either $f(i) < f(j)$ or both $f(i) = f(j)$ and $i < j$ hold.

As noted above, this tester always accepts monotone functions, and the point is lower-bounding the rejection probability of the tester as a function of the distance of f from being monotone. We shall show that if $f : [n] \to R_n$ is δ-far from monotone, then the foregoing tester rejects it with probability greater than δ. We shall actually prove the counter-positive.

Claim 6 (on the rejection probability of the binary search tester): *If the binary search tester accepts $f : [n] \to R_n$ with probability $1 - \delta$, then f is δ-close to monotone.*

Teaching note: The foregoing tester as well as the following proof are presented in a somewhat loose style, since we shall later provide a more rigorous presentation and analysis of a related tester (see Algorithm 7 and its analysis). In fact, the reader may skip the following proof and proceed directly to Algorithm 7 (and later derive a proof of Claim 6 by minor modifications to the proof of Lemma 8).

Proof: Note that the only *random choice* performed by the tester is the choice of $i \in [n]$ made at its very first step. We call $i \in [n]$ good if an execution that starts with choosing i is completed with acceptance. For simplicity, we assume that all values in f are distinct or alternatively consider an execution in which f is replaced by f' such that $f'(i) = (f(i), i)$.

We first claim that if $i < j$ are both good, then $f(i) < f(j)$. To prove this claim, we consider the pair of binary searches conducted for $f(i)$ and for $f(j)$. Since both i and j are good, the first binary search ended at location i and the second binary search ended at j. Let $t \in [\lceil \log_2 n \rceil]$ be the first step in which these two binary searchers took different choices when comparing the "sought for" value against a "pivot" value associated with location p_t (which is at the end of the first half of the currently eligible interval).[14] Since the two searches took different halves (of the current interval) and ended at i and j, respectively, the search for $f(i)$ went to the first half whereas the search for $f(j)$ went to the second half. But due to the comparisons made at this step, it follows that $f(i) \leq f(p_t)$ and $f(j) > f(p_t)$. Hence, $f(i) < f(j)$, as claimed.

Finally, we observe that the restriction of f to the set of good points yields a monotone function. Hence, by modifying f on the non-good points, we obtain a monotone function over $[n]$. Recalling that there are $(1 - \delta) \cdot n$ good points, the claim follows. ■

A related tester. The foregoing tester was presented as if the tester is adaptive. Specifically, after selecting a random $i \in [n]$, the tester takes choices that supposedly depend on the values of f that it obtains. However, a closer look reveals that the correct choices (of which half interval to take) can be determined a priori (by the value of i), and if the examined values of f do not match these choices, then it is safe to reject immediately. This observation leads to the following non-adaptive tester, where the sequence of intervals and pivot points is determined a priori in Step 2.

Algorithm 7 (testing whether $f : [n] \to R_n$ is monotone): *Let $\ell = \lceil \log_2 n \rceil$ and $[a_0, b_0] = [1, n]$.*

1. *Uniformly select $i \in [n]$.*

[14]The binary search for the value v starts with the eligible interval $[1, n]$. In the first step, the pivot location $p_1 = \lceil (n+1)/2 \rceil$ is used, and the search takes the half interval $[1, p_1]$ if and only if $v \leq f(p_1)$; otherwise, the search takes $[p_1 + 1, n]$. The description of subsequent steps is analogous: The pivot of the interval $[a, b]$ is $p = \lceil (a+b)/2 \rceil$, and the search takes $[a, p]$ if and only if $v \leq f(p)$.

2. For $t = 1, ..., \ell$, let $p_t = \lceil (a_{t-1} + b_{t-1})/2 \rceil$ and

$$[a_t, b_t] = \begin{cases} [a_{t-1}, p_t] & \textit{if } i \leq p_t p_t + 1, b_{t-1} > p_t + 1, b_{t-1} \\ & \textit{otherwise} \end{cases}$$

[redacted]

3. Query f at i as well as at $p_1, ..., p_\ell$.

4. For $t = 1, ..., \ell$, if $i \leq p_t$ and $f(i) > f(p_t)$, then reject. Likewise, for $t = 1, ..., \ell$, if $i > p_t$ and $f(i) < f(p_t)$, then reject.

If the algorithm did not reject in Step 4, then it accepts.

Algorithm 7 performs $1 + \lceil \log_2 n \rceil$ queries and always accepts a monotone function. To complete its analysis we show that if $f : [n] \to R_n$ is δ-far from being monotone, then Algorithm 7 rejects it with probability greater than δ.

Lemma 8 (on the rejection probability of Algorithm 7): *If Algorithm 7 accepts $f : [n] \to R_n$ with probability $1 - \delta$, then f is δ-close to monotone.*

The proof is analogous to the proof of Claim 6, but it is more rigorous due to the more detailed description of the algorithm, which facilitates clear references to its steps. (The main clarification is in the second paragraph of the proof.)

Proof: Note that the only random choice performed by Algorithm 7 is the choice of $i \in [n]$ made in Step 1, and call $i \in [n]$ good if an execution that starts with choosing i is completed with acceptance.

We first claim that if $i < j$ are *both good*, then $f(i) \le f(j)$. Let $t \in [\ell]$ be the smallest integer for which the t^{th} interval (i.e., $[a_t, b_t]$) determined (in Step 2) for i is different from the t^{th} interval determined for j. It follows that p_t is assigned the same value in both executions, but exactly one element in $\{i, j\}$ took the first half. Therefore, exactly one of these two elements is smaller or equal to p_t, and since $i < j$, it must be that $i \le p_t$ and $j > p_t$. Now, by the corresponding part of Step 4, it follows that $f(i) \le f(p_t)$ and $f(j) \ge f(p_t)$, or else the corresponding execution would have rejected (in contradiction to the hypothesis that both i and j are good). Hence, $f(i) \le f(j)$, as claimed.

