THEORY OF COMPUTING, Volume 15 (1), 2019, pp. 1–55
www.theoryofcomputing.org

Testing k-Monotonicity

The Rise and Fall of Boolean Functions

Clément L. Canonne[*] Elena Grigorescu[†] Siyao Guo[‡] Akash Kumar[§]
Karl Wimmer

Received March 10, 2017; Revised August 18, 2018; Published April 22, 2019

Abstract: A Boolean *k-monotone* function defined over a finite poset domain $\mathcal{D}$ alternates between the values 0 and 1 at most k times on any ascending chain in $\mathcal{D}$. Therefore, k-monotone functions are natural generalizations of the classical *monotone* functions, which are the 1-*monotone* functions.

Motivated by the recent interest in k-monotone functions in the context of circuit complexity and learning theory, and by the central role that monotonicity testing plays in the context of property testing, we initiate a systematic study of k-monotone functions, in the property testing model. In this model, the goal is to distinguish functions that are k-monotone (or are close to being k-monotone) from functions that are far from being k-monotone.
Our results include the following.

1. We demonstrate a separation between testing k-monotonicity and testing monotonicity, on the hypercube domain $\{0,1\}^d$, for $k \geq 3$;

An extended abstract of this paper appeared in the Proceedings of the 8th Innovations in Theoretical Computer Science conference, 2017 [14].

*Work performed while a graduate student at Columbia University, supported by NSF CCF-1115703 and NSF CCF-1319788.

†Research supported in part by NSF CCF-1649515.

‡Part of the work was done at New York University and Simons Institute, UC Berkeley.

§Research supported in part by NSF CCF-1649515.

ACM Classification: F.2, G.2

AMS Classification: 68Q32, 68W20, 68Q25, 68Q17

Key words and phrases: property testing, Boolean functions, monotonicity, learning

DOI: 10.4086/toc.2019.v015a001

2. We demonstrate a separation between testing and learning on $\{0,1\}^d$, for $k = \omega(\log d)$: testing k-monotonicity can be performed with $\exp(O(\sqrt{d} \cdot \log d \cdot \log(1/\varepsilon)))$ queries, while learning k-monotone functions requires $\exp(\Omega(k \cdot \sqrt{d} \cdot 1/\varepsilon))$ queries (Blais et al. (RANDOM 2015));

3. We present a tolerant test for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with complexity independent of n. The test implies a tolerant test for monotonicity of functions $f\colon [n]^d \to [0,1]$ in ℓ_1 distance, which makes progress on a problem left open by Berman et al. (STOC 2014).

Our techniques exploit the testing-by-learning paradigm, use novel applications of Fourier analysis on the grid $[n]^d$, and draw connections to distribution testing techniques.

1 Introduction

A function $f\colon \mathcal{D} \to \{0,1\}$, defined over a finite partially ordered domain $(\mathcal{D}, \preceq)$ is said to be *k-monotone*, for some integer $k \geq 0$, if there does not exist $x_1 \preceq x_2 \preceq \cdots \preceq x_{k+1}$ in $\mathcal{D}$ such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$ for all $i \in [k]$. Note that 1-monotone functions are the classical *monotone* functions, satisfying $f(x_1) \leq f(x_2)$, whenever $x_1 \preceq x_2$.

Monotone functions have been well-studied on multiple fronts in computational complexity due to their natural structure. They have been studied for decades in the property testing literature [28, 22, 26, 12, 16, 18, 17], where we have recently witnessed exciting results [37, 19, 7], in the circuit complexity literature, where we now have strong lower bounds [47, 48], and in computational learning, where we now have learning algorithms in numerous learning models [13, 4, 36, 50, 44, 45].

The generalized notion of k-monotonicity has also been studied in the context of circuit lower bounds for more than 50 years. In particular, Markov [40] showed that any k-monotone function (even with multiple outputs) can be computed using circuits containing only $\log k$ negation gates. The presence of negation gates appears to be a challenge in proving circuit lower bounds: "the effect of such gates on circuit size remains to a large extent a mystery" [33]. The recent results of Blais *et al.* [11] on learning lower bounds have prompted renewed interest in understanding k-monotone functions from multiple angles, including cryptography, circuit complexity, learning theory, and Fourier analysis ([49, 31, 30, 39]).

Motivated by the exponential lower bounds on PAC learning k-monotone functions due to [11], we initiate the study of k-monotonicity in the closely related *Property Testing* model. In this model, given query access to a function, one must decide if the function is k-monotone, or is far from being k-monotone, by querying the input only in a small number of places.

1.1 Our results

We focus on testing k-monotonicity of Boolean functions defined over the d-dimensional hypergrid $[n]^d$, and the hypercube $\{0,1\}^d$. We begin our presentation with the results for the hypercube, in order to build intuition about the difficulty of the problem, while comparing our results with the current literature on testing monotonicity. Our stronger results concern the hypergrid $[n]^d$.

1.1.1 Testing k-monotonicity on the hypercube $\{0,1\}^d$

In light of the recent results of [37] that provide an (optimal non-adaptive one-sided) $\widetilde{O}(\sqrt{d})$-query tester for monotonicity, we first show that testing k-monotonicity is strictly harder than testing monotonicity on $\{0,1\}^d$, for $k \geq 3$.

Theorem 1.1. *For $1 \leq k \leq d^{1/4}/2$, any one-sided non-adaptive tester for k-monotonicity of functions $f\colon \{0,1\}^d \to \{0,1\}$ must make $\Omega\big(d/k^2\big)^{k/4}$ queries.*

Both Theorem 1.1 and its proof generalize the $\Omega(d^{1/2})$ lower bound for testing monotonicity, due to Fischer *et al.* [26].[1]

On the upper bounds side, while the monotonicity testing problem provides numerous potential techniques for approaching this new problem [28, 22, 18, 12, 20, 37], most common techniques appear to resist generalizations to k-monotonicity. However, our upper bounds demonstrate a separation between testing and PAC learning k-monotonicity, for large enough values of $k = \omega(\log d)$.

Theorem 1.2. *There exists a one-sided non-adaptive tester for k-monotonicity of functions $f\colon \{0,1\}^d \to \{0,1\}$ with query complexity*

$$q(d,\varepsilon,k) = \exp\left(O\left(\sqrt{d}\cdot \log d \cdot \log \frac{1}{\varepsilon}\right)\right).$$

Indeed, in the related PAC learning model, [11] shows that learning k-monotone functions on the hypercube requires $\exp(\Omega(k\cdot\sqrt{d}\cdot 1/\varepsilon))$ many queries.

We further observe that the recent non-adaptive and adaptive two-sided lower bounds of [19, 7] imply the same bounds for k-monotonicity, using black box reductions. In Table 1, we summarize our results on testing k-monotonicity over hypercube and best bounds on monotonicity testing (see Section 1.4.2) for comparison.

	upper bound	1. s.-n. a. lower bound	1. s.-n. a. lower bound	1. s.-a. lower bound
$k=1$	$\widetilde{O}\big(d^{1/2}/\varepsilon^2\big)$ [37]	$\Omega\big(d^{1/2}\big)$ [26]	$\Omega\left(d^{1/2-o(1)}\right)$ [19]	$\widetilde{\Omega}\big(d^{1/3}\big)$ [21]
$k\geq 2$	$d^{O\left(\sqrt{d}\cdot\log(1/\varepsilon)\right)}$ Thm 1.2	$\Omega\big(d/k^2\big)^{k/4}$ Thm 1.1	$\Omega\left(d^{1/2-o(1)}\right)$ Cor 4.7	$\widetilde{\Omega}\big(d^{1/3}\big)$ Cor 4.7

Table 1: Testing k-monotonicity of a function $f\colon \{0,1\}^d \to \{0,1\}$.

1.1.2 Testing k-monotonicity on the hypergrid $[n]^d$

In the remainder of the paper we discuss functions defined over the d-dimensional hypergrid domain $[n]^d$, where we denote by $(i_1,i_2,\ldots,i_d) \preceq (j_1,j_2,\ldots,j_d)$ the partial order in which $i_1 \leq j_1, i_2 \leq j_2, \ldots, i_d \leq j_d$. Testing monotonicity for functions over hypergrids has received a lot of attention [28, 23, 25, 6, 1, 32,

[1] More specifically, our theorem generalizes the *first* half of Fischer et al.'s argument, before their spanning tree improvement, which yields an $\Omega(d^{1/2})$ lower bound for monotonicity (instead of $\Omega(d^{1/4})$). The reason for this is that this second part does not extend beyond monotonicity, i. e., to $k \geq 2$.

