Optimal Unateness Testers for Real-Valued Functions: Adaptivity Helps*†

Roksana Baleshzar[1], Deeparnab Chakrabarty[2], Ramesh Krishnan S. Pallavoor[3], Sofya Raskhodnikova[4], and C. Seshadhri[5]

1 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
rxb5410@cse.psu.edu
2 **Department of Computer Science, Dartmouth College, Hanover, NH, USA**
deeparnab@dartmouth.edu
3 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
ramesh@psu.edu
4 **Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA**
sofya@cse.psu.edu
5 **Department of Computer Science, University of California – Santa Cruz, Santa Cruz, CA, USA**
sesh@ucsc.edu

Abstract

We study the problem of testing unateness of functions $f : \{0,1\}^d \to \mathbb{R}$. We give an $O(\frac{d}{\varepsilon} \cdot \log \frac{d}{\varepsilon})$-query nonadaptive tester and an $O(\frac{d}{\varepsilon})$-query adaptive tester and show that both testers are optimal for a fixed distance parameter ε. Previously known unateness testers worked only for Boolean functions, and their query complexity had worse dependence on the dimension both for the adaptive and the nonadaptive case. Moreover, no lower bounds for testing unateness were known. We generalize our results to obtain optimal unateness testers for functions $f : [n]^d \to \mathbb{R}$.

Our results establish that adaptivity helps with testing unateness of real-valued functions on domains of the form $\{0,1\}^d$ and, more generally, $[n]^d$. This stands in contrast to the situation for monotonicity testing where there is no adaptivity gap for functions $f : [n]^d \to \mathbb{R}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Property testing, unate and monotone functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.5

1 Introduction

We study the problem of testing whether a given real-valued function f on domain $[n]^d$, where $n, d \in \mathbb{N}$, is unate. A function $f : [n]^d \to \mathbb{R}$ is *unate* if for every coordinate $i \in [d]$, the function is either nonincreasing in the coordinate i or nondecreasing in the coordinate i. Unate functions naturally generalize monotone functions, which are nondecreasing in all

* A preliminary full version of this paper appears in ECCC [2], https://eccc.weizmann.ac.il/report/2017/049/.

† The authors R.B., R.P. and S.R. were partially supported by NSF award CCF-1422975. This work was done while D.C. was at Microsoft Research, India.

coordinates, and **b**-monotone functions, which have a particular direction in each coordinate (either nonincreasing or nondecreasing), specified by a bit-vector $\mathbf{b} \in \{0,1\}^d$. More precisely, a function is **b**-monotone if it is nondecreasing in coordinates i with $\mathbf{b}_i = 0$ and nonincreasing in the other coordinates. Observe that a function f is unate iff there exists some $\mathbf{b} \in \{0,1\}^d$ for which f is **b**-monotone.

A *tester* [33, 25] for a property $\mathcal{P}$ of a function f is an algorithm that gets a distance parameter $\varepsilon \in (0,1)$ and query access to f. It has to accept with probability at least $2/3$ if f has property $\mathcal{P}$ and reject with probability at least $2/3$ if f is ε-far (in Hamming distance) from $\mathcal{P}$. We say that f is ε-far from $\mathcal{P}$ if at least an ε fraction of values of f must be modified to make f satisfy $\mathcal{P}$. A tester has *one-sided error* if it always accepts a function satisfying $\mathcal{P}$, and has *two-sided error* otherwise. A *nonadaptive* tester makes all its queries at once, while an *adaptive* tester can make queries after seeing answers to the previous ones.

Testing of various properties of functions, including monotonicity (see, e.g., [24, 19, 20, 31, 22, 21, 26, 1, 27, 3, 8, 7, 10, 13, 9, 6, 14, 15, 12, 17, 16, 29, 4, 5, 18] and recent surveys [32, 11]), the Lipschitz property [28, 13, 9, 16], bounded-derivative properties [12], and unateness [24, 30], has been studied extensively over the past two decades. Even though unateness testing was initially discussed in the seminal paper by Goldreich et al. [24] that gave first testers for properties of functions, relatively little is known about testing this property. All previous work on unateness testing focused on the special case of Boolean functions on domain $\{0,1\}^d$. The domain $\{0,1\}^d$ is called the *hypercube* and the more general domain $[n]^d$ is called the *hypergrid*. Goldreich et al. [24] provided a $O(\frac{d^{3/2}}{\varepsilon})$-query nonadaptive tester for unateness of Boolean functions on the hypercube. Recently, Khot and Shinkar [30] improved the query complexity to $O(\frac{d \log d}{\varepsilon})$, albeit with an adaptive tester.

In this paper, we improve upon both these works, and our results hold for a more general class of functions. Specifically, we show that unateness of real-valued functions on hypercubes can be tested nonadaptively with $O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon})$ queries and adaptively with $O(\frac{d}{\varepsilon})$ queries. More generally, we describe a $O(\frac{d}{\varepsilon} \cdot (\log \frac{d}{\varepsilon} + \log n))$-query nonadaptive tester and a $O(\frac{d \log n}{\varepsilon})$-query adaptive tester of unateness of real-valued functions over hypergrids.

In contrast to the state of knowledge for unateness testing, the complexity of testing monotonicity of real-valued functions over the hypercube and the hypergrid has been resolved. For constant distance parameter ε, it is known to be $\Theta(d \log n)$. Moreover, this bound holds for all *bounded-derivative* properties [12], a large class that includes **b**-monotonicity and some properties quite different from monotonicity, such as the Lipschitz property. Amazingly, the upper bound for all these properties is achieved by the same simple and, in particular, nonadaptive, tester. Even though proving lower bounds for adaptive testers has been challenging in general, a line of work, starting from Fischer [21] and including [8, 14, 12], has established that adaptivity does not help for this large class of properties. Since unateness is so closely related, it is natural to ask whether the same is true for testing unateness.

We answer this in the negative: we prove that any non-adaptive tester of real valued functions over the hypercube (for some constant distance parameter) must make $\Omega(d \log d)$ queries. More generally, it needs $\Omega(d(\log d + \log n))$ queries for the hypergrid domain. These lower bounds complement our algorithms, completing the picture for unateness testing of real-valued functions. From a property testing standpoint, our results establish that unateness is different from monotonicity and, more generally, any derivative-bounded property.

1.1 Formal Statements and Technical Overview

Our upper bounds are summarized in the following theorem, stated for functions over the hypergrid domains. (Recall that the hypercube is a special case of the hypergrid with $n = 2$.)