Denoting the set of good choices by G, we observe that the restriction of f to G yields a monotone function. Hence, by modifying f only on points in $[n] \setminus G$, we obtain a monotone function over $[n]$. (For example, we can modify f at $i \in [n] \setminus G$ such that $f(i) = f(j)$, where j is the smallest element in G that is greater than i, and if no such element exists we set $f(i)$ to equal the largest element in R_n.) Using $|G| = (1 - \delta) \cdot n$, the lemma follows. ■

Corollaries. Let Π_n denote the set of monotone of functions with domain $[n]$ and range R_n. Then, by Lemma 8, we have –

Theorem 9 (Algorithm 7 is a POT for monotonicity): *Algorithm 7 is a* (one-sided error) $(1 + \lceil \log_2 n \rceil)$*-query proximity oblivious tester for Π_n with detection probability δ, where δ denotes the distance of the given function from being monotone.*

Observing that Algorithm 7 rejects if and only if at least one of the checks of Step 4 rejects, we obtain a two-query POT with detection probability δ/ℓ. Specifically, we refer to a version of Algorithm 7 in which Steps 3 and 4 are replaced by selecting $t \in [\ell]$ uniformly at random, and comparing $f(i)$ to $f(p_t)$; that is, the test rejects if and only if either $i \le p_t$ and $f(i) > f(p_t)$ or $i > p_t$ and $f(i) < f(p_t)$.

Theorem 10 (a two-query POT for monotonicity): *The foregoing algorithm is a* (one-sided error) *two-query proximity oblivious tester for Π_n with detection probability δ/ℓ, where δ denotes the distance of the given function from being monotone and $\ell = \lceil \log_2 n \rceil$.*

Proof: Using the terminology of Lemma 8, we observe that if i is not good (w.r.t Algorithm 7), then the two-query algorithm rejects with probability at least $1/\ell$ (since at least one of the ℓ relevant checks fails). On the other hand, by Lemma 8, a function that is at distance δ from Π_n must have at least $\delta \cdot n$ points $i \in [n]$ that are not good. ■

3.2 Other testers

Theorem 10 presents a two-query POT with detection probability $\delta/\lceil \log_2 n \rceil$ for monotone functions over $[n]$ (i.e., for the property Π_n). An alternative proof of a similar lower bound follows as a special case of the following result.

Theorem 11 (general analysis of two-query POTs for monotonicity): *Let $G = ([n], E)$ be a connected multi-graph such that for every $1 \le i < j \le n$ either $\{i,j\} \in E$ or there exists $k \in (i,j)$ such that $\{i,k\},\{k,j\} \in E$. Consider an algorithm that selects an edge $\{i,j\} \in E$ uniformly at random, and accepts if and only if $f(i) \le f(j)$, where $i < j$. Then, this algorithm constitutes a* (one-sided error) *two-query proximity oblivious tester for Π_n with detection probability $\delta \cdot n/2|E|$, where δ denotes the distance of the given function from being monotone.*

Theorem 10 follows as a special case by noting that the $n \cdot \ell$ pairs of possible queries (of the corresponding two-query version of Algorithm 7) define a graph that satisfies the hypothesis of Theorem 11, since every $i < j$ are connected via p_t (for an adequate t). (We also mention that the algorithm that compares the values at random pair of points $(i,j) \in [n]^2$ is a POT with detection probability δ/n.)[15]

Proof: Fix $f \notin \Pi_n$ and let δ denotes the distance of f from Π_n. We say that a pair $(i,j) \in [n]^2$ such that $i < j$ is a violation if $f(i) > f(j)$. Viewing the set of violating edges as a graph, denoted G^f, we observe that G^f has no vertex cover of size smaller than δn, since the restriction of f to any independent set is a monotone function (and so f can be made monotone by modifying its value at the vertices of the vertex cover).[16] It follows that G^f has a matching of size at least $\delta n/2$; in fact, each maximal matching in G^f must have such a size (or else we obtain a vertex cover of size smaller than δn).

Note, however, that this matching, denoted M^f, need not be a subset of E, since M^f is a matching in the ("violation") graph G^f and E is the edge-set of the ("query") graph G. Nevertheless, for each $\{i,j\} \in M^f \setminus E$ such that $i < j$ there exists $k \in \{i+1, ..., j-1\}$ such that $\{i,k\},\{k,j\} \in E$. It follows that either $f(i) > f(k)$ or $f(k) > f(j)$, or else $f(i) \le f(k) \le f(j)$ in contradiction to the hypothesis that the pair (i,j) is a violation. In other words, each violating pair in M^f yields a violating pair in E, and the latter pairs are distinct since M^f is a matching. Hence, the tester rejects with probability at least $\frac{|M^f|}{|E|} > \frac{\delta n}{2|E|}$. ■

[15] The analysis of this naive tester is asymptotically optimal: Consider, for example, the function $f : [n] \to [n]$ such that $f(i) = 2 \cdot \lceil i/2 \rceil + (i \bmod 2) - 1 \in \{2\lceil i/2 \rceil - 1, 2\lceil i/2 \rceil\}$, which is at distance 0.5 from being monotone, but has only $n/2$ violating pairs (and hence is rejected with $\frac{n/2}{\binom{n}{2}} \approx 1/n$). It is even easier to see that the analysis of the two-query POT referred to by Theorem 10 is asymptotically optimal: Consider, for example, the function $f : [n] \to \{0,1\}$ such that $f(i) = 1$ if and only if $i < n/2$.