10, 16, 18, 17, 8]. As these papers demonstrate, the problem is well-understood. We refer the reader to Table 4 in Section 1.4.2 for a detailed review on the state of the art in the area. We summarize our results on testing k-monotonicity over $[n]^d$ in Table 2.

	General k	$k=2$	$k=1$ (monotonicity)
$d=1$	$\Theta\left(\frac{k}{\varepsilon}\right)$ 1. s.-n. a., $\widetilde{O}\left(\frac{1}{\varepsilon^7}\right)$ 1. s.-n. a.	$O\left(\frac{1}{\varepsilon}\right)$ 1. s.-n. a.	$\Theta\left(\frac{1}{\varepsilon}\right)$ 1. s.-n. a.
$d=2$	$\widetilde{O}\left(\frac{k^2}{\varepsilon^4}\right)$ 1. s.-n. a. (from the row below)	$\Theta\left(\frac{1}{\varepsilon}\right)$ 1. s.-a.	$\Theta\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1. s.-n. a., $\Theta\left(\frac{1}{\varepsilon}\right)$ 1. s.-a.
$d\geq 3$	$\widetilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\right)$ 1. s.-n. a., $\exp\left(\widetilde{O}\left(k\sqrt{d}/\varepsilon^2\right)\right)$ 1. s.-n. a.	$\widetilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{10d}{\varepsilon}\right)^d\right)$ 1. s.-n. a., $\exp\left(\widetilde{O}\left(\sqrt{d}/\varepsilon^2\right)\right)$ 1. s.-n. a.	$O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$ 1. s.-n. a.

Table 2: Summary of our results: testing k-monotonicity of a function $f\colon [n]^d \to \{0,1\}$ (first two columns). The last column contains known bounds on monotonicity testing (see Section 1.4.2) and is provided for comparison.

Testing k-monotonicity on the line and the 2-dimensional grid. We begin with a study of functions $f\colon [n] \to \{0,1\}$. Note that one-sided testers should always accept k-monotone functions, and so, they must accept unless they discover a violation to k-monotonicity in the form of a sequence $x_1 \preceq x_2 \preceq \cdots \preceq x_{k+1}$ in $[n]^d$, such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$. Therefore, lower bounds for one-sided k-monotonicity testing must grow at least linearly with k. We show that an even better lower bound of $\Omega(k/\varepsilon)$ holds for both adaptive and non-adaptive testers, and moreover, we give a tight non-adaptive algorithm. Consequently, our results demonstrate that adaptivity does not help in testing k-monotonicity with one-sided error on the line domain.

Theorem 1.3. *Any one-sided (possibly adaptive) tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ must have query complexity $\Omega(k/\varepsilon)$.*

The upper bound generalizes the $O(1/\varepsilon)$ tester for monotonicity on the line.

Theorem 1.4. *There exists a one-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n,\varepsilon,k) = O(k/\varepsilon)$.*

Testing with two-sided error, however, does not require a dependence on k. In fact the problem has been well-studied in the machine learning literature in the context of testing/learning "union of intervals" [35, 5], and in testing geometric properties, in the context of testing surface area [38, 42],[2] resulting in an $O(1/\varepsilon^{7/2})$-query algorithm. Namely, the starting point of [5] (later improved by [38]) is a "Buffon's Needle"-type argument. There, the crucial quantity to analyze is the noise sensitivity of the function, that is the probability that a randomly chosen pair of nearby points cross a "boundary," i. e., have different values. (Moreover, the algorithm of [5] works in the *active testing* setting: it only requires a weaker access model that the standard query model.)

[2] We thank Eric Blais for mentioning the connection, and pointing us to these articles.

We provide an alternate proof of a $\mathrm{poly}(1/\varepsilon)$ bound (albeit with a worse exponent) that reveals a surprising connection with *distribution testing*, namely with the problem of estimating the support size of a distribution.

Theorem 1.5. *There exists a two-sided non-adaptive tester for k-monotonicity of functions* $f\colon [n] \to \{0,1\}$ *with query complexity* $q(n,\varepsilon,k) = \widetilde{O}(1/\varepsilon^7)$*,* independent of *k*.

An immediate implication of Theorem 1.5 is that one can test even $n^{1-\alpha}$-monotonicity of $f\colon [n] \to \{0,1\}$, for every $\alpha > 0$, with a constant number of queries. Hence, there is a separation between one-sided and two-sided testing, for $k = \omega(1)$.

Turning to the 2-dimensional grid, we show that 2-monotone functions can be tested with the minimum number of queries one could hope for:

Theorem 1.6. *There exists a two-sided adaptive tester for* 2*-monotonicity of functions* $f\colon [n]^2 \to \{0,1\}$ *with query complexity* $q(n,\varepsilon) = O(1/\varepsilon)$*.*

We also discuss possible generalizations of Theorem 1.6 to general k or d in Section 6.2.

Testing k-monotonicity on $[n]^d$, tolerant testing, and distance approximation. Moving to the general grid domain $[n]^d$, we show that k-monotonicity is testable with $\mathrm{poly}(1/\varepsilon,k)$ queries in constant-dimension grids.

Theorem 1.7. *There exists a non-adaptive two-sided tester for k-monotonicity of functions* $f\colon [n]^d \to \{0,1\}$ *with query complexity*

$$q(n,d,\varepsilon,k) = \min\left(\widetilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\right), \exp\left(\widetilde{O}\left(k\sqrt{d}/\varepsilon^2\right)\right)\right).$$

In fact, we obtain more general testing algorithms than in Corollary 1.7, namely our results hold for *tolerant* testers (as we define next).

The notion of tolerant testing was first introduced in [46] to account for the possibility of noisy data. In this notion, a test should accept inputs that are ε_1-close to the property, and reject inputs that are ε_2-far from the property, where ε_1 and ε_2 are given parameters. Tolerant testing is intimately connected to the notion of distance approximation: given tolerant testers for every $(\varepsilon_1,\varepsilon_2)$, there exists an algorithm that estimates the distance to the property within any (additive) ε, while incurring only a $\widetilde{O}(\log(1/\varepsilon))$ factor blow up in the number of queries. Furthermore, [46] shows that both tolerant testing and distance approximation are no harder than agnostic learning. We prove the following general result.

Theorem 1.8. *There exist*

- *a non-adaptive (fully) tolerant tester for k-monotonicity of functions* $f\colon [n]^d \to \{0,1\}$ *with query complexity*

$$q(n,d,\varepsilon_1,\varepsilon_2,k) = \widetilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2-\varepsilon_1}\right)^d\right);$$

- *a non-adaptive tolerant tester for k-monotonicity of functions $f\colon [n]^d \to \{0,1\}$ with query complexity*
$$q(n,d,\varepsilon_1,\varepsilon_2,k) = \exp\left(\widetilde{O}\left(k\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)\right),$$
under the restriction that $\varepsilon_2 > 3\varepsilon_1$.

To the best of our knowledge, the only previous results for tolerant testing for monotonicity on $[n]^d$ are due to Fattal and Ron [24]. They give both additive and multiplicative distance approximations algorithms, and obtain $O(d)$-multiplicative and ε-additive approximations with query complexity $\operatorname{poly}(1/\varepsilon)$. While very efficient, their results only give fully tolerant testers for dimensions $d=1$ and $d=2$. Our results generalize results of [24] by showing the existence of tolerant testers for k-monotonicity (and hence for monotonicity) for any dimension $d \geq 1$, and any $k \geq 1$, but paying the price in the query complexity.

As a consequence to Theorem 1.8, we make progress on an open problem of Berman *et al.* [8], as explained next.

Testing k-monotonicity under L_p distance. The property of being a monotone Boolean function has a natural extension to real-valued functions. Indeed, a real-valued function defined over a finite domain D is monotone if $f(x) \leq f(y)$ whenever $x \preceq y$. For real-valued functions the more natural notion of distance is L_p distance, rather than Hamming distance. The study of monotonicity has been extended to real-valued functions in a recent article by Berman *et al.* [8]. They give tolerant testers for grids of dimension $d=1$ and $d=2$, and leave open the problem of extending the results to general d, as asked explicitly at the recent Sublinear Algorithms Workshop 2016 [52].