▶ **Theorem 1.1.** *Consider functions $f : [n]^d \to \mathbb{R}$ and a distance parameter $\varepsilon \in (0, 1/2)$.*
1. *There is a nonadaptive unateness tester that makes $O(\frac{d}{\varepsilon}(\log \frac{d}{\varepsilon} + \log n))$ queries*[1].
2. *There is an adaptive unateness tester that makes $O(\frac{d \log n}{\varepsilon})$ queries.*

Both testers have one-sided error.

Our main technical contribution is the proof that the extra $\Omega(\log d)$ is needed for nonadaptive testers. This result demonstrates a gap between adaptive and nonadaptive unateness testing.

▶ **Theorem 1.2.** *Any nonadaptive unateness tester (even with two-sided error) for real-valued functions $f : \{0,1\}^d \to \mathbb{R}$ with distance parameter $\varepsilon = 1/8$ must make $\Omega(d \log d)$ queries.*

The lower bound for adaptive testers is an easy adaptation of the monotonicity lower bound in [14]. We state this theorem for completeness and prove it in the full version [2].

▶ **Theorem 1.3.** *Any unateness tester for functions $f : [n]^d \to \mathbb{R}$ with distance parameter $\varepsilon \in (0, 1/4)$ must make $\Omega\left(\frac{d \log n}{\varepsilon} - \frac{\log 1/\varepsilon}{\varepsilon}\right)$ queries.*

Theorems 1.2 and 1.3 directly imply that our nonadaptive tester is optimal for constant ε, even for the hypergrid domain. All missing proofs and details from the technical sections appear in the full version of this paper [2].

1.1.1 Overview of Techniques

We first consider the hypercube domain. For each $i \in [d]$, an *i-edge* of the hypercube is a pair (x, y) of points in $\{0,1\}^d$, where $x_i = 0, y_i = 1$, and $x_j = y_j$ for all $j \in ([d] \setminus \{i\})$. Given an input function $f : \{0,1\}^d \to \mathbb{R}$, we say an i-edge (x, y) is *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

Our nonadaptive unateness tester on the hypercube uses the work investment strategy from [6] (also refer to Section 8.2.4 of Goldreich's book [23]) to "guess" a good dimension where to look for violations of unateness (specifically, both increasing and decreasing edges). For all $i \in [d]$, let α_i be the fraction of the i-edges that are decreasing, β_i be the fraction of the i-edges that are increasing, and $\mu_i = \min(\alpha_i, \beta_i)$. The dimension reduction theorem from [12] implies that if the input function is ε-far from unate, then the average of μ_i over all dimensions is at least $\frac{\varepsilon}{4d}$. If the tester knew which dimension had $\mu_i = \Omega(\varepsilon/d)$, it could detect a violation with high probability by querying the endpoints of $O(1/\mu_i) = O(d/\varepsilon)$ uniformly random edges. However, the tester does not know which μ_i is large and, intuitively, nonadaptively checks the following $\log d$ different scenarios, one for each $k \in [\log d]$: exactly 2^k different μ_i's are $\varepsilon/2^k$, and all others are 0. This leads to the query complexity of $O(\frac{d \log d}{\varepsilon})$.

With adaptivity, this search through $\log d$ different scenarios is not required. A pair of queries in each dimension detects influential coordinates (i.e., dimensions with many non-constant edges), and the algorithm focuses on finding violations among those coordinates. This leads to the query complexity of $O(d/\varepsilon)$, removing the $\log d$ factor.

It is relatively easy to extend (both adaptive and nonadaptive) testers from hypercubes to hypergrids by incurring an extra factor of $\log n$ in the query complexity. The role of i-edges is now played by *i-lines*. An i-line is a set of n domain points that differ only on coordinate i. The domain $[n]$ is called a line. Monotonicity on the line (a.k.a. sortedness) can be tested with $O(\frac{\log n}{\varepsilon})$ queries, using, for example, the classical *tree tester* from [20].

[1] For many properties, when the domain is extended from the hypercube to the hypergrid, testers incur an extra multiplicative factor of $\log n$ in the query complexity. This is the case for our adaptive tester. However, note that the complexity of nonadaptive unateness testing (for constant ε) is $\Theta(d(\log d + \log n))$ rather than $\Theta(d \log d \log n)$.

Instead of sampling a random i-edge, we sample a random i-line ℓ and run the tree tester on the restriction $f_{|\ell}$ of function f to the line ℓ. This is optimal for adaptive testers, but, interestingly, not for nonadaptive testers. We show that for each function f on the line that is ε-far from unateness, one of the two scenarios happen: (1) the tree tester is likely to find a violation of unateness; (2) function f is increasing (and also decreasing) on a constant fraction of pairs in $[n]$. This new angle on the classical tester allows us to replace the factor $(\log d)(\log n)$ with $\log d + \log n$ in the query complexity. Thus, the nonadaptive complexity becomes $O(d(\log d + \log n))$, which we show is optimal.

The nonadaptive lower bound. Our most significant finding is the $\log d$ gap in the query complexity between adaptive and nonadaptive testing of unateness. By previous work [21, 14], it suffices to prove lower bounds for *comparison-based* testers, i.e., testers that can only perform comparisons of the function values at queried points, but cannot use the values themselves. Our main technical contribution is the $\Omega(d \log d)$ lower bound for nonadaptive comparison-based testers of unateness on hypercube domains.

Intuitively, we wish to construct $K = \Theta(\log d)$ families of functions where, for each $k \in [K]$, functions in the k^{th} family have 2^k dimensions i with $\mu_i = \Theta(1/2^k)$, while $\mu_i = 0$ for all other dimensions. What makes the construction challenging is the existence of a *single, universal* nonadaptive $O(d)$-tester for all $\mathbf{b}$-monotonicity properties, proven in [12]. In other words, there is a single distribution on $O(d)$ queries that defines a nonadaptive property tester for $\mathbf{b}$-monotonicity, regardless of $\mathbf{b}$. Since unateness is the union of all $\mathbf{b}$-monotonicity properties, our construction must be able to fool such algorithms. Furthermore, nonadaptivity must be critical, since we obtained a $O(d)$-query adaptive tester for unateness.

Another obstacle is that once a tester finds a non-constant edge in each dimension, the problem reduces to testing $\mathbf{b}$-monotonicity for a vector $\mathbf{b}$ determined by the directions (increasing or decreasing) of the non-constant edges. That is, intuitively, most edges in our construction must be constant. This is one of the main technical challenges. The previous lower bound constructions for monotonicity testing [8, 14] crucially used the fact that all edges in the hard functions were non-constant.