[16] Indeed, the same observation is implicit in the proof of Lemma 8. See Exercise 3.

Comments. It turns out that a graph satisfying the hypothesis of Theorem 11 must have $\Omega(n \log n)$ edges. (See [15] for a proof as well as a wider perspective.) On the other hand, some two-query POTs for Π_n are not covered by Theorem 11: For example, an algorithm that selects $i \in [n-1]$ uniformly and accepts if and only if $f(i) \leq f(i+1)$ rejects each $f \notin \Pi_n$ with probability at least $1/(n-1)$. Finally, recall that in the special case of Boolean functions (i.e., $R_n = \{0,1\}$), we have seen (in the first lecture) a two-query POT with detection probability $\Omega(\delta^2)$.

4 Multi-valued functions on the Hypergrid

Generalizing the previous cases, we now consider the case of $D_n = [m]^\ell$, for any $m, \ell \in \mathbb{N}$ such that $m^\ell = n$, and general R_n. (Indeed, in Section 2 we had $m = 2$ and $R_n = \{0,1\}$, whereas in Section 3 we had $m = n$.) That is, we consider functions of the form $f : [m]^\ell \to R_n$. Such a function f is called monotone if for every $x < y$ in $[m]^\ell$ it holds that $f(x) \leq f(y)$, where $x = x_1 \cdots x_\ell < y = y_1 \cdots y_\ell$ if and only if $x_i \leq y_i$ for every $i \in [\ell]$ (and, indeed, $x_1 \cdots x_\ell = y_1 \cdots y_\ell$ if and only if $x_i = y_i$ for every $i \in [\ell]$).

It turns out that testing monotonicity in this case reduces to testing monotonicity in the one dimensional case. This is based on the observation that $f : [m]^\ell \to R_n$ is monotone if and only if f is monotone in each direction (i.e., if and only if for every $\alpha \in [m]^{i-1}$ and $\beta \in [m]^{\ell-i}$ the function $f'(z) = f(\alpha z \beta)$ is monotone in z).[17] Hence, we consider the following algorithmic schema.

Algorithm 12 (testing whether $f : [m]^\ell \to R_n$ is monotone):

1. *Select uniformly $i \in [\ell]$, as well as $\alpha \in [m]^{i-1}$ and $\beta \in [m]^{\ell-i}$.*

2. *Invoke a monotonicity tester for functions from $[m]$ to R_n, while providing it with oracle access to the function f' such that $f'(z) = f(\alpha z \beta)$.*

Algorithm 12 preserves the query complexity of the tester used in Step 2. Also, by the foregoing characterization, it follows that if a one-sided error tester is used in Step 2, then Algorithm 12 has one-sided error. The analysis of the rejection probability of this testing schema combines two reductions (which refer only to two-query POTs). The first reduction refers only to Boolean functions, and it lower-bounds the rejection probability of the schema in terms of the rejection probability of the (two-query) tester used in Step 2.

Lemma 13 (dimension reduction for the Boolean case): *Let T be a two-query POT for monotonicity of Boolean functions over $[m]$ that selects pairs according to distribution $\mathcal{D}$ and accepts $h : [m] \to \{0,1\}$ if and only if $h(x) \leq h(y)$, where $(x,y) \leftarrow \mathcal{D}$. Let ϱ_m denotes the detection probability function of T; that is, if h is at distance δ from a monotone Boolean function, then $\mathbf{Pr}[T^h(m) = 0] = \mathbf{Pr}_{(x,y)\leftarrow\mathcal{D}}[h(x) > h(y)] \geq \varrho_m(\delta)$. Suppose that ϱ_m is convex. Then, using T in Step 2 of Algorithm 12 yields a two-query POT for monotonicity of Boolean functions over $[m]^\ell$ with detection probability function $\varrho(\delta) = \varrho_m(\delta/2\ell)$; that is, if $g : [m]^\ell \to \{0,1\}$ is at distance δ from a monotone Boolean function, then the algorithm rejects it with probability at least $\varrho_m(\delta/2\ell)$.*

In particular, using the POT of Theorem 10 in Step 2, we obtain a POT for monotone Boolean functions over $[m]^\ell$ such that functions that are at distance δ from monotone are rejected with probability at least $\frac{\delta/\lceil \log_2 m \rceil}{2\ell} = \Omega(\delta/\log n)$, where $n = m^\ell$.

[17]See Exercise 1.

The second reduction refers to functions over any partial order, and it relates the performance of any two-query POT in the case of a general range to the performance of the same POT on a binary range. Specifically, the probability that this POT rejects any function that is δ-far from the set of monotone functions (with general range) is lower-bounded in terms of the probability that this very POT rejects any Boolean function that is δ-far from the set of monotone (Boolean) functions. (This is reminiscent of the "0-1 principle for sorting network" that states that a comparison-based sorting network that works on binary inputs also works on general inputs, except that here the "extension of the range" does not come for free.)

Lemma 14 (range reduction): *Let P be an arbitrary partial order set over n elements, and R be an arbitrary totally ordered set. Let $\mathcal{D}$ be an arbitrary distribution over pairs $(x,y) \in P \times P$ such that $x < y$* (according to the partial order P). *Suppose that for some convex function $\varrho : (0,1] \to (0,1]$ and for every $g : P \to \{0,1\}$ it holds that*

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[g(x) > g(y)] \;\geq\; \varrho(\delta_2(g)),$$

where $\delta_2(g)$ denotes the distance of g from the set of Boolean monotone functions. Then, for every $f : P \to R$ it holds that

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x) > f(y)] \;\geq\; \frac{\varrho(\delta(f))}{\lceil \log_2 |R| \rceil},$$

where $\delta(f)$ denotes the distance of f from the set of monotone functions (with range R).