We make progress towards solving this open problem, by combining our Theorem 1.8 with a reduction from L_p testing to Hamming testing inspired by [8]. This reduction relates L_1-distance to monotonicity of a function $f\colon [n]^d \to [0,1]$ to *Hamming* distance to monotonicity of a "rounded" function $\widetilde{f}\colon [n]^d \times [m] \to \{0,1\}$, essentially trading the range for an extra dimension (where m is a rounding parameter to be suitably chosen). Moreover, simulating query access to $\widetilde{f}$ can be performed efficiently given query access to f.

Theorem 1.9. *There exists a non-adaptive tolerant L_1-tester for monotonicity of functions $f\colon [n]^d \to [0,1]$ with query complexity*

- $\widetilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5d}{\varepsilon_2-\varepsilon_1}\right)^d\right)$*, for any $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$;*
- $\exp\left(\widetilde{O}\left(\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)\right)$*, for any $0 \leq 3\varepsilon_1 < \varepsilon_2 \leq 1$.*

1.2 Proof overview and technical contribution

Structural properties and the separation between testing and learning on $\{0,1\}^d$. We first observe that basic structural properties, such as *extendability* (i. e., the feature that a function that is monotone on a sub-poset of $[n]^d$ can be extended into a monotone function on the entire poset domain), and properties of the *violation graph* (i. e., the graph whose edges encode the violations to monotonicity), extend easily to k-monotonicity (see Section 3). These properties help us to argue the separation between testing and learning (Theorem 1.2). However, unlike the case of monotonicity testing, these properties do not seem to be enough for showing upper bounds that grow polynomially in d.

Grid coarsening and testing by implicit/explicit learning. One technique, which underlies all the hypergrid upper bounds in this article is that of *gridding*: i. e., partitioning the domain into "blocks" whose size no longer depends on the main parameter of the problem, n. This technique generalizes the approach of [24] who performed a similar gridding for dimension $d = 2$. By simulating query access to the "coarsened" version of the unknown function (with regard to these blocks), we are able to leverage methods such as testing-by-learning (either fully or partially learning the function), or reduce our testing problem to a (related) question on these nicer "coarsenings." (The main challenge here lies in providing efficient and consistent oracle access to the said coarsenings.)

At a high-level, the key aspect of k-monotonicity which makes this general approach possible is reminiscent of the concept of *heredity* in property testing. Specifically, we rely upon the fact that "gridding preserves k-monotonicity": if f *is* k-monotone, then so will be its coarsening g—but now g is much simpler to handle. This allows us to trade the domain $[n]^d$ for what is effectively $[m]^d$, with $m \ll n$. We point out that this differs from the usual paradigm of *dimension reduction*: indeed, the latter would reduce the study of a property of functions on $[n]^d$ to that of functions on $[n]^{d'}$ for $d' \ll d$ (usually even $d' = 1$) by projecting f on a lower-dimensional domain. In contrast, we do not take the dimension down, but instead reduce the size of the *alphabet*. Moreover, it is worth noting that this gridding technique is also orthogonal to that of *range reduction*, as used, e. g., in [22]. Indeed, the latter is a reduction of the range of the function from $[R]$ to $\{0,1\}$, while gridding is only concerned about the domain size.

Estimating the support of distributions. Our proof of the $\mathrm{poly}(1/\varepsilon)$ upper bound for testing k-monotonicity on the line (Theorem 1.5) rests upon an unexpected connection to *distribution testing*, namely to the question of support size estimation of a probability distribution. In more detail, we describe how to reduce k-monotonicity testing to the support size estimation problem in (a slight modification of) the *Dual access model* introduced by Canonne and Rubinfeld [15], where the tester is granted samples from an unknown distribution as well as query access to its probability mass function.

For our reduction to go through, we first describe how any function $f\colon [n] \to \{0,1\}$ determines a probability distribution D_f (on $[n]$), whose effective support size is directly related to the k-monotonicity of f. We then show how to implement dual access to this D_f from queries to f: in order to avoid any dependence on k and n in this step, we resort both to the gridding approach outlined above (allowing us to remove n from the picture) and to a careful argument to "cap" the values of D_f returned by our simulated oracle. Indeed, obtaining the exact value of $D_f(x)$ for arbitrary x may require $\Omega(k)$ queries to f, which we cannot afford; instead, we argue that only returning $D_f(x)$ whenever this value is "small enough" is sufficient. Finally, we show that implementing this "capped" dual access oracle is possible with no dependence on k whatsoever, and we can now invoke the support size estimation algorithm of [15] to conclude.

Fourier analysis on the hypergrid. We give an algorithm for fully tolerantly testing k-monotonicity whose query complexity is exponential in d. We also describe an alternate tester (with a slightly worse tolerance guarantee) whose query complexity is instead exponential in $\widetilde{O}(k\sqrt{d})$ for constant distance parameters. As mentioned above, we use our gridding approach combined with tools from learning theory. Specifically, we employ an agnostic learning algorithm of [34] using polynomial regression. Our coarsening methods allow us to treat the domain as if it were $[m]^d$ for some m that is independent of n. To prove that this agnostic learning algorithm will succeed, we turn to Fourier analysis over $[m]^d$. We extend the bound on average sensitivity of k-monotone functions over the Boolean hypercube from [11] to the hypergrid, and we show that this result implies that the Fourier coefficients are concentrated on "simple" functions.

1.3 Discussion and open problems

This is the first article to study k-monotonicity in the property testing setting, a natural and well-motivated generalization of monotonicity. Hence these results open up many intriguing questions in the area of property testing, with potential applications to learning theory, circuit complexity and cryptography.

A natural question emerging from our results here is whether k-monotonicity on the hypercube $\{0,1\}^d$ can be tested with $\operatorname{poly}(d^k)$ queries, as one may guess by "smoothly" generalizing monotonicity. As it turns out, this is actually not the case: subsequent results [29] show that even testing 2-monotonicity, non-adaptively and with one-sided error, requires $2^{\Omega(\sqrt{d})}$ many queries.

Another important open question concerns the hypergrid domain, and in particular it pushes for a significant strengthening of Corollary 1.7 and Corollary 1.9:

Can k-monotonicity on the hypergrid $[n]^d$ be (tolerantly) tested with $2^{o_k(\sqrt{d})}$ queries?

Answering this question would imply further progress on the L_1-testing question for monotonicity, left open in [8, 52].

There also remains the question of establishing two-sided lower bounds that would go beyond those of monotonicity. Specifically:

Is there an $d^{\Omega(k)}$-query two-sided *lower bound for k-monotonicity on the hypercube $\{0,1\}^d$?*

In this article we also show surprising connections to distribution testing (e. g., in the proof of Theorem 1.5), and to testing union of intervals and testing surface area (as discussed in Section 5.2). An intriguing direction is to generalize this connection to union of intervals and surface area in higher dimensions, to leverage or gain insight on k-monotonicity on the d-dimensional hypergrid.

Finally, while we only stated here a few directions, we emphasize that every question that is relevant to monotonicity is relevant and interesting in the case of k-monotonicity as well.

1.4 Related work

1.4.1 Previous work on k-monotonicity

As mentioned, k-monotonicity has deep connections with the notion of *negation complexity* of functions, which is the minimum number of negation gates needed in a circuit to compute a given function. The power of negation gates is intriguing and far from being understood in the context of circuit lower bounds.

Quoting from Jukna's book [33], *the main difficulty in proving nontrivial lower bounds on the size of circuits using* AND*,* OR*, and* NOT *is the presence of* NOT *gates: we already know how to prove even exponential lower bounds for monotone functions if no* NOT *gates are allowed. The effect of such gates on circuit size remains to a large extent a mystery.*

This gap has motivated the study of circuits with *few* negations. Two notable works successfully extend lower bounds in the monotone setting to negation-limited setting: in [3], Amano and Maruoka show superpolynomial circuit lower bounds for $(1/6)\log\log n$ negations using the CLIQUE function; and recently the breakthrough results of Rossman [49] establishes circuit lower bounds for NC^1 with roughly $(1/2)\log n$ negations by drawing upon his lower bound for monotone NC^1.

The divide between the understanding of monotone and non-monotone computation exists in general: while we usually have a fairly good understanding of the monotone case, many things get murky or fail to hold even when a single negation gate is allowed. In order to get a better grasp on negation-limited circuits, a body of recent work has been considering this model in various contexts: Blais *et al*. [11] study negation-limited circuits from a computational learning viewpoint, Guo *et al*. [31] study the possibility of implementing cryptographic primitives using few negations, and Lin and Zhang [39] are interested in verifying whether some classic Boolean function conjectures hold for the subset of functions computed by negation-limited circuits.