We briefly describe how we overcome the problems mentioned above. By Yao's minimax principle, it suffices to construct **Yes** and **No** distributions that a deterministic nonadaptive tester cannot distinguish. First, for some parameter m, we partition the hypercube into m subcubes based of the first $\log_2 m$ most significant coordinates. Both distributions, **Yes** and **No**, sample a uniform k from $[K]$, where $K = \Theta(\log d)$, and a set $R \subseteq [d]$ of cardinality 2^k. Furthermore, each subcube $j \in [m]$ selects an "action dimension" $r_j \in R$ uniformly at random. For both distributions, in any particular subcube j, the function value is completely determined by the coordinates *not in* R, and the random coordinate $r_j \in R$. Note that all the i-edges for $i \in (R \setminus \{r_j\})$ are constant. Within the subcube, the function is a linear function with exponentially increasing coefficients. In the **Yes** distribution, any two cubes j, j' with the same action dimension orient the edges in that dimension the same way (both increasing or both decreasing), while in the **No** distribution each cube decides on the orientation independently. The former correlation maintains unateness while the latter independence creates distance to unateness. We prove that to distinguish the distributions, any comparison-based nonadaptive tester must find two distinct subcubes with the same action dimension r_j and, furthermore, make a specific query (in both) that reveals the coefficient of r_j. We show that, with $o(d \log d)$ queries, the probability of this event is negligible.

Algorithm 1: The Nonadaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

for $r = 1$ *to* $\lceil 3 \log(4d/\varepsilon) \rceil$ **do**
  repeat $s_r = \lceil \frac{16d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
    Sample a dimension $i \in [d]$ uniformly at random.
    Sample $3 \cdot 2^r$ i-edges uniformly and independently at random and **reject** if there exists an increasing edge and a decreasing edge among the sampled edges.
accept

2 Upper Bounds

In this section, we prove parts 1–2 of Theorem 1.1, starting from the hypercube domain.

Recall the definition of i-edges and i-lines from Section 1.1.1 and what it means for an edge to be increasing, decreasing, and constant.

The starting point for our algorithms is the dimension reduction theorem from [12]. It bounds the distance of $f : [n]^d \to \mathbb{R}$ to monotonicity in terms of average distances of restrictions of f to one-dimensional functions.

▶ **Theorem 2.1** (Dimension Reduction, Theorem 1.8 in [12]). *Fix a bit vector $\mathbf{b} \in \{0,1\}^d$ and a function $f : [n]^d \to \mathbb{R}$ which is ε-far from $\mathbf{b}$-monotonicity. For all $i \in [d]$, let μ_i be the average distance of $f_{|\ell}$ to $\mathbf{b}_i$-monotonicity over all i-lines ℓ. Then $\sum_{i=1}^{d} \mu_i \geq \varepsilon/4$.*

For the special case of the hypercube domains, i-lines become i-edges, and the average distance μ_i to $\mathbf{b}_i$-monotonicity is the fraction of i-edges on which the function is not $\mathbf{b}_i$-monotone.

2.1 The Nonadaptive Tester over the Hypercube

We now describe Algorithm 1, the nonadaptive tester for unateness over the hypercubes.

It is evident that Algorithm 1 is a nonadaptive, one-sided error tester. Furthermore, its query complexity is $O\left(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon}\right)$. It suffices to prove the following.

▶ **Lemma 2.2.** *If f is ε-far from unate, Algorithm 1 rejects with probability at least $2/3$.*

Proof. Recall that α_i is the fraction of i-edges that are decreasing, β_i is the fraction of i-edges that are increasing and $\mu_i = \min(\alpha_i, \beta_i)$.

Define the d-dimensional bit vector $\mathbf{b}$ as follows: for each $i \in [d]$, let $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$ and $\mathbf{b}_i = 1$ otherwise. Observe that the average distance of f to $\mathbf{b}_i$-monotonicity over a random i-edge is precisely μ_i. Since f is ε-far from being unate, f is also ε-far from being $\mathbf{b}$-monotone. By Theorem 2.1, $\sum_{i \in [d]} \mu_i \geq \frac{\varepsilon}{4}$. Hence, $\mathbf{E}_{i \in [d]}[\mu_i] \geq \frac{\varepsilon}{4d}$. We now apply the work investment strategy due to Berman et al. [6] to get an upper bound on the probability that Algorithm 1 fails to reject.

▶ **Theorem 2.3** ([6]). *For a random variable $X \in [0,1]$ with $\mathbf{E}[X] \geq \mu$ for $\mu < \frac{1}{2}$, let $p_r = \Pr[X \geq 2^{-r}]$ and $\delta \in (0,1)$ be the desired error probability. Let $s_r = \frac{4 \ln 1/\delta}{\mu \cdot 2^r}$. Then,*

$$\prod_{r=1}^{\lceil 3 \log(1/\mu) \rceil} (1 - p_r)^{s_r} \leq \delta.$$

Algorithm 2: The Adaptive Unateness Tester over Hypercubes

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : \{0,1\}^d \to \mathbb{R}$.

1 **repeat** $10/\varepsilon$ **times**
2 **for** $i = 1$ *to* d **do**
3 Sample an i-edge e_i uniformly at random.
4 **if** e_i *is non-constant (i.e., increasing or decreasing)* **then**
5 Sample i-edges uniformly at random till we obtain a non-constant edge e_i'.
6 **reject** if one of the edges e_i, e_i' is increasing and the other is decreasing.
7 **accept**

Consider running Algorithm 1 on a function f that is ε-far from unate. Let $X = \mu_i$ where i is sampled uniformly at random from $[d]$. Then $\mathbf{E}[X] \geq \frac{\varepsilon}{4d}$. Applying the work investment strategy (Theorem 2.3) on X with $\mu = \frac{\varepsilon}{4d}$, we get that the probability that, in some iteration, Step 3 samples a dimension i such that $\mu_i \geq 2^{-r}$ is at least $1-\delta$. We set $\delta = 1/4$. Conditioned on sampling such a dimension, the probability that Step 4 fails to obtain an increasing edge and a decreasing edge among its $3 \cdot 2^r$ samples is at most $2\left(1 - 1/2^r\right)^{3 \cdot 2^r} \leq 2e^{-3} < 1/9$, as the fraction of both increasing and decreasing edges is at least $1/2^r$. Hence, the probability that Algorithm 1 rejects f is at least $\frac{3}{4} \cdot \frac{8}{9} = \frac{2}{3}$. This completes the proof of Lemma 2.2. ◀

2.2 The Adaptive Tester over the Hypercube

We now describe Algorithm 2, an adaptive tester for unateness over the hypercube domain with good expected query complexity. The final tester is obtained by repeating this tester and accepting if the number of queries exceeds a specified bound.