Hence, we lose a factor of $\log_2 |R|$ in the detection probability, where without loss of generality we may use R as the range of the tested function (and so $|R| \leq n$). Letting Π_n denote the set of all monotone functions from $[m]^\ell$ to R_n, where $n = m^\ell$, and combining all the above[18], we get –

Corollary 15 (a two-query POT for multi-value monotonicity over $[m]^\ell$): *There exists an efficient* (one-sided error) *two-query proximity oblivious tester for Π_n with detection probability $\Omega(\delta/\log^2 n)$, where δ denotes the distance of the given function from being monotone.*

Indeed, the above lower bound is a simplification of $\frac{\delta}{2\ell \cdot \lceil \log_2 m \rceil \cdot \lceil \log_2 |R_n| \rceil}$.

4.1 Dimension reduction (proof of Lemma 13)

This proof generalizes the proof of Theorem 2. Specifically, monotonicity in direction $i \in [\ell]$ is defined in the natural manner (extending Definition 2.1), whereas the switch operator is replaced by a sorting operator; that is, for every $i \in [\ell]$, the sorting operator S_i is defined such that for every function $f : [m]^\ell \to \{0,1\}$ the function $S_i(f) : [m]^\ell \to \{0,1\}$ is monotone in direction i and satisfies $\sum_{k\in[m]} S_i(f)(\alpha k \beta) = \sum_{k\in[m]} f(\alpha k \beta)$ for every $\alpha \in [m]^{i-1}$ and $\beta \in [m]^{\ell-i}$. (Indeed, this generalizes Definition 2.2, since for $v_1, v_2 \in \{0,1\}$ there is a 1-1 correspondence between the possible sets $\{v_1, v_2\}$ and the possible values $v_1 + v_2$.)

When counting violations in direction i we shall use a more refined extension of Definition 2.3. Specifically, for every $i \in [\ell]$, $\alpha \in [m]^{i-1}$ and $\beta \in [m]^{\ell-i}$, we have $\binom{m}{2}$ directed pairs rather than one: For $1 \leq k_1 < k_2 \leq m$, the directed pair $(\alpha k_1 \beta, \alpha k_2 \beta)$ is a violating (k_1,k_2)-pair of g in direction

[18]Starting with a POT for Boolean functions over $[m]$, we first apply Lemma 13 to obtain a POT for Boolean functions over $[m]^\ell$, and then apply Lemma 14 to obtain a POT for multi-valued functions over $[m]^\ell$.

i if $g(\alpha k_1\beta) > g(\alpha k_2\beta)$. We denote by $V_i^{k_1,k_2}(g)$ the set of violating (k_1,k_2)-pairs of g in direction i.

The generalization of Claim 2.4 asserts that *for every* $g : [m]^\ell \to \{0,1\}$ *and* $i,j \in [\ell]$ *and* $1 \le k_1 < k_2 \le m$, *it holds that* $|V_j^{k_1,k_2}(S_i(g))| \le |V_j^{k_1,k_2}(g)|$. This is proved by fixing $i = 1$, $j = 2$, $k_1 < k_2$ and $u \in [m]^{\ell-2}$, and considering the function $g_u^{k_1,k_2} : [m] \times [2] \to \{0,1\}$ such that $g_u^{k_1,k_2}(\sigma,\tau) = g(\sigma k_\tau u)$. The key observation is that the effect of S_1 on $V_2^{k_1,k_2}$ can be decomposed among the various $g_u^{k_1,k_2}$'s. Furthermore, considering an m-by-2 Boolean submatrix, note that the number of unsorted rows may only decrease when the columns are sorted. This is the case, because the minimal number of unsorted rows in a submatrix with t_τ ones in the τ's column is $\min(t_1-t_2,0)$, and this minimum is obtained when the columns are sorted.

Starting with an arbitrary Boolean function $f_0 : [m]^\ell \to \{0,1\}$, we consider the (analogous) sequence of f_i's defined by $f_i = S_i(f_{i-1})$. Generalizing Corollary 2.5, we infer that f_ℓ is monotone and that $|V_j^{k_1,k_2}(f_i)| \le |V_j^{k_1,k_2}(f_0)|$, for every $i,j \in [\ell]$ and $1 \le k_1 < k_2 \le m$. Letting $\delta_{i,\alpha,\beta}$ denote the (relative) distance of the sequence $f_{i-1}(\alpha 1\beta), ..., f_{i-1}(\alpha m\beta)$ from a monotone sequence, we get

$$\begin{aligned}
\delta(f_0,f_\ell) &\le \sum_{i\in[\ell]} \delta(f_{i-1},f_i) \\
&= \sum_{i\in[\ell]} \mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}\left[|\{k\in[m] : f_{i-1}(\alpha k\beta) \ne f_i(\alpha k\beta)\}|\right]/m \\
&\le \sum_{i\in[\ell]} \mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}\left[2\delta_{i,\alpha,\beta}\right]
\end{aligned}$$

where the last inequality follows by observing that the distance of a Boolean sequence $s = (e_1, ..., e_m)$ from its sorted version (i.e., $0^{m-\mathrm{wt}(s)}1^{\mathrm{wt}(s)}$) is at most twice the distance of s to being monotone.[19] On the other hand, the probability, denoted ρ, that Algorithm 12 rejects f_0, when using T in Step 2 (where T selects pairs of queries according to the distribution $\mathcal{D}$), is