Many of these results implicitly or explicitly rely on a simple but powerful tool: the decomposition of negation-limited circuits into a composition of some "nice" function with monotone components. Doing so enables one to apply results on separate monotone components, and finally to carefully combine the outcomes (e. g., [30]). Though these techniques can yield results for as many as $O(\log n)$ negations, they also leave open surprisingly basic questions:

- [11] Is there an efficient algorithm that weakly learns circuits with a *single* negation?
- [31] Is there a pseudorandom generator computed with a *single* negation gate?

In contexts where the circuit size is not the quantity of interest, the equivalent notion of 2-monotone functions is more natural than that of circuits allowing only one negation. Albeit seemingly simple, even the class of 2-monotone functions remains largely a mystery: as exemplified above, many basic yet non-trivial questions, ranging from the structure of their Fourier spectrum to their expressive power of k-monotone functions, remain open.

1.4.2 Previous work on monotonicity testing

In this section, we summarize the state-of-the-art on monotonicity testing. We observe that this question has been considered for functions over various domains (e. g., hypergrids, hypercubes and general posets) and ranges (notably Boolean range $\{0,1\}$ and unbounded range $\mathbb{N}$); as hypergrids and hypercubes are arguably the domains that have received the most attention in the literature, we will in this overview restrict ourselves to work on these, and refer readers to [26, 9] for other various posets. We will also focus on the Boolean range $\{0,1\}$, which is most relevant to our work. We then briefly mention the best known results (which are tight as well) for unbounded range $\mathbb{N}$. At the end of this section, we include known results for *tolerant* monotonicity testing.

Before we go over those results, we recall some notation: namely, testers can make adaptive (a.) or non-adaptive (n. a.) queries and have one-sided (1. s.) or two-sided (1. s.) error. The best one could hope for would then be to obtain one-sided non-adaptive upper bounds complemented with two-sided adaptive lower bounds. We note that all testers included in below except tolerant testers are one-sided (almost all of them are non-adaptive) algorithms.

Hypercubes with Boolean Range. The problem of monotonicity testing is introduced by Goldreich *et al.* [28] for functions $f\colon \{0,1\}^d \to \{0,1\}$. [28] present a simple "edge tester" with query complexity $O(d/\varepsilon)$. A tester with $O(d^{7/8}/\varepsilon^{3/2})$ queries, the first improvement in terms of the dependence on d and the first to "break" the linear barrier, was presented by Chakrabarty and Seshadhri [18], further improved to $\widetilde{O}(d^{5/6}/\varepsilon^4)$ by Chen *et al.* [20]. Recently, a $\widetilde{O}(\sqrt{d}/\varepsilon^2)$ upper bound was established by Khot *et al.* [37] which is optimal (for non-adaptive one-sided testers) up to logarithmic factors. All these upper bounds are obtained for one-sided, non-adaptive testers.

For one-sided non-adaptive testers, Fischer *et al.* [26] showed an $\Omega(\sqrt{d})$ lower bound. For two-sided non-adaptive testers, Chen *et al.* [20] obtained an $\widetilde{\Omega}(d^{1/5})$ lower bound, further improved by Chen *et al.* [19]) to $\Omega(d^{1/2-c})$ (for any constant $c > 0$). All these lower bounds applying to non-adaptive testers, they only imply an $\Omega(\log d)$ lower bound for adaptive ones. Recently, Belovs and Blais [7] showed an $\widetilde{\Omega}(d^{1/4})$ lower bound for two-sided adaptive testers, i. e., an exponential improvement over the previous bounds, further improved by Chen, Waingarten, and Xie [21] to $\widetilde{\Omega}(d^{1/3})$. All mentioned lower bounds hold for constant $\varepsilon > 0$, and are summarized in Table 3.

Domain	Upper bound	Lower bound
$\{0,1\}^d$	$\widetilde{O}\big(d^{1/2}/\varepsilon^2\big)$ 1. s.-n. a. [37]	$\Omega\big(d^{1/2}\big)$ 1. s.-n. a. [26] $\Omega\big(d^{1/2-o(1)}\big)$ 1. s.-n. a. [19] $\widetilde{\Omega}\big(d^{1/3}\big)$ 1. s.-a. [21]

Table 3: Testing monotonicity of a function $f\colon \{0,1\}^d \to \{0,1\}$.

Hypergrids with Boolean Range. We remark that most known previous upper bounds for testing monotonicity over hypergrids are for unbounded range, which we will be the focus of the next section. Instead, we only mention here the case of Boolean range, giving in each setting the current best known results. For testing monotonicity over the line with Boolean range (i. e., $d = 1$ case), both a one-sided non-adaptive $O(1/\varepsilon)$ upper bound and a two-sided adaptive $\Omega(1/\varepsilon)$ lower bound are known (both of them being folklore). For $d = 2$, Berman *et al.* [8] showed a tight bound of $\Theta((\log 1/\varepsilon)/\varepsilon)$ for one-sided non-adaptive testers. Interestingly, they also prove that "adaptivity" helps in the $d = 2$ case: that is, they establish a one-sided tight adaptive $O(1/\varepsilon)$ upper bound which beats $\Omega(\log 1/\varepsilon)/\varepsilon)$ lower bound for one-sided non-adaptive testers. For general d, Berman *et al.* [8] give both a one-sided non-adaptive tester with query complexity $O(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon})$, and a one-sided adaptive tester with query complexity

$$O\left(d2^d \log^{d-1}\frac{1}{\varepsilon} + \frac{d^2\log d}{\varepsilon}\right).$$

The best known results can be found in Table 4.

Domain	Upper bound	Lower bound
$[n]$	$O\left(\frac{1}{\varepsilon}\right)$ 1. s.-n. a.	$\Omega\left(\frac{1}{\varepsilon}\right)$ 1. s.-a.
$[n]^2$	$O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1. s.-n. a. [8]	$\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1. s.-n. a. [8]
	$O\left(\frac{1}{\varepsilon}\right)$ 1. s.-a. [8]	$\Omega\left(\frac{1}{\varepsilon}\right)$ 1. s.-a.
$[n]^d$	$O\left(\frac{d}{\varepsilon}\log\frac{d}{\varepsilon}\right)$ 1. s.-n. a. [8]	$\Omega\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1. s.-n. a. [8]
	$O\left(d2^d\log^{d-1}\frac{1}{\varepsilon}+\frac{d^2\log d}{\varepsilon}\right)$ 1. s.-a. [8]	$\Omega\left(\frac{1}{\varepsilon}\right)$ 1. s.-a.

Table 4: Testing monotonicity of a function $f\colon [n]^d \to \{0,1\}$.

Unbounded Range. For unbounded range, tight upper and lower bounds are known for both hypergrid and hypercube domains. Chakrabarty and Seshadhri [17] describe a one-sided non-adaptive tester with $O(d\log n/\varepsilon)$ queries for the hypergrid $[n]^d$. Later, they show that $O(d\log n/\varepsilon)$ is essentially optimal even for two-sided adaptive tester [16]. For the hypercube, [17] give a one-sided non-adaptive tester making $O(n/\varepsilon)$ queries, and a matching two-sided adaptive lower bound is proved by Joshua Brody (mentioned as private communication in [16]). We refer readers to [17, 16] for overviews on previous results for testing monotonicity over the hypercube and hypergrid with unbounded range. The best known results are summarized in Table 5.

Domain	Upper bound	Lower bound
$\{0,1\}^d$	$O\left(\frac{d}{\varepsilon}\right)$ 1. s.-n. a. [17]	$\Omega\left(\frac{d}{\varepsilon}\right)$ 1. s.-a. [16]
$[n]^d$	$O\left(\frac{d\log n}{\varepsilon}\right)$ 1. s.-n. a. [17]	$\Omega\left(\frac{d\log n}{\varepsilon}-\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ 1. s.-a. [16]

Table 5: Testing monotonicity of a function $f\colon D \to \mathbb{N}$.

Tolerant Testing. To the best of our knowledge, prior to our results tolerant testers for monotonicity for Boolean functions over the hypergrid were only known for dimension $d \in \{1,2\}$. Specifically, an $O(\varepsilon_2/(\varepsilon_2-\varepsilon_1)^2)$-query upper bound is known for $d=1$, while an $\widetilde{O}\left(1/(\varepsilon_2-\varepsilon_1)^4\right)$-query one is known for $d=2$ [8, 24].