▶ **Claim 2.4.** *The expected number of queries made by Algorithm 2 is* $40d/\varepsilon$.

Proof. Consider one iteration of the **repeat**-loop in Step 1. We prove that the expected number of queries in this iteration is $4d$. The number of queries in Step 3 is $2d$, as 2 points per dimension are queried. Let E_i be the event that edge e_i is non-constant and T_i be the random variable for the number of i-edges sampled in Step 5. Then $\mathbf{E}[T_i] = \frac{1}{\alpha_i + \beta_i} = \frac{1}{\Pr[E_i]}$. Therefore, the expected number of all edges sampled in Step 5 is $\sum_{i=1}^{d} \Pr[E_i] \cdot \mathbf{E}[T_i] = \sum_{i=1}^{d} \Pr[E_i] \cdot \frac{1}{\Pr[E_i]} = d$. Hence, the expected number of queries in Step 5 is $2d$. Since there are $10/\varepsilon$ iterations in Step 1, the expected number of queries in Algorithm 2 is $40d/\varepsilon$. ◀

▶ **Claim 2.5.** *If f is ε-far from unate, Algorithm 2 accepts with probability at most* $1/6$.

Proof. First, we bound the probability that a violation of unateness is detected in some dimension $i \in [d]$ in one iteration of the **repeat**-loop. Consider the probability of finding a decreasing i-edge in Step 3, and an increasing i-edge in Step 5. The former is exactly α_i, and the latter is $\beta_i/(\alpha_i + \beta_i)$. Therefore, the probability we detect a violation from dimension i is $2\alpha_i\beta_i/(\alpha_i + \beta_i) \geq \min(\alpha_i, \beta_i) = \mu_i$. The probability that we fail to detect a violation in any of the d dimensions is at most $\prod_{i=1}^{d}(1 - \mu_i) \leq \exp\left(-\sum_{i=1}^{d} \mu_i\right)$, which is at most $e^{-\varepsilon/4}$ by Theorem 2.1 (Dimension Reduction). By Taylor expansion of $e^{-\varepsilon/4}$, the probability of finding a violation in one iteration is at least $1 - e^{-\varepsilon/4} \geq \frac{\varepsilon}{4} - \frac{\varepsilon^2}{32} \geq \frac{\varepsilon}{5}$. The probability that Algorithm 2 does not reject in any iteration is at most $(1 - \varepsilon/5)^{10/\varepsilon} < 1/6$. ◀

Proof of Theorem 1.1, part 2 (the special case of the hypercube domain). We run Algorithm 2, aborting and accepting if we ever make more than $240d/\varepsilon$ queries. By Markov's

Algorithm 3: Tree Tester

input : Query access to a function $h : [n] \mapsto \mathbb{R}$.

Pick $x \in [n]$ uniformly at random.
Let $Q_x \subseteq [n]$ be the set of points visited in a binary search for x. Query h on all points in Q_x.
If there is an increasing pair in Q_x, set $\mathsf{dir} \leftarrow \{\uparrow\}$; otherwise, $\mathsf{dir} \leftarrow \emptyset$.
If there is a decreasing pair in Q_x, update $\mathsf{dir} \leftarrow \mathsf{dir} \cup \{\downarrow\}$.
Return dir.

Algorithm 4: The Adaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $10/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ_i uniformly at random.
    Let dir_i be the output of Algorithm 3 on $f_{|\ell_i}$.
    if $\mathsf{dir}_i \neq \emptyset$ **then**
      Sample i-lines uniformly at random and run Algorithm 3 on f restricted to each line until it returns a non-empty set. Call it dir_i'.
      If $\mathsf{dir}_i \cup \mathsf{dir}_i' = \{\uparrow, \downarrow\}$, **reject**.
accept

inequality, the probability of aborting is at most 1/6. By Claim 2.5, if f is ε-far from unate, Algorithm 2 accepts with probability at most 1/6. The theorem follows by a union bound. ◀

2.3 Extension to Hypergrids

We start by establishing terminology for lines and pairs. Consider a function $f : [n]^d \to \mathbb{R}$. Recall the definition of i-lines from Section 1.1.1. A pair of points that differ only in coordinate i is called an i-pair. An i-pair (x, y) with $x_i < y_i$ is called *increasing* if $f(x) < f(y)$, *decreasing* if $f(x) > f(y)$, and *constant* if $f(x) = f(y)$.

The main tool for extending Algorithms 1 and 2 to work on hypergrids is the *tree tester*, designed by Ergun et al. [20] to test monotonicity of functions $h : [n] \to \mathbb{R}$. We modify the tree tester to return information about directions it observed instead of just accepting or rejecting. See Algorithm 3. We call a function $h : [n] \to \mathbb{R}$ *antimonotone* if $f(x) \geq f(y)$ for all $x < y$. The following lemma summarizes the guarantee of the tree tester.

▶ **Lemma 2.6** ([20, 12]). *If $h : [n] \mapsto \mathbb{R}$ is ε-far from monotone (respectively, antimonotone), then the output of Algorithm 3 on h contains $\downarrow$ (respectively, $\uparrow$) with probability at least ε.*

Our hypergrid testers are stated in Algorithms 4 and 5. Next, we explain how Lemma 2.6 and Theorem 2.1 are used in the analysis of the adaptive tester. For a dimension $i \in [d]$, let α_i and β_i denote the average distance of $f_{|\ell}$ to monotonicity and antimonotonicity, respectively, over all i-lines ℓ. Then $\mu_i := \min(\alpha_i, \beta_i)$ is the average fraction of points per i-line that needs to change to make f unate. Define the $\mathbf{b}$-vector with $\mathbf{b}_i = 0$ if $\alpha_i < \beta_i$, and $\mathbf{b}_i = 1$ otherwise. By Theorem 2.1, if f is ε-far from unate, and thus ε-far from $\mathbf{b}$-monotone, then $\sum_{i=1}^{d} \mu_i \geq \varepsilon/4$. By Lemma 2.6, the probability that the output of Algorithm 3 on $f_{|\ell}$ contains