$$\begin{aligned}
\rho &= \mathbb{E}_{i\in[\ell]}\mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}\left[\mathbf{Pr}_{(k_1,k_2)\sim\mathcal{D}}[f_0(\alpha k_1\beta) > f_0(\alpha k_2\beta)]\right] \\
&= m^{-(\ell-1)}\cdot\mathbb{E}_{i\in[\ell]}\left[\mathbb{E}_{(k_1,k_2)\sim\mathcal{D}}[|V_i^{k_1,k_2}(f_0)|]\right] \\
&\ge m^{-(\ell-1)}\cdot\mathbb{E}_{i\in[\ell]}\left[\mathbb{E}_{(k_1,k_2)\sim\mathcal{D}}[|V_i^{k_1,k_2}(f_{i-1})|]\right] \\
&\ge \mathbb{E}_{i\in[\ell]}\mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}\left[\varrho_m(\delta_{i,\alpha,\beta})\right]
\end{aligned}$$

where the first equality is due to the definition of T, the second equality is due to the definition of V_{k_1,k_2}, the first inequality is due to the (second item of the) generalization of Corollary 2.5, and the last inequality is due to the definitions of T and ϱ_m. Using the convexity of ϱ_m and combining the above, we get

$$\begin{aligned}
\rho &\ge \mathbb{E}_{i\in[\ell]}\mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}\left[\varrho_m(\delta_{i,\alpha,\beta})\right] \\
&\ge \varrho_m\left(\mathbb{E}_{i\in[\ell]}\mathbb{E}_{(\alpha,\beta)\in[m]^{i-1}\times[m]^{\ell-i}}[\delta_{i,\alpha,\beta}]\right) \\
&\ge \varrho_m(\delta(f_0,f_\ell)/2\ell).
\end{aligned}$$

Recalling that f_ℓ in monotone (by the first item of the generalization of Corollary 2.5), the lemma follows (since f_0 is at distance at most $\delta(f_0,f_\ell)$ from being monotone).

[19] This assertion relies on the hypothesis that the sequence is binary and does not hold otherwise; see Exercise 4.

4.2 Range reduction (overview of the proof of Lemma 14)

Without loss of generality, we assume that $R = [r]$ and that r is a power of two. The key idea is that the values assigned to the two endpoints of a violating edge are either both at the same half of the interval $[1, r]$ or are in different halves (i.e., one value is in $[1, 0.5r]$ and the other is in $[0.5r + 1, r]$). The first type of edges can be handled by a reduction to two disjoint intervals each of length $r/2$, whereas the edges of the second type can be handled by a reduction to an interval of length two (with the two values representing the two halves). Needless to say, these separate handlings should be performed in a way that allows for their later integration.

Towards this end, for every $1 \le a < b \le r$, we define a filter operator $F_{a,b}$ such that for every function $f : P \to [r]$ the function $F_{a,b}(f) : P \to [a, b]$ satisfies

$$F_{a,b}(f)(x) = \begin{cases} a & \text{if } f(x) \le a \\ b & \text{if } f(x) \ge b \\ f(x) & \text{otherwise (i.e., if } f(x) \in [a,b]) \end{cases} \tag{1}$$

Hence, the two types of violations with respect to f appear either in $F_{1,0.5r}(f)$ (or in $F_{0.5r+1,r}(f)$) or in $F_{0.5r,0.5r+1}(f)$. In order to facilitate the integration, we introduce a corresponding discarding operator $D_{a,b}$, which allows ignoring the modifications that are required for making $F_{a,b}(f)$ monotone and focusing on what is required beyond this in order to make f itself monotone. Specifically, for every function $f : P \to [r]$, we fix a monotone function $g : P \to [a, b]$ that is closest to $F_{a,b}(f)$, and define

$$D_{a,b}(f)(x) = \begin{cases} g(x) & \text{if } F_{a,b}(f)(x) \ne g(x) \\ f(x) & \text{otherwise (i.e., if } F_{a,b}(f)(x) = g(x)) \end{cases} \tag{2}$$

In particular, if $f(x) \notin [a, b]$, then $D_{a,b}(f)(x) = f(x)$ unless $F_{a,b}(f)(x) \ne g(x)$; that is, values outside of $[a, b]$ are modified by $D_{a,b}$ only if they are modified when making the filtered function monotone.[20] Now, given an arbitrary function $f : P \to [r]$, we consider the following sequence of auxiliary functions:

1. $f' = F_{0.5r,0.5r+1}(f)$, which ranges over $\{0.5r, 0.5r + 1\}$
2. $f'' = F_{1,0.5r}(D_{0.5r,0.5r+1}(f'))$, which ranges over $[1, 0.5r]$.
3. $f''' = F_{0.5r+1,r}(D_{1,0.5r}(f''))$, which ranges over $[0.5r + 1, r]$.

Denoting by $\delta_{[a,b]}(g)$ the relative distance of $g : P \to [a, b]$ from the set of monotone functions over P with range $[a, b]$, one can prove the following claims.

Claim 1: $\delta_{[1,r]}(f) \le \delta_{[0.5r,0.5r+1]}(f') + \delta_{[1,0.5r]}(f'') + \delta_{[0.5r+1,r]}(f''')$.

Claim 2: $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x)>f(y)] \;\ge\; \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f'(x)>f'(y)]$.

Claim 3: $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x)>f(y)] \;\ge\; \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f''(x)>f''(y)] \;+\; \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f'''(x)>f'''(y)]$.