Domain	Upper bound	Lower bound
$[n]$	$O\left(\frac{\varepsilon_2}{(\varepsilon_2-\varepsilon_1)^2}\right)$ [8] 1. s.-a.	?
$[n]^2$	$\widetilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^4}\right)$ [24] 1. s.-n. a.	?

Table 6: Tolerant testing monotonicity of a function $f\colon [n]^d \to \{0,1\}$.

1.5 Organization of the paper

After recalling some notations and definitions in Section 2 and providing some insight into the structural properties of functions far from k-monotonicity in Section 3, we consider the case of the Boolean hypercube in Section 4, where we establish lower bounds on testing k-monotonicity of functions

$f\colon \{0,1\}^d \to \{0,1\}$ for both one- and two-sided algorithms, and provide an algorithm which "beats" the testing-by-learning approach, showing that testing is provably easier than learning.

Next, we establish our results for functions on the line in Section 5, starting with the lower and upper bounds for one-sided testers before turning in Section 5.2 to the two-sided upper bound of Theorem 1.3. We then describe in Section 6 our results for functions on the grid $[n]^2$, focusing on the case $k = 2$; and discussing possible extensions in Section 6.2.

Section 7 contains our general algorithms for k-monotonicity on the hypergrid $[n]^d$, for arbitrary k and d. We prove Theorem 1.8 in two parts. We establish its first item (general tolerant testing algorithm with exponential dependence in d) in Section 7.1 (Proposition 7.2). The second item (with query complexity exponential in $k\sqrt{d}$) is proven in Section 7.2, where we analyze the Fourier-based tolerant tester of Proposition 7.11. We then apply these results to the question of tolerant L_1-testing of monotonicity in Section 8, after describing a reduction between monotonicity of functions $[n]^d \to [0,1]$ and of $[n]^{d+1} \to \{0,1\}$.

Except maybe Section 8 which depends on Section 7, all sections are independent and self-contained, and the reader may choose to read them in any order.

2 Preliminaries

We denote by log the binary logarithm, and use $\widetilde{O}(\cdot)$ to hide polylogarithmic factors in the argument (so that $\widetilde{O}(f) = O(f \log^c f)$ for some $c \geq 0$).

Given two functions $f, g\colon \mathcal{X} \to \mathcal{Y}$ on a finite domain $\mathcal{X}$, we write $\operatorname{dist}(f,g)$ for the (normalized) Hamming distance between them, i. e.,

$$\operatorname{dist}(f,g) = \frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} \mathbb{1}_{\{f(x)\neq g(x)\}} = \Pr_{x\sim\mathcal{X}}[\,f(x) \neq g(x)\,]$$

where $x \sim \mathcal{X}$ refers to x being drawn from the uniform distribution on $\mathcal{X}$. A *property* of functions from $\mathcal{X}$ to $\mathcal{Y}$ is a subset $\mathcal{P} \subseteq \mathcal{X}^{\mathcal{Y}}$ of these functions; we define the distance of a function f to $\mathcal{P}$ as the minimum distance of f to any $g \in \mathcal{P}$:

$$\operatorname{dist}(f,\mathcal{P}) = \inf_{g\in\mathcal{P}} \operatorname{dist}(f,g)\ .$$

For some of our applications, we will also use another notion of distance specific to real-valued functions, the L_1 distance. Testing under L_1 distance is a natural notion that has been studied ever since the first works in Property Testing (e. g., [27]).

For $f, g\colon \mathcal{X} \to [0,1]$, we write

$$L_1(f,g) = \frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} |f(x) - g(x)| = \mathbb{E}_{x\sim\mathcal{X}}[|f(x) - g(x)|] \in [0,1]$$

and extend the definition to $L_1(f,\mathcal{P})$, for $\mathcal{P} \subseteq \mathcal{X}^{[0,1]}$, as before.

Property testing. We recall the standard definition of testing algorithms, as well as some terminology:

Definition 2.1. Let $\mathcal{P}$ be a property of functions from $\mathcal{X}$ to $\mathcal{Y}$. A *q-query testing algorithm for $\mathcal{P}$* is a randomized algorithm $\mathcal{T}$ which takes as input $\varepsilon \in (0,1]$ as well as query access to a function $f\colon \mathcal{X} \to \mathcal{Y}$. After making at most $q(\varepsilon)$ queries to the function, $\mathcal{T}$ either outputs ACCEPT or REJECT, such that the following holds:

- if $f \in \mathcal{P}$, then $\mathcal{T}$ outputs ACCEPT with probability at least $2/3$; (Completeness)
- if $\operatorname{dist}(f,\mathcal{P}) \geq \varepsilon$, then $\mathcal{T}$ outputs REJECT with probability at least $2/3$; (Soundness)

where the probability is taken over the algorithm's randomness. If the algorithm only errs in the second case but accepts any function $f \in \mathcal{P}$ with probability 1, it is said to be a *one-sided* tester; otherwise, it is said to be *two-sided*. Moreover, if the queries made to the function can only depend on the internal randomness of the algorithm, but not on the values obtained during previous queries, it is said to be *non-adaptive*; otherwise, it is *adaptive*.

Additionally, we will also be interested in *tolerant* testers—roughly, algorithms robust to a relaxation of the first item above:

Definition 2.2. Let $\mathcal{P}$, $\mathcal{X}$, and $\mathcal{Y}$ be as above. A *q-query tolerant testing algorithm for $\mathcal{P}$* is a randomized algorithm $\mathcal{T}$ which takes as input $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, as well as query access to a function $f\colon \mathcal{X} \to \mathcal{Y}$. After making at most $q(\varepsilon_1,\varepsilon_2)$ calls to the oracle, $\mathcal{T}$ outputs either ACCEPT or REJECT, such that the following holds:

- if $\operatorname{dist}(f,\mathcal{P}) \leq \varepsilon_1$, then $\mathcal{T}$ outputs ACCEPT with probability at least $2/3$; (Completeness)
- if $\operatorname{dist}(f,\mathcal{P}) \geq \varepsilon_2$, then $\mathcal{T}$ outputs REJECT with probability at least $2/3$; (Soundness)

where the probability is taken over the algorithm's randomness. The notions of one-sidedness and adaptivity of Definition 2.1 extend to tolerant testing algorithms as well.

Note that as stated, in both cases the algorithm "knows" $\mathcal{X}, \mathcal{Y}$, and $\mathcal{P}$; so that the query complexity q can be parameterized by these quantities. More specifically, when considering $\mathcal{X} = [n]^d$ and the property $\mathcal{P}$ of k-monotonicity, we will allow q to depend on n, d, and k. Finally, we shall sometimes require a probability of success $1-\delta$ instead of the (arbitrary) constant $2/3$; by standard techniques, this can be obtained at the cost of a multiplicative $O(\log(1/\delta))$ in the query complexity.

PAC and agnostic learning [51]. A learning algorithm $\mathcal{A}$ for a *concept class* $\mathcal{C}$ of functions $f\colon \mathcal{X} \to \mathcal{Y}$ (under the uniform distribution) is given parameters $\varepsilon, \delta > 0$ and sample access to some target function $f \in \mathcal{C}$ *via* labeled samples $\langle x, f(x)\rangle$, where x is drawn uniformly at random from $\mathcal{X}$. The algorithm should output a hypothesis $h\colon \mathcal{X} \to \mathcal{Y}$ such that $\operatorname{dist}(h,f) \leq \varepsilon$ with probability at least $1-\delta$. The algorithm is *efficient* if it runs in time $\operatorname{poly}(n, 1/\varepsilon, 1/\delta)$. If $\mathcal{A}$ must output $h \in \mathcal{C}$ we say it is a *proper learning algorithm*, otherwise, we say it is an *improper learning* one.

Moreover, if $\mathcal{A}$ still succeeds when f does not actually belong to $\mathcal{C}$, we say it is an *agnostic learning algorithm*. Specifically, the hypothesis function h that it outputs must satisfy $\operatorname{dist}(f,h) \leq \textsc{opt}_f + \varepsilon$ with probability at least $1-\delta$, where $\textsc{opt}_f = \min_{g\in\mathcal{C}} \operatorname{dist}(f,h)$.