Algorithm 5: The Nonadaptive Unateness Tester over Hypergrids

input : distance parameter $\varepsilon \in (0, 1/2)$; query access to a function $f : [n]^d \to \mathbb{R}$.

repeat $220/\varepsilon$ **times**
  for $i = 1$ *to* d **do**
    Sample an i-line ℓ uniformly at random.
    Reject if Algorithm 3, on input $f_{|\ell}$, returns $\{\uparrow, \downarrow\}$.
for $r = 1$ *to* $\lceil 3 \log(200d/\varepsilon) \rceil$ **do**
  repeat $s_r = \lceil \frac{800d \ln 4}{\varepsilon \cdot 2^r} \rceil$ **times**
    Sample a dimension $i \in [d]$ uniformly at random.
    Sample $3 \cdot 2^r$ i-pairs uniformly and independently at random.
    If we find an increasing and a decreasing pair among the sampled pairs, **reject**.
accept

$\downarrow$ (respectively, $\uparrow$), where ℓ is a uniformly random i-line, is at least α_i (respectively, β_i). The rest of the analysis of Algorithm 4 is similar to that in the hypercube case.

To analyze the nonadaptive tester, we prove Lemma 2.7, which demonstrates the power of the tree tester and may be of independent interest.

▶ **Lemma 2.7.** *Consider a function $h : [n] \to \mathbb{R}$ which is ε-far from monotone (respectively, antimonotone). At least one of the following holds:*

1. $\Pr[$*Algorithm 3, on input h, returns* $\{\uparrow, \downarrow\}] \geq \varepsilon/25$.
2. $\Pr_{u,v \in [n]}[(u, v)$ *is a decreasing (respectively, increasing) pair*$] \geq \varepsilon/25$.

3 The Lower Bound for Nonadaptive Testers over Hypercubes

In this section, we prove Theorem 1.2, which gives a lower bound for nonadaptive unateness testers for functions over the hypercube.

Previous work of [14] on lower bounds for monotonicity testing shows that, for a special class of properties, which includes unateness, it is sufficient to prove lower bounds for *comparison-based testers*. Comparison-based testers base their decisions only on the *order* of the function values at queried points, and not on the values themselves.

We first state the reduction to comparison-based testers from [14]. Let a (t, ε, δ)-tester for a property $\mathcal{P}$ be a 2-sided error t-query tester, with distance parameter ε, that errs with probability at most δ. Consider functions of the form $f : D \to \mathbb{R}$, where D is an arbitrary partial order (in particular the hypergrid/hypercube). A property $\mathcal{P}$ is invariant under monotone transformations if, for all strictly increasing maps $\phi : \mathbb{R} \to \mathbb{R}$ and all functions f, it holds that $\mathsf{dist}(f, \mathcal{P}) = \mathsf{dist}(\phi \circ f, \mathcal{P})$. In particular, unateness is invariant under monotone transformations. The following theorem is implicitly proven in [14]. Specifically, Theorem 2.1 of [14] is stated for monotonicity, but the proof only uses the fact that monotonicity is a property invariant under monotone transformations, so it applies to all such properties.

▶ **Theorem 3.1** (implicit in [21, 14])**.** *Let $\mathcal{P}$ be a property invariant under monotone transformations. Suppose there exists a nonadaptive (resp., adaptive) (t, ε, δ)-tester for $\mathcal{P}$. Then there exists a nonadaptive (resp., adaptive) comparison-based $(t, \varepsilon, 2\delta)$-tester for $\mathcal{P}$.*

Our main lower bound theorem is stated next. In the light of the previous discussion, it implies Theorem 1.2.

▶ **Theorem 3.2.** *Any nonadaptive comparison-based tester for unateness of functions* $f : \{0,1\}^d \to \mathbb{R}$ *must make* $\Omega(d \log d)$ *queries.*

By Theorem 3.1 and Yao's minimax principle [34], it suffices to prove the lower bound for deterministic, nonadaptive, comparison-based testers over a known distribution of functions. It may be useful for the reader to recall the sketch of the main ideas given in Section 1.1.1. For convenience, assume d is a power of 2 and let $d' := d + \log_2 d$. We will focus on proving the lower bound for functions $h : \{0,1\}^{d'} \to \mathbb{R}$, as $d \log d = \Theta(d' \log d')$.

3.1 The Hard Distributions

We partition $\{0,1\}^{d'}$ into d subcubes based on the $\log_2 d$ most significant bits. Specifically, for $i \in [d]$, the i^{th} subcube is defined as $C_i := \{x \in \{0,1\}^{d'} : \mathsf{val}(x_{d'} x_{d'-1} \cdots x_{d+1}) = i-1\}$, where $\mathsf{val}(z_p z_{p-1} \dots z_1) := \sum_{i=1}^{p} z_i 2^{i-1}$ is the integer equivalent of the binary string $z_p z_{p-1} \dots z_1$.

Let $m := d$. We denote the set of indices of the subcube by $[m]$ and the set of dimensions by $[d]$. We use $i, j \in [m]$ to index subcubes and $a, b \in [d]$ to index dimensions. We define a series of random variables, where each subsequent variable may depend on the previous ones:

- k: a number picked uniformly at random from $\{1, 2, \dots, \frac{1}{2}\log_2 d\}$.
- R: a uniformly random subset of $[d]$ of size 2^k.
- r_i: for each $i \in [m]$, r_i is picked from R uniformly and independently at random.
- α_b: for each $b \in [d]$, α_b is picked from $\{-1, +1\}$ uniformly and independently at random.
- β_i: for each $i \in [m]$, β_i is picked from $\{-1, +1\}$ uniformly and independently at random.

We denote by $\boldsymbol{S}$ the tuple $(k, R, \{r_i\})$, also referred to as the *shared randomness*. We use $\boldsymbol{T}$ to refer to the entire set of random variables. Given $\boldsymbol{T}$, define the following functions:

$$f_{\boldsymbol{T}}(x) := \sum_{b \in [d'] \setminus R} x_b 3^b + \alpha_{r_i} \cdot x_{r_i} 3^{r_i}, \text{ where } i \text{ is the subcube with } x \in C_i.$$

$$g_{\boldsymbol{T}}(x) := \sum_{b \in [d'] \setminus R} x_b 3^b + \beta_i \cdot x_{r_i} 3^{r_i}, \text{ where } i \text{ is the subcube with } x \in C_i.$$

The distribution **Yes** generates $f_{\boldsymbol{T}}$ and the distribution **No** generates $g_{\boldsymbol{T}}$.