Claim 2 is quite easy to establish (see Exercise 5). Claims 1 and 3 seem quite intuitive, but they do require proofs, which are a bit tedious (and are omitted here). Once all claims are proved, the lemma can be proved by induction. The induction step proceeds as follows, when $s = \log_2 r$:

[20] Note that, if $f(x) \in [a, b]$, then $F_{a,b}(f)(x) = f(x)$, which implies $D_{a,b}(f)(x) = g(x)$.

$$\begin{aligned}\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x)>f(y)] &\geq \frac{1}{s}\cdot\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f'(x)>f'(y)] \\ &\quad+\frac{s-1}{s}\cdot\left(\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f''(x)>f''(y)]\ +\ \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f'''(x)>f'''(y)]\right) \\ &\geq \frac{1}{s}\cdot\varrho(\delta_{[0.5r,0.5r+1]}(f'))+\frac{s-1}{s}\cdot\left(\frac{\varrho(\delta_{[1,0.5r]}(f''))}{s-1}+\frac{\varrho(\delta_{[0.5r+1,r]}(f'''))}{s-1}\right) \\ &\geq \frac{\varrho(\delta_{[0.5r,0.5r+1]}(f')+\delta_{[1,0.5r]}(f''))+\delta_{[0.5r+1,r]}(f'''))}{s} \\ &\geq \frac{\varrho(\delta_{[1,r]}(f))}{s}\end{aligned}$$

where the first inequality uses Claims 2 and 3, the second inequality uses the induction hypothesis, the third inequality uses the convexity of ϱ, and the last inequality uses Claim 1.

5 Suggested Reading and Exercises

Monotonicity testing was first considered by Goldreich *et al.* [11] and Ergün *et al.* [7]: While Goldreich *et al.* [11] considered Boolean functions over the partial order associated with the hypercube, Ergun *et al.* [7] considered multi-valued functions over the total order associated with the line (see Sections 2.1 and 2.2, respectively).[21] The interpolation of both cases, presented in Section 4, refers to multi-valued functions over the partial order associated with the hypergrid $[m]^\ell$. This case is reduced to the case of Boolean functions over $[m]^\ell$, which is then reduced to the case of Boolean functions over $[m]$. The range reduction is due to Dodis *et al.* [6], whereas the dimension reduction appears in [11, 6]. Recall that the resulting two-query POT has detection probability at least $\delta/O(\log n\cdot\log|R_n|)$, where $n=m^\ell$ and R_n denotes the range of these functions (see Corollary 15). An improved bound of $\delta/O(\log n)$ was recently obtained in [3], and this is optimal for $|R_n|=\Omega(\sqrt{\ell})$. We mention that a study of monotonicity testing in general partially ordered sets was initiated by Fischer *et al.* [9].

The exact complexity of testing monotonicity of Boolean functions over the Boolean hypercube has attracted much attention for over a decade. In particular, the question was whether the detection probability must decrease linearly with the dimension. In fact, it was conjectured that the detection probability may decrease linearly with the square root of the dimension. This conjecture was established, up to polylogarithmic factors (see Theorem 4), by Khot, Minzer, and Safra [13]. Their result improved over a prior result of Chakrabarty and Seshadhri [2], which established a sub-linear dependence on the dimension. Interestingly, this upper bound is almost tight [4] (improving over [5]). For other related results regarding the complexity of testing monotonicity (and related problems), the reader is referred to [8, 3].

Related problems

Two properties of functions that are related to monotonicy via their reference to a (partially) ordered domain as well as the specific domains considered are the property of satisfying the *Lipschitz*

[21]The focus of Goldreich *et al.* [11] was on monotonicity testing, whereas the investigation of Ergun *et al.* [7] was far broader than that.

condition and *submodularity*. In both cases, we consider the domain $[m]^\ell$ as well as special cases in which either $m = 2$ or $\ell = 1$.

Lipschitz functions. A function $f : [m]^\ell \to \mathbb{R}$ is called c-Lipschitz if for every $x, y \in [m]^\ell$ it holds that $|f(x) - f(y)| \leq c \cdot \|x - y\|_1$, where $\|x - y\|_1 = \sum_{i\in[\ell]} |x_i - y_i|$. The study of testing (and reconstructing) Lipschitz functions was initiated by Jha and Raskhodnikova [12], who were motivated by applications to data privacy. Although it seem that testing Lipschitz functions can not be reduced to testing monotonicity, Chakrabarty and Seshadhri presented a uniform framework that covers both problems [3], and obatined a two-query POT of detection probability $\delta/O(\ell \log m)$ for both problems.[22]

Submodular functions. A function $f : [m]^\ell \to \mathbb{R}$ is called submodular if for every $x = (x_1, ..., x_\ell)$ and $y = (y_1, ..., y_\ell)$ in $[m]^\ell$ it holds that

$$f(\max(x, y)) - f(\min(x, y)) \leq (f(x) - f(\min(x, y))) + (f(y) - f(\min(x, y))) \qquad (3)$$

where $\max((x_1, ..., x_\ell), (y_1, ..., y_\ell)) = (\max(x_1, y_1), ..., \max(x_\ell, y_\ell))$ and ditto for $\min(x, y)$. Indeed, Eq. (4) is equivalent to $f(\max(x, y)) + f(\min(x, y)) \leq f(x) + f(y)$, and it is meaningless for $\ell = 1$. The study of testing submodularity was initiated by Parnas, Ron, and Rubinfeld [14], who focused on the case of $\ell = 2$ (which corresponds to "Monge matrices"), and presented a $O(1)$-query POT that has detection probability $\Omega(\delta/\log^2 m)$.[23] Seshadhri and Vondrak [16] considered the case of $m = 2$ (which corresponds to "modular set functions"), and showed a natural four-query POT of detection probability $\delta^{\widetilde{O}(\sqrt{\ell})}$.