3 Structural results

In this section, we will prove that the distance to k-monotonicity of a Boolean function f can be expressed in a combinatorial way—which does not require measuring the distance between f and the closest k-monotone function to f. We will prove this for a general finite poset domain building up on the ideas of [26]. In the rest of this section, we denote by $\mathcal{P} = (V, \preceq)$ an arbitrary poset, the underlying domain of the function.

Definition 3.1. We define the *violated pattern K* as the sequence of alternating bits

$$K = (b_1, b_2, \cdots, b_k, b_{k+1})$$

of length $(k+1)$, where $b_1 = 1$ and all the bits in the sequence alternate, i. e., $b_i \neq b_{i+1}$ $\forall i \in [k]$.

A function $f\colon \mathcal{P} \to \{0,1\}$ is k-monotone only if it avoids K. That is, for any $x_1 \prec x_2 \prec \ldots \prec x_{k+1} \in \mathcal{P}$ we have $f(x_i) \neq K(i)$ for some $i \in [k+1]$.

Using insights from the literature on monotonicity testing, we show that functions far from k-monotonicity have a large matching of "violated hyperedges" in the "violation hypergraph" which we define shortly. Let us recall the definition of "violation graph" which has been extremely useful with monotonicity testing as seen in [23, 46, 32, 2, 26].

Definition 3.2 (Violation graph)**.** Given a function $f\colon \mathcal{P} \to \{0,1\}$, the *violation graph of f* is defined as $G_{\text{viol}}(f) = (\mathcal{P}, E(G_{\text{viol}}))$ where $(x,y) \in E(G_{\text{viol}})$ if $x, y \in \mathcal{P}$ form a monotonicity violating pair in f—that is $x \preceq y$ but $f(x) > f(y)$.

The following theorem, proved in [26], about violation graphs has been extremely useful in monotonicity testing literature.

Theorem 3.3. *Let $f\colon \mathcal{P} \to \{0,1\}$ be a function that is ε-far from monotone. Then, there exists a matching of edges in the violation graph for f of size at least $\varepsilon\,|\mathcal{P}|/2$.*

Now let us define a generalization of this concept, the violation hypergraph.

Definition 3.4 (Violation hypergraph)**.** Given a function $f\colon \mathcal{P} \to \{0,1\}$, the *violation hypergraph of f* is $H_{\text{viol}}(f) = (\mathcal{P}, E(H_{\text{viol}}))$ where $(x_1, x_2, \cdots, x_{k+1}) \in E(H_{\text{viol}})$ if the ordered $(k+1)$-tuple $x_1 \prec x_2 \prec \cdots \prec x_{k+1}$ forms a violation to k-monotonicity in f.

Note that $H_{rmviol}(f)$ is a $(k+1)$-uniform hypergraph. For a $(k+1)$-tuple $(x_1 \prec x_2 \prec \cdots \prec x_{k+1}) \in E(H_{\text{viol}})$, we will sometimes refer to the tuple as a $(k+1)$-uniform hyperedge. Now, we state the main theorem we prove in this section. This theorem offers an alternate characterization of distance to k-monotonicity that we seek. We recall that a set of edges forms a matching in a hypergraph if any pair of hyperedges is pairwise disjoint.

Theorem 3.5. *Let $f\colon \mathcal{P} \to \{0,1\}$ be a function that is ε-far from k-monotone. Then, there exists a matching of $(k+1)$-uniform hyperedges of size at least $\varepsilon\,|\mathcal{P}|/(k+1)$ in the violation hypergraph.*

To prove this theorem we first exploit the key notion of *extendability* (also explored in the context of testing monotonicity in [26]) which we define below. Later we will show that k-monotone functions are extendable.

Definition 3.6 (Extendability)**.** A property of Boolean functions is said to be *extendable* over a poset domain if the following holds for any set $X \subseteq \mathcal{P}$: given a function $f\colon X \to \{0,1\}$ which has the property (on X), it is possible to define a function $g\colon \mathcal{P} \to \{0,1\}$ such that $g(x) = f(x), \forall x \in X$ and g has the property.

In other words, a property is extendable if for any subset $X \subseteq \mathcal{P}$, given a function defined over the set X which respects the property, it is possible to fill in values outside X such that the new function obtained continues to respect the property. Next, we show that k-monotonicity is an extendable property:

Lemma 3.7. *k-monotonicity is an extendable property.*

Proof of Lemma 3.7. Consider $X \subseteq \mathcal{P}$ and some function $f\colon X \to \{0,1\}$ which is k-monotone over X. That is, for any $x_1 \prec x_2 \prec \cdots \prec x_{k+1} \in X$ there exists $i \in [k+1]$ such that $f(x_i) \neq K[i]$. Take a minimal point $v \in \mathcal{P} \setminus X$. That is, for any other point $v' \in \mathcal{P} \setminus X$ either $v \leq v'$ or v and v' are not comparable. We will use the following result:

Claim 3.8. *There exists a function $g\colon X \cup \{v\} \to \{0,1\}$ such that $g(x) = f(x)$ for all $x \in X$, and g respects k-monotonicity over its domain.*

Before proving this claim, we show how it implies Lemma 3.7. Namely, starting with any function $f\colon X \to \{0,1\}$ which is k-monotone on its domain X, we just keep applying the Claim 3.8 inductively until we get a function defined over the entire poset which respects k-monotonicity.

Proof of Claim 3.8. We will show this by contradiction. Suppose there is no good assignment available for $g(v)$, that is that both the choices $g(v) = 0$ and $g(v) = 1$ lead to a violation to k-monotonicity in g. Consider the choice $g(v) = 0$. Since this results in a violation to k-monotonicity, we know that there is a path $P_0 = (x_1 \prec x_2 \prec \cdots \prec x_{k+1})$ which is a violation to k-monotonicity. It is clear that $v \in P_0$; let i be such that $x_i = v$. Similarly, there is path $P_1 = (y_1 \prec y_2 \prec \cdots \prec y_{k+1})$ corresponding to $g(v) = 1$ which also contains the violated pattern, and some j such that $y_j = v$. And thus, $g(x_t) = g(y_t)$, for all $t \in [k+1]$ (as both of the paths indexed by x and y form a violation to k-monotonicity). By the above discussion 2 paths, P_0 and P_1, meet at v. We will see that one of the two paths

$$P_0' = (x_1 \prec x_2 \prec \cdots \prec x_{i-1} \prec y_i \prec y_{i+1} \prec \cdots \prec y_{k+1})$$

or

$$P_1' = (y_1 \prec y_2 \prec \cdots \prec y_{j-1} \prec x_j \prec x_{j+1} \prec \cdots \prec x_{k+1})$$

is already a violation to k-monotonicity in f. To see this, let us begin by recalling that we let v be the i^{th} vertex on P_0 and the j^{th} vertex on P_1. Now it is clear that $i \neq j$. Without loss of generality, suppose $i < j$. In this case, the evaluations of f along path P_1' form the violated pattern. This is because the function values alternate along the segment $(y_1 \prec y_2 \prec \cdots \prec y_{j-1})$. Also, the function values alternate along the segment $(x_i \prec x_{i+1} \prec x_{i+2} \prec \cdots \prec x_{k+1})$. And finally note that since $f(y_{j-1}) \neq f(y_j)$ and $f(y_j) = f(x_j)$ we get that $f(y_{j-1}) \neq f(x_j)$ as well. So, the path P_1' indeed contains a violation to k-monotonicity as claimed. The other case, $i > j$, is analogous. Hence the claim follows. $\square$

□

In the next lemma, we show that there is a nice characterization of distance to k-monotonicity in terms of the size of the *minimum vertex cover* of the violation hypergraph.[3]

Lemma 3.9. *Let $\mathcal{M}_k$ denote the set of k-monotone functions over the poset $\mathcal{P}$, and $f\colon \mathcal{P} \to \{0,1\}$. Then* $\mathrm{dist}(f,\mathcal{M}_k) = \varepsilon_f$ *if, and only if, the size of the minimum vertex cover in $H_{\mathrm{viol}}(f)$, denoted $|VC_{\min}(H_{\mathrm{viol}}(f))|$ is $\varepsilon_f\,|\mathcal{P}|$.*

Proof of Lemma 3.9. We establish separately the two inequalities.

Claim 3.10. $\mathrm{dist}(f,\mathcal{M}_k) \geq \dfrac{1}{|\mathcal{P}|}\,|VC_{\min}(H_{\mathrm{viol}})|$.