In all cases, the function restricted to any subcube C_i is linear. Consider some dimension $b \in R$. There can be numerous r_i's that are equal to b. For $f_{\boldsymbol{T}}$, in all of these subcubes, the coefficient of x_{r_i} has the same sign, namely α_{r_i}. For $g_{\boldsymbol{T}}$, the coefficient β_i is potentially different, as it depends on the actual subcube. We write $f \sim \mathcal{D}$ to denote that f is sampled from distribution $\mathcal{D}$.

▶ **Claim 3.3.** *Every function* $f \in \mathsf{supp}(\mathbf{Yes})$ *is unate. A function* $g \sim \mathbf{No}$ *is* $\frac{1}{8}$*-far from unate with probability at least* $9/10$.

3.2 From Functions to Signed Graphs that are Hard to Distinguish

For convenience, denote $x \prec y$ if $\mathsf{val}(x) < \mathsf{val}(y)$. Note that $\prec$ forms a total ordering on $\{0,1\}^{d'}$. Given $x \prec y \in \{0,1\}^{d'}$ and a function $h : \{0,1\}^{d'} \to \mathbb{R}$, define $\mathsf{sgn}_h(x, y)$ to be 1 if $h(x) < h(y)$, 0 if $h(x) = h(y)$, and -1 if $h(x) > h(y)$.

Any deterministic, nonadaptive, comparison-based tester is defined as follows: It makes a set of queries Q and decides whether or not the input function h is unate depending on the $\binom{|Q|}{2}$-comparisons in Q. More precisely, for every pair $(x, y) \in Q \times Q$, $x \prec y$, we insert an edge labelled with $\mathsf{sgn}_h(x, y)$. Let this signed graph be called G_h^Q. Any nonadaptive,

comparison-based algorithm can be described as a partition of the universe of all signed graphs over Q into $\mathcal{G}_Y$ and $\mathcal{G}_N$. The algorithm accepts the function h iff $G_h^Q \in \mathcal{G}_Y$.

Let $\boldsymbol{G}_Y^Q$ be the distribution of the signed graphs G_h^Q when $h \sim \mathbf{Yes}$. Similarly, define $\boldsymbol{G}_N^Q$ when $h \sim \mathbf{No}$. Our main technical theorem is Theorem 3.4, which is proved in Section 3.3.

▶ **Theorem 3.4.** *For small enough $\delta > 0$ and large enough d, if $|Q| \leq \delta d \log d$, then $\|\boldsymbol{G}_Y^Q - \boldsymbol{G}_N^Q\|_{\mathrm{TV}} = O(\delta)$.*

The proof of Theorem 3.4 is naturally tied to the behavior of sgn_h. Ideally, we would like to say that $\mathsf{sgn}_h(x, y)$ is almost identical regardless of whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$. Towards this, we determine exactly the set of pairs (x, y) that potentially differentiate **Yes** and **No**.

▶ **Claim 3.5.** *For all $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$, for all $x \in C_i$ and $y \in C_j$ such that $i < j$, we have $\mathsf{sgn}_h(x, y) = 1$.*

Thus, comparisons between points in different subcubes reveal no information about which distribution h was generated from. Thus the "interesting" pairs that can distinguish whether $h \sim \mathbf{Yes}$ or $h \sim \mathbf{No}$ must lie in the same subcube. The next claim shows a further criterion that is needed for a pair to be interesting. We first define another notation.

▶ **Definition 3.6.** For any setting of the shared randomness $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$, we define $t^i_{\boldsymbol{S}}(x, y)$ to be the most significant coordinate of difference (between x, y) in $([d] \setminus R) \cup \{r_i\}$.

Note that $\boldsymbol{S}$ determines R and $\{r_i\}$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$ and any function, the restriction to C_i is unaffected by the coordinates in $R \setminus r_i$. Thus, $t^i_{\boldsymbol{S}}(x, y)$ is the first coordinate of difference that is influential in C_i.

▶ **Claim 3.7.** *Fix some $\boldsymbol{S}$, subcube C_i, and points $x, y \in C_i$. Let $c = t^i_{\boldsymbol{S}}(x, y)$, and assume $x \prec y$. For any $\boldsymbol{T}$ that extends $\boldsymbol{S}$:*

- *If $c \neq r_i$, then $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = 1$.*
- *If $c = r_i$, $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y) = \alpha_c$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y) = \beta_i$.*

3.3 Proving Theorem 3.4: Good and Bad Events

For a given Q, we first identify certain "bad" values for $\boldsymbol{S}$, on which Q could potentially distinguish between $f_{\boldsymbol{S}}$ and $g_{\boldsymbol{S}}$. We will prove that the probability of a bad $\boldsymbol{S}$ is small for a given Q. Furthermore, we show that Q cannot distinguish between $f_{\boldsymbol{S}}$ and $g_{\boldsymbol{S}}$ for any good $\boldsymbol{S}$. We set up some definitions.

▶ **Definition 3.8.** Given a pair (x, y), define $\mathsf{cap}(x, y)$ to be the 5 most significant coordinates[2] in which they differ. We say (x, y) *captures* these coordinates. For any set $S \subseteq \{0, 1\}^{d'}$, define $\mathsf{cap}(S) := \bigcup_{x,y \in S} \mathsf{cap}(x, y)$ to be the coordinates captured by the set S.

Fix any Q. We set $Q_i := Q \cap C_i$. We define two bad events for $\boldsymbol{S}$.

- Abort Event $\mathcal{A}$: There exist $x, y \in Q$ with $\mathsf{cap}(x, y) \subseteq R$.
- Collision Event $\mathcal{C}$: There exist $i, j \in [d]$ with $r_i = r_j$, $r_i \in \mathsf{cap}(Q_i)$ and $r_j \in \mathsf{cap}(Q_j)$.

If the abort event doesn't occur, then for any pair (x, y), the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. The heart of the analysis lies in Theorem 3.9, which states that the bad events happen rarely. Theorem 3.9 is proved in Section 3.4.

[2] There is nothing special about the constant 5. It just needs to be sufficiently large.

▶ **Theorem 3.9.** *If* $|Q| \leq \delta d \log d$*, then* $\Pr[\mathcal{A} \cup \mathcal{C}] = O(\delta)$.