Convex functions. Another property considered in [14] is convexity. A function $f : [m]^\ell \to \mathbb{R}$ is called convex if for every $x, y \in [m]^\ell$ and every $\alpha \in [0, 1]$ such that $z = \alpha x + (1 - \alpha)y \in [m]^\ell$ it holds that $f(z) \leq \alpha \cdot f(x) + (1 - \alpha) \cdot f(y)$. While submodularity refers to the "rectangle spanned by x and y" (along with $\max(x, y)$ and $\min(x, y)$), convexity refers to the line that connects x and y. Focusing on the case of $\ell = 1$, a $O(1)$-query POT was shown in [14] to have detection probability $\Omega(\delta/\log m)$.[24]

Invariances. We note that all properties studied in this lecture are invariant under a permutation of the variables; that is, for each of these properties Π, the function $f : \{0, 1\}^\ell \to \{0, 1\}$ is in Π if and only if for every permutation $\pi : [\ell] \to [\ell]$ the function $f_\pi(x_1, ..., x_\ell) = f(x_{\pi(1)}, ..., x_{\pi(\ell)})$ is in Π.

Exercises

The following exercises detail some claims that were made in the main text. In addition, Exercise 5 calls for proving Claims 1-3 of Section 4.2.

[22] Throughout this section, δ denotes the distance of the tested function from the property.

[23] The property tester presented in [14, Alg. 3] employs a $O(\log^2 m)$-query POT of detection probability $\Omega(\delta)$, but this POT conducts $O(\log^2 m)$ unrelated checks, which are determined non-adaptively, such that each check uses only $O(1)$ queries (see [14, Def. 9] and [14, Clm. 4]). (See the analogous passage from Theorem 9 to Theorem 10.)

[24] The property tester presented in [14, Alg. 1] employs a $O(\log m)$-query POT of detection probability $\Omega(\delta)$, but this POT (see [14, Proc. 1]) proceeds in $\log_2 m$ iterations that are actually non-adaptive and check unrelated conditions. (See an analogous passage in Footnote 23.)

Exercise 1 (characterization of monotonicity over the hypergrid): *Prove that the function* $f : [m]^\ell \to R_n$ *is monotone if and only if it is monotone in each direction* (i.e., if and only if for every $i \in [\ell]$ and for every $\alpha \in [m]^{i-1}$ and $\beta \in [m]^{\ell-i}$, the function $f'(z) = f(\alpha z \beta)$ is monotone in z).

Guideline: For the less obvious direction, given $x = x_1 \cdots x_\ell < y = y_1 \cdots y_\ell$ in $[m]^\ell$, consider the sequence points $x_1 \cdots x_i y_{i+1} \cdots y_\ell \in [m]^\ell$, for $i = 0, 1, ..., \ell$.

Exercise 2 (a typical case in which Algorithm 1 is asymptotically optimal): *Recall that* $\delta_{\mathtt{M}}(f)$ *denotes the relative distance of* $f : \{0,1\}^\ell \to \{0,1\}$ *from being monotone, whereas* $\rho(f)$ *denotes the probability that Algorithm 1 rejects* f. *Note that* $\rho(f) \leq 2\delta_{\mathtt{M}}(f)$ *for every* f. *Show that for* $f(x) = \mathrm{wt}(x) \bmod 2$ *it holds that* $\delta_{\mathtt{M}}(f) \approx 0.5$ *and* $\rho(f) \approx 0.5$

Guideline: Consider the set of edges between strings of odd Hamming weight and strings that are one unit heavier (i.e., the edge (x, y) is in this set if and only if $\mathrm{wt}(x)$ is odd and $\mathrm{wt}(y) = \mathrm{wt}(x)+1$). Note that Algorithm 1 rejects f if and only if it selected such an edge, and that this set of edges constitutes a matching.

Exercise 3 (vertex covers in the graph of violating pairs): *Let* P *be an arbitrary partial order set over* n *elements, and suppose that* $f : P \to R$ *is at distance* δ *from the set of monotone functions over* P (with range R). *Consider the graph* G^f *such that* $\{x, y\}$ *is an edge if* $x < y$ *but* $f(x) > f(y)$. *Then,* G^f *has no vertex cover of size smaller than* δn.

Guideline: Since the restriction of f to any independent set of G^f is a monotone function, f can be made monotone by modifying its values at the vertex cover.

Exercise 4 (distance to monotone vs distance to the sorted version): *Prove that the distance of a sequence* $\overline{s} = (e_1, ..., e_m) \in R^m$ *to its sorted version is at most* $|R|$ *times the distance of* $\overline{s}$ *to a monotone* m*-sequence over* R. *Show that this upper bound is tight.*