Proof. Suppose the distance to k-monotonicity is ε_f, and let g be a k-monotone function achieving it, so that $\mathrm{dist}(f,g) = \varepsilon_f$. Define $X = \{\, x \in \mathcal{P} : \ f(x) \neq g(x) \,\}$ (thus, $|X| = \varepsilon_f\,|\mathcal{P}|$). Let us consider the violation hypergraph for f given as $H_{\mathrm{viol}}(f) = (\mathcal{P}, E(H_{\mathrm{viol}}))$. Now, delete vertices in X and the hyperedges containing any vertex $v \in X$ from this hypergraph. Because X is the smallest set of vertices changing values at which gives a k-monotone function, it follows that every hyperedge in $E(H_{\mathrm{viol}})$ must contain a vertex in X. Thus, X indeed forms a vertex cover in $H_{\mathrm{viol}}(f)$. □

Claim 3.11. $\mathrm{dist}(f,\mathcal{M}_k) \leq \dfrac{1}{|\mathcal{P}|}\,|VC_{\min}(H_{\mathrm{viol}})|$.

Proof. Suppose the minimum vertex cover in the violation hypergraph has size $\varepsilon_f\,|\mathcal{P}|$. We will show that the distance of the function to k-monotonicity is ε_f. To see this, let $C \subseteq \mathcal{P}$ be the smallest vertex cover of the violation hypergraph of the said size. Observe that deleting C from $H_{\mathrm{viol}}(f)$ removes all the hyperedges, and therefore that the function f restricted to $X = \mathcal{P} \setminus C$ is k-monotone. And by the extendability of k-monotone functions established in Lemma 3.7, it follows that the function can be extended to the rest of the domain (by providing values in the cover C) such that it keeps respecting k-monotonicity. Thus, the distance to k-monotonicity is at most $|C|\,/\,|\mathcal{P}|$. □

□

Having characterized distance to k-monotonicity as the size of the smallest vertex cover in the violation graph, we are ready to establish Theorem 3.5. To do so, we will require the following standard fact:

Fact 3.12. *Let $G = (V,E)$ be a t-uniform hypergraph. Let M be the maximum matching in G. Then,* $|M| \leq VC_{\min}(G) \leq t\,|M|$

Proof of Theorem 3.5. By Lemma 3.9, we know that the violation hypergraph, $H_{\mathrm{viol}}(f)$ has a minimum vertex cover of size at least $\varepsilon_f\,|\mathcal{P}|$. And by Claim 3.12, it is seen that it contains a matching of t-uniform hyperedges of size at least $(\varepsilon_f/t)\,|\mathcal{P}|$. □

[3] Recall that a vertex cover in a hypergraph is just a set of vertices such that every hyperedge contains at least one of the vertices from this set.

4 On the Boolean hypercube

In this section, we focus on k-monotonicity of Boolean functions over the hypercube $\{0,1\}^d$. We begin in Section 4.1 with a tester with query complexity $2^{\widetilde{O}(\sqrt{d})}$, establishing a strict separation between learning and testing. Section 4.2 is then dedicated to our lower bounds on k-monotonicity testing.

4.1 Upper bound: beating the learning approach

In this section, we prove the following theorem:[4]

Theorem 1.2. *There exists a one-sided non-adaptive tester for k-monotonicity of functions* $f\colon \{0,1\}^d \to \{0,1\}$ *with query complexity*

$$q(d,\varepsilon,k) = \exp\left(O\left(\sqrt{d}\cdot\log d\cdot\log\frac{1}{\varepsilon}\right)\right).$$

Let us recall the following standard fact which we use in the proof.

Fact 4.1. *There exists an absolute constant $C > 0$ such that the number of points of $\{0,1\}^d$ that do not have integer weights in the* middle levels

$$\left[\frac{d}{2}-\sqrt{d}\log\frac{C}{\varepsilon}, \frac{d}{2}+\sqrt{d}\log\frac{C}{\varepsilon}\right]$$

is at most $\varepsilon 2^{d-1}$.

Proof of Theorem 1.2. Let us begin by describing the following tester. We will later see that this tester has the claimed query complexity.

(1) Sample $O(1/\varepsilon)$ random points from the middle levels.

(2) For each of the queries in the first step, query all points with Hamming weight in the middle levels which fall in the subcube below and in the subcube above each such random point. We call each of these $O(1/\varepsilon)$ collections of queries a *superquery*.

The key idea behind the tester is that there is a big "matching of violated hyperedges" in a function far from k-monotonicity, as seen in Theorem 3.5. The tester tries to find one of the "violated hyperedges." We recall that the violation hypergraph for a function f over the hypercube has $\{0,1\}^d$ as its vertex set. And a tuple $(x_1 < x_2 < \cdots < x_{k+1})$ of $(k+1)$ vertices forms an edge iff it is a violation to k-monotonicity. In the rest of the analysis, we assume that k is odd.[5] In this case, given a function f ε-far from k-monotonicity we can assume without loss of generality that f equals 0 on points with Hamming weight $< d/2-\sqrt{d}\log(C/\varepsilon)$, and equals 1 on points with Hamming weight $> d/2+\sqrt{d}\log(C/\varepsilon)$. From Fact 4.1 it follows that the resulting function is still $\varepsilon/2$-far from being k-monotone.

[4] We note that this result is only interesting in the regime $k \le \sqrt{d}$: indeed, for $k = \Omega(\sqrt{d}\log(1/\varepsilon))$ *every* function is ε-close to k-monotone.

[5] The case k even is similar. In this case, one may assume that f evaluates to 0 on all points outside the middle levels.

Now, by Theorem 3.5, we know that there exists a matching M_f of violations to k-monotonicity in f of size

$$\frac{\varepsilon/2 \cdot 2^d}{k+1} = \frac{\varepsilon 2^{d-1}}{k+1}\,.$$

The set of vertices that participate in these violations has cardinality

$$(k+1) \cdot \frac{\varepsilon 2^{d-1}}{k+1} = \varepsilon 2^{d-1}\,.$$

With constant probability, in $O(1/\varepsilon)$ queries in the first step, the tester queries a vertex that belongs to some violation from M_f. Then in the next step, the tester finds this violation. Now we bound the number of queries made in a single superquery. Since it involves only $2\sqrt{d}\log(C/\varepsilon)$ levels of the cube, the number of points queried in a single superquery is no more than $b = d^{O(\sqrt{d}\log(1/\varepsilon))}$. The total query complexity of the tester can therefore be upper bounded by $b/\varepsilon = d^{O(\sqrt{d}\log(1/\varepsilon))}$. We emphasize that the number of queries made by this tester has no dependence on k.

□

Remark 4.2. We can design and analyze another standalone tester with an almost identical asymptotic upper bound on query complexity. Our analysis of the performance of this tester *does not use* the extendability property. The tester makes $O(1/\varepsilon)$ iterations. In every iteration, it picks a *random seed* $x \in \{0,1\}^n$ and queries all points on chains going through x in the truncated cube. The idea behind our analysis is a notion of "rank" for points in the cube. We say $rank(x) \geq \ell$ if there is some ascending chain ending at x which has $\geq l$ flips on it. Observe that if the function is indeed far, then there is at least an ε fraction of points with rank at least $k+1$. Thus, with constant probability, some iteration hits a seed with large rank. And in the next step, the tester finds the violation. To upper bound the query complexity, note that the number of chains in the truncated cube going through any initial seed x is at most $b = d^{O(\sqrt{d}\log(1/\varepsilon))}$. This gives an upper bound of $O(b/\varepsilon)$.

4.2 Lower bounds

We now turn to lower bounds for testing k-monotonicity of Boolean functions over the hypercube $\{0,1\}^d$. In Section 4.2.1, we show that for $1 \leq k \leq d^{1/4}/2$, any one-sided non-adaptive tester for k-monotonicity requires $\Omega(d/k^2)^{k/4}$ queries, generalizing the $\Omega(\sqrt{d})$ lower bound for monotonicity due to Fischer *et al*. [26]. This bound suggests the problem becomes strictly harder when k increases: specifically, for $2 < k \leq d^{1/4}/2$ testing k-monotonicity requires $\omega(\sqrt{d})$ queries, while an $\widetilde{O}(\sqrt{d}/\varepsilon^2)$ one-sided non-adaptive upper bound holds for monotonicity testing [37].