When neither the abort nor the collision events happen, we say $\boldsymbol{S}$ is good for Q. Next, we show that conditioned on a good $\boldsymbol{S}$, the set Q cannot distinguish $f \sim \mathbf{Yes}$ from $g \sim \mathbf{No}$.

▶ **Lemma 3.10.** *For any signed graph* G *over* Q*,*

$$\Pr_{f \sim \mathbf{Yes}}[G_f^Q = G | \boldsymbol{S} \text{ is good}] = \Pr_{g \sim \mathbf{No}}[G_g^Q = G | \boldsymbol{S} \text{ is good}].$$

Proof Sketch. As stated above, when the abort event doesn't happen, the sign $\mathsf{sgn}_h(x, y)$ is determined by $\mathsf{cap}(x, y)$ for any $h \in \mathsf{supp}(\mathbf{Yes}) \cup \mathsf{supp}(\mathbf{No})$. Furthermore, a pair (x, y) has a possibility of distinguishing (that is, the pair is interesting) only if $x, y \in C_i$ and $r_i \in \mathsf{cap}(x, y)$. Focus on such interesting pairs. For such a pair, both $\mathsf{sgn}_{f_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ are equally likely to be $+1$ or -1. Therefore, to distinguish, we would need two interesting pairs, $(x, y) \in C_i$ and $(x', y') \in C_j$ with $i \neq j$. Note that, when $g \sim \mathbf{No}$, the signs $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x, y)$ and $\mathsf{sgn}_{g_{\boldsymbol{T}}}(x', y')$ are independently set, whereas when $f \sim \mathbf{Yes}$, the signs are either the same when $r_i = r_j$, or independently set. But if the collision event doesn't occur, we have $r_i \neq r_j$ for interesting pairs in different subcubes. Therefore, the probabilities are the same. ◀

Now, we are armed to prove Theorem 3.4.

Proof of Theorem 3.4. Given any subset of signed graphs, $\mathcal{G}$, it suffices to upper bound

$$\left| \Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G}] \right| \leq sum_{\text{good } \boldsymbol{S}} \left| \Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] \right) \right|$$
$$+ \sum_{\text{bad } \boldsymbol{S}} \left| \Pr[\boldsymbol{S}] \cdot \left(\Pr_{f \sim \mathbf{Yes}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] - \Pr_{f \sim \mathbf{No}}[G_f^Q \in \mathcal{G} | \boldsymbol{S}] \right) \right|.$$

The first term of the RHS is 0 by Lemma 3.10. The second term is at most the probability of bad events, which is $O(\delta)$ by Theorem 3.9. ◀

3.4 Bounding the Probability of Bad Events: Proof of Theorem 3.9

We prove Theorem 3.9 by individually bounding $\Pr[\mathcal{A}]$ and $\Pr[\mathcal{C}]$.

▶ **Lemma 3.11.** *If* $|Q| \leq \delta d \log d$*, then* $\Pr[\mathcal{A}] \leq d^{-1/4}$.

Proof. Fix any choice of k (in $\boldsymbol{S}$). For any pair of points $x, y \in Q$, we have $\Pr[\mathsf{cap}(x, y) \subseteq R] \leq (\frac{2^k}{d-5})^5$. Since $d - 5 \geq d/2$ for all $d \geq 10$ and $k \leq (\log_2 d)/2$, the probability is at most $32d^{-5/2}$. For a large enough d, a union bound over all pairs in $Q \times Q$, which are at most $d^2 \log^2 d$ in number, completes the proof. ◀

The collision event is more challenging to bound. Bounding it is the heart of the lower bound. We start by showing that, if each Q_i captures few coordinates, then the collision event has low probability. A critical point is the appearance of $d \log d$ in this bound.

▶ **Lemma 3.12.** *If* $\sum_i |\mathsf{cap}(Q_i)| \leq M$*, then* $\Pr[\mathcal{C}] = O\left(\frac{M}{d \log d}\right)$.

Proof. For any $r \in [d]$, define $A_r := \{j : r \in \mathsf{cap}(Q_j)\}$ to be the set of indices of Q_j's that capture coordinate r. Let $a_r := |A_r|$. Define $n_\ell := |\{r : a_r \in (2^{\ell-1}, 2^\ell]\}|$. Observe that $\sum_{\ell \leq \log_2 d} n_\ell 2^\ell \leq 2 \sum_{r \in [d]} a_r \leq 2M$.

Fix k. For $r \in [d]$, we say the event $\mathcal{C}_r$ occurs if (a) $r \in R$, and (b) there exists $i, j \in [d]$ such that $r_i = r_j = r$, and $r_i \in \mathtt{cap}(Q_i)$ and $r_j \in \mathtt{cap}(Q_j)$. By the union bound, $\Pr[\mathcal{C}|k] \leq \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k]$.

Let us now compute $\Pr[\mathcal{C}_r|k]$. Only sets Q_j's with $j \in A_r$ are of interest, since the others do not capture r. Event $\mathcal{C}_r$ occurs if at least two of these sets have $r_i = r_j = r$. Hence,

$$\begin{aligned}\Pr[\mathcal{C}_r|k] &= \Pr[r \in R] \cdot \Pr[\exists i, j \in A_r : r_i = r_j = r \mid r \in R] \\ &= \frac{2^k}{d} \cdot \sum_{c \geq 2} \binom{a_r}{c} \left(\frac{1}{2^k}\right)^c \left(1 - \frac{1}{2^k}\right)^{a_r - c}. \end{aligned} \tag{1}$$

A fixed r is in R with probability $\binom{d-1}{2^k-1}/\binom{d}{2^k} = \frac{2^k}{d}$. Given that $|R| = 2^k$, the probability that $r_i = r$ is precisely 2^{-k}.