Guideline: As a warm-up consider the case of $R = \{0,1\}$. Suppose that $\overline{s}$ has z zeros and let t denote the number of ones in the z-bit long prefix of $\overline{s}$. Then, $\overline{s}$ is at distance $2t$ from its sorted version, and at distance at least t from a monotone sequence, where the last assertion is proved by considering a matching between the t ones in the z-bit long prefix of $\overline{s}$ and the t zeros at its $(m-z)$-bit long suffix. For a general R, suppose that $\overline{s}$ has m_i occurrences of the value $i \in R$ and let $m'_i = \sum_{j \leq i} m_j$. Let $D_i^+ \subseteq [m'_{i-1}+1, m'_i]$ (resp., $D_i^- \subseteq [m'_{i-1}+1, m'_i]$) be the set of positions that hold the value i in the sorted version of $\overline{s}$ but hold a value larger (resp., smaller) than i in $\overline{s}$ itself. (In the warm-up, $D_0^+ \subseteq [1, z]$ had size t, and ditto $D_1^- \subseteq [z+1, m]$.) Note that $\overline{s}$ differs from its sorted version on $\sum_i |D_i^+ \cup D_i^-|$ positions. Show that $\cup_{i<j} D_i^+ \times D_j^-$ contains a matching of size $\sum_i |D_i^+ \cup D_i^-|/|R|$, by considering the cycle structure of a permutation that sorts $\overline{s}$ by moving a minimal number of elements. (Note that each such cycle has at least one edge in $\cup_{i<j} D_i^+ \times D_j^-$, whereas w.l.o.g it has at most one position in each interval $[m'_{i-1}+1, m'_i]$.)[25] To see that the upper bound is tight consider the sequence $(m, 1, 2, ..., m-1)$.

Exercise 5 (Claims 1-3 of Section 4.2): *Prove Claims 1-3 of Section 4.2. Claim 2 is proved by showing that the filter operator never increases the number of violating pairs. As a warm-up towards proving Claims 1 and 3, prove the following weaker analogues:*

[25] A simpler proof of the upper bound is indeed welcomed.

Claim 1w: $\delta_{[1,r]}(f) \leq \delta_{[0.5r,0.5r+1]}(f) + \delta_{[1,0.5r]}(f) + \delta_{[0.5r+1,r]}(f)$.

Claim 3w: $\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[f(x) > f(y)]$ *is lower-bounded by*

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[F_{1,0.5r}(f)(x) > F_{1,0.5r}(f)(y)] + \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[F_{0.5r+1,r}(f)(x) > F_{0.5r+1,r}(f)(y)].$$

Claims 1-3 appear as items of [6, Lem. 14], using somewhat different notations, and their proofs appear in [6, Sec. 4.1-4.2].

Guideline: Moving from the warm-up claims to the actual claims requires establishing some features of the operator $D_{a,b}$. Denoting the set of violating pairs for g by $V(g)$, the most useful features include

1. $V(D_{a,b}(h)) \subseteq V(h)$;
2. if $(x,y) \in V(h)$ and $|\{h(x),h(y)\} \cap [a,b]| = 2$, then $(x,y) \notin V(D_{a.b}(h))$;
3. if $(x,y) \in V(D_{a,b}(h))$, then $[D_{a,b}(h)(y), D_{a,b}(h)(x)] \subseteq [h(y), h(x)]$.

These facts appear as items of [6, Lem. 13].

References

[1] J. Briet, S. Chakraborty, D. Garcia-Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, Vol. 32 (1), pages 35–53, 2012.

[2] D. Chakrabarty and C. Seshadhri. A o(n) monotonicity tester for boolean functions over the hypercube. In *45th ACM Symposium on the Theory of Computing*, pages 411–418, 2013.

[3] D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *45th ACM Symposium on the Theory of Computing*, pages 419–428, 2013.

[4] X. Chen, A. De, R.A. Servedio, and L. Tan. Boolean Function Monotonicity Testing Requires (Almost) $n^1/2$ Non-adaptive Queries. In *47th ACM Symposium on the Theory of Computing*, pages 519–528, 2015.

[5] X. Chen, R.A. Servedio, and L. Tan. New Algorithms and Lower Bounds for Monotonicity Testing. In *55th IEEE Symposium on Foundations of Computer Science*, pages 286–295, 2014.

[6] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved Testing Algorithms for Monotonicity. *ECCC*, TR99-017, 1999. Extended abstract in *3rd RANDOM*, 1999.

[7] F. Ergun, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot checkers. *Journal of Computer and System Science*, Vol. 60, pages 717–751, 2000. Extended abstract in *30th STOC*, 1998.

[8] E. Fischer. On the strength of comparisons in property testing. *Information and Computation*, Vol. 189(1), pages 107–116, 2004.

[9] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity Testing Over General Poset Domains. In *34th ACM Symposium on the Theory of Computing*, pages 474–483, 2002.

[10] O. Goldreich (ed.). *Property Testing: Current Research and Surveys.* Springer, LNCS, Vol. 6390, 2010.

[11] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing Monotonicity. *Combinatorica*, Vol. 20 (3), pages 301–337, 2000. Extended abstract in *39th FOCS*, 1998.

[12] M. Jha and S. Raskhodnikova. Testing and Reconstruction of Lipschitz Functions with Applications to Data Privacy. *SIAM Journal on Computing*, Vol. 42(2), pages 700–731, 2013. Extended abstract in *52nd FOCS*, 2011.

[13] S. Khot, D. Minzer, and S. Safra. On Monotonicity Testing and Boolean Isoperimetric type Theorems. *ECCC*, TR15-011, 2015.

[14] M. Parnas, D. Ron, and R. Rubinfeld. On Testing Convexity and Submodularity. *SIAM Journal on Computing*, Vol. 32 (5), pages 1158–1184, 2003.

[15] S. Raskhodnikova. Transitive Closure Spanners: A Survey. In [10].

[16] C. Seshadhri, J. Vondrak. Is Submodularity Testable? *Algorithmica*, Vol. 69(1), pages 1–25, 2014. Extended abstract in *2nd Innovations in (Theoretical) Computer Science*, 2011.