If $a_r \geq \frac{2^k}{4}$, then we simply upper bound (1) by $\frac{2^k}{d}$. For $a_r < \frac{2^k}{4}$, we upper bound (1) by

$$\frac{2^k}{d}\left(1 - \frac{1}{2^k}\right)^{a_r} \sum_{c \geq 2} \left(a_r \cdot \frac{1}{2^k} \cdot \left(1 - \frac{1}{2^k}\right)^{-1}\right)^c \leq \frac{2^k}{d} \sum_{c \geq 2} \left(\frac{a_r}{2^{k-1}}\right)^c \leq \frac{8a_r^2}{2^k d}.$$

Summing over all r and grouping according to n_ℓ, we get

$$\Pr[\mathcal{C}|k] \leq \sum_{r=1}^{d} \Pr[\mathcal{C}_r|k] \leq \sum_{r: a_r \geq 2^{k-2}} \frac{2^k}{d} + \frac{8}{d} \sum_{r: a_r < 2^{k-2}} \frac{a_r^2}{2^k} \leq \frac{2^k}{d} \sum_{\ell > k-2} n_\ell + \frac{8}{d} \sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k}.$$

Averaging over all k, we get

$$\begin{aligned}\Pr[\mathcal{C}] = \frac{2}{\log_2 d} \sum_{k=1}^{(\log_2 d)/2} \Pr[\mathcal{C}|k] \quad &\leq \quad \frac{16}{d \log_2 d} \sum_{k=1}^{(\log_2 d)/2} \left(\sum_{\ell=1}^{k-2} n_\ell 2^{2\ell - k} + \sum_{\ell > k-2} n_\ell 2^k\right) \\ &= \frac{16}{d \log_2 d} \left(\sum_{\ell=1}^{(\log_2 d)/2} n_\ell \sum_{k \geq \ell+2} 2^{2\ell - k} + \sum_{\ell=1}^{\log_2 d} n_\ell \sum_{k < \ell+2} 2^k\right). \end{aligned} \tag{2}$$

Now, $\sum_{k \geq \ell+2} 2^{2\ell-k} \leq 2^\ell$ and $\sum_{k < \ell+2} 2^k \leq 4 \cdot 2^\ell$. Substituting, $\Pr[\mathcal{C}] \leq \frac{80}{d \log_2 d} \sum_{\ell=1}^{\log_2 d} n_\ell 2^\ell \leq \frac{160M}{d \log_2 d}$, proving the lemma. ◀

We are now left to bound $\sum_i |\mathtt{cap}(Q_i)|$. This is done by the following combinatorial lemma.

▶ **Lemma 3.13.** *Let V be a set of vectors over an arbitrary alphabet and any number of dimensions. For any natural number c and $x, y \in V$, let $\mathtt{cap}_c(x, y)$ denote the (set of) first c coordinates at which x and y differ. Then $|\mathtt{cap}_c(V)| \leq c(|V| - 1)$.*

Proof. We construct c different edge-coloured graphs $G_1, \ldots, G_c$ over the vertex set V. For every coordinate $i \in \mathtt{cap}_c(V)$, there must exist at least one pair of vectors x, y such that $i \in \mathtt{cap}_c(x, y)$. Thinking of each $\mathtt{cap}_c(x, y)$ as an ordered set, find a pair (x, y) where i appears "earliest" in $\mathtt{cap}_c(x, y)$. Let the position of i in this $\mathtt{cap}_c(x, y)$ be denoted t. We add edge (x, y) to G_t, and colour it i. Note that the same edge (x, y) cannot be added to G_t with multiple colours, and hence all G_t's are simple graphs. Furthermore, observe that each colour is present only once over all G_t's.

We claim that each G_t is acyclic. Suppose not. Let there be a cycle C and let (x, y) be the edge in C with the smallest colour i. Clearly, $x_i \neq y_i$ since $i \in \mathtt{cap}_c(x, y)$. There must exist another edge (u, v) in C such that $u_i \neq v_i$. Furthermore, the colour of (u, v) is $j > i$. Thus, j is the t^{th} entry in $\mathtt{cap}_c(u, v)$. Note that $i \in \mathtt{cap}_c(u, v)$ and must be the s^{th} entry for some $s < t$. But this means that the edge (u, v) coloured i should be in G_s, contradicting the presence of $(x, y) \in G_t$. ◀

We wrap up the bound now.

▶ **Lemma 3.14.** *If* $|Q| \leq \delta d \log d$, *then* $\Pr[\mathcal{C}] = O(\delta)$.

Proof. Lemma 3.13 applied to each Q_i, yields $\sum_i |\mathtt{cap}(Q_i)| \leq 5|Q_i| = 5|Q|$. An application of Lemma 3.12 completes the proof. ◀

Acknowledgments. We thank Oded Goldreich for useful discussions and Meiram Murzabulatov for participation in initial discussions on this work.

References

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

2 Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and C. Seshadhri. Optimal unateness testers for real-valued functions: Adaptivity helps. *Electronic Colloquium on Computational Complexity (ECCC)*, 2017, 2017. Also appeared as arXiv report 1703.05199 (`https://arxiv.org/abs/1703.05199`). URL: `https://eccc.weizmann.ac.il/report/2017/049/`.

3 Tugkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Inf. Comput.*, 196(1):42–56, 2005.

4 Aleksandrs Belovs and Eric Blais. Quantum algorithm for monotonicity testing on the hypercube. *Theory of Computing*, 11:403–412, 2015.

5 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 1021–1032, 2016.

6 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 164–173, 2014.

7 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.

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

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

10 Jop Briët, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

11 Deeparnab Chakrabarty. Monotonicity testing. In *Encyclopedia of Algorithms*, pages 1352–1356. Springer, 2016.

12 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1809–1828, 2015.

13 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013.

14 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

15 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for boolean functions over the hypercube. *SIAM J. Comput.*, 45(2):461–472, 2016.

16 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $O(n^{1/2})$ non-adaptive queries. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 519–528, 2015.

17 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 286–295, 2014.

18 Kashyap Dixit, Sofya Raskhodnikova, Abhradeep Thakurta, and Nithin M. Varma. Erasure-resilient property testing. In *Proceedings, International Colloquium on Automata, Languages and Processing (ICALP)*, pages 91:1–91:15, 2016.

19 Yevgeny Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. *Proceedings, International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 97–108, 1999.

20 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000.

21 Eldar Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107–116, 2004.

22 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002.

23 Oded Goldreich. Introduction to property testing (working draft), 2015. URL: `http://www.wisdom.weizmann.ac.il/~oded/PDF/pt-v1.pdf`.

24 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.

25 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

26 Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.

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

28 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of Lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013.

29 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 52–58, 2015.

30 Subhash Khot and Igor Shinkar. An $\widetilde{O}(n)$ queries adaptive tester for unateness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016*, pages 37:1–37:7, 2016.

31 Eric Lehman and Dana Ron. On disjoint chains of subsets. *J. Combin. Theory Ser. A*, 94(2):399–404, 2001.

32 Sofya Raskhodnikova. Testing if an array is sorted. In *Encyclopedia of Algorithms*, pages 2219–2222. Springer, 2016.

33 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

34 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.