We then describe in Section 4.2.2 a general reduction from monotonicity testing to k-monotonicity testing, for arbitrary constant k. This blackbox reduction allows us to carry any lower bound (possibly two-sided or adaptive) for monotonicity testing to k-monotonicity. In particular, combining it with the recent lower bounds [19, 7] for two-sided monotonicity testing, we obtain an $\Omega(d^{1/2-o(1)})$ lower bound for non-adaptive k-monotonicity testers, and an $\Omega(d^{1/4})$ lower bound for adaptive ones.

4.2.1 One-sided lower bounds

Theorem 1.1. *For $1 \le k \le d^{1/4}/2$, any one-sided non-adaptive tester for k-monotonicity of functions $f\colon \{0,1\}^d \to \{0,1\}$ must make $\Omega\big(d/k^2\big)^{k/4}$ queries.*

One natural way to prove Theorem 1.1 is to extend the monotonicity lower bound in [26]. To this end, we consider the $\binom{n}{k}$-sized family of functions $\{g_S\colon S \subseteq [d] \text{ of size } k\}$ where every single function in the family is a truncated anti-parity over some subset S of size k. By an argument analogous to the one used in [26], one can show $\Omega(d^{k/4})$ lower bound on query complexity. However, functions in this family are only $\Omega(2^{-k})$-far from being k-monotone. We refine this argument to obtain functions that are $\Omega(1)$-far from being k-monotone, for all k.

Proof of Theorem 1.1. Consider the family of functions $\{f_S\colon S \subseteq [d] \text{ of size } t\}$ where $t \ge k$ is a parameter to be determined later and f_S is a truncated anti-parity over t input coordinates indexed by S, namely,

$$f_S = \begin{cases} \overline{\oplus_{i\in S} x_i} & \text{if } ||x| - d/2| \le \sqrt{d}, \\ 0 & \text{otherwise.} \end{cases}$$

The theorem immediately follows from the two claims below.

Claim 4.3. *For any non-adaptive q-query algorithm $\mathcal{A}$, there exists f_S where $|S| = t$ such that $\mathcal{A}$ reveals a violation on f_S with probability at most*

$$q^2 \binom{2\sqrt{d}}{k}\binom{t}{k} \Big/ \binom{d}{k}.$$

Claim 4.4. *For $k \le t \le \sqrt{d}$, f_S is $\Omega\big(\sum_{i=0}^{\lfloor (t-k-1)/2 \rfloor} \binom{t}{i}/2^t\big)$-far from any k-monotone function.*

In particular, let $t = 4k^2$. By Claim 4.4, for $t \le \sqrt{d}$, f_S is $\Omega(1)$-far from any k-monotone function, and by Claim 4.3, to reject every f_S with $\Omega(1)$ probability, q needs to be at least

$$d^{k/4} \cdot \left(\frac{k}{2e^2 t}\right)^{k/2} = \Omega(d/k^2)^{k/4}.$$

Proof of Claim 4.3. Let Q be an arbitrary set of q queries. Without loss of generality, we assume every query z in Q has Hamming weight $|z| \in [d/2 - \sqrt{d}, d/2 + \sqrt{d}]$. We define $Q_{x,z} = \{\, y \in Q : \ x \preceq y \preceq z \,\}$ and $\mathrm{Rej}(Q) = \{\, f_S : \ Q \text{ contains a violation for } f_S \,\}$. Note that $\mathrm{Rej}(Q) = \bigcup_{(x,z)\colon x\preceq z \in Q} \mathrm{Rej}(Q_{x,z})$. Hence we can bound the size of $\mathrm{Rej}(Q)$ by

$$|\mathrm{Rej}(Q)| \le \sum_{(x,z)\in Q^2\colon x\preceq z} |\mathrm{Rej}(Q_{x,z})|. \tag{4.1}$$

Fix any $x \preceq z \in Q$. Fix any violation for f_S in $Q_{x,z}$. Note that there exists $S' \subseteq S$ with $|S'| = k$ such that $Q_{x,z}$ contains two points x' and z' with $x \preceq x' \preceq z' \preceq z$ and $x'_{S'} = 0^k$ and $z'_{S'} = 1^k$. Also, note that x and z differ in at most $2\sqrt{d}$ coordinates so that there are at most $\binom{2\sqrt{d}}{k}$ distinct S' on which $x_{S'}$ is 0^k and $z_{S'}$ is

1^k. Moreover, for each S' there are at most $\binom{d-k}{t-k}$ distinct S such that $S' \subseteq S$. Therefore we can bound the size of $\mathrm{Rej}(Q_{x,z})$ by

$$|\mathrm{Rej}(Q_{x,z})| \leq \binom{2\sqrt{d}}{k}\binom{d-k}{t-k}. \tag{4.2}$$

Combining (4.1) and (4.2),

$$|\mathrm{Rej}(Q)| \leq q^2\binom{2\sqrt{d}}{k}\binom{d-k}{t-k}.$$

It follows that for any non-adaptive algorithm making at most q queries,

$$\sum_{S\subseteq[d]:|S|=t} \Pr[\mathcal{A} \text{ reveals a violation for } f_S] \leq \mathbb{E}[|\mathrm{Rej}(Q)|] \leq q^2\binom{2\sqrt{d}}{k}\binom{d-k}{t-k}.$$

Hence there exists f_S such that

$$\Pr[\mathcal{A} \text{ reveals a violation for } f_S] \leq q^2\frac{\binom{2\sqrt{d}}{k}\binom{d-k}{t-k}}{\binom{d}{t}} = q^2\frac{\binom{2\sqrt{d}}{k}\binom{t}{k}}{\binom{d}{k}}. \qquad \square$$

Proof of Claim 4.4. Let f'_S be the closest k-monotone function to f_S. Let Z denote the set $\{z \in \{0,1\}^{d-t} : d/2-\sqrt{d} \leq |z| \leq d/2+\sqrt{d}-t\}$. For any $t \leq \sqrt{d}$,

$$\left[\frac{d-t}{2}-\frac{\sqrt{d-t}}{2}, \frac{d-t}{2}+\frac{\sqrt{d-t}}{2}\right] \quad \text{is contained in} \quad \left[\frac{d}{2}-\sqrt{d}, \frac{d}{2}+\sqrt{d}-t\right]$$

so that $|Z| = \Omega(2^{d-t})$.

For any assignment $z \in Z$ on coordinates indexed by $[d]\setminus S$, $f_S(\cdot,z)$ agrees with $\overline{\oplus_{i\in S}x_i}$ and $f'_S(\cdot,z)$ is k-monotone. To finish the proof, it suffices to show that an anti-parity over t inputs is $\sum_{i=0}^{\lfloor(t-k-1)/2\rfloor}\binom{t}{k}/2^t$-far from any k-monotone function over t inputs. Indeed, this will imply that, for every $z \in Z$, $f_S(\cdot,z)$ differs from $f'_S(\cdot,z)$ on $\sum_{i=0}^{\lfloor(t-k-1)/2\rfloor}\binom{t}{k}$ points, and thus finally that f_S differs from $f_{S'}$ on

$$|Z|\cdot\sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor}\binom{t}{k} = \Omega\left(2^{d-t}\cdot\sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor}\binom{t}{k}\right)$$

points.

Now we show it is the case that an anti-parity function over t inputs $h(x_1,\ldots,x_t) = \overline{\oplus_{i=1}^t x_i}$ is far from any k-monotone function g (over $x_1,\ldots,x_t$). We begin by noting that we can sample a random point from $\{0,1\}^t$ by first sampling a random chain $\mathcal{C} = (x^0 = 0^t, x^1, \ldots, x^t = 1^t)$ (from all possible chains from 0^t to 1^t) then outputting x^i with probability $\binom{t}{i}/2^t$. Thus the distance between h and g, namely $\Pr_x[h(x) \neq g(x)]$, is the same as $\Pr_{\mathcal{C},i}[h(x^i) \neq g(x^i)]$ which can be further expanded as follows

$$\mathbb{E}_{\mathcal{C}}\left[\sum_{i=0}^{t}\frac{\binom{t}{i}}{2^t}\cdot\mathbb{1}_{\{h(x^i)\neq g(x^i)\}}\right] = \mathbb{E}_{\mathcal{C}}\left[\frac{1}{2^t}\sum_{i=1}^{t}\binom{t-1}{i-1}\cdot\left(\mathbb{1}_{\{h(x^i)\neq g(x^i)\}}+\mathbb{1}_{\{h(x^{i-1})\neq g(x^{i-1})\}}\right)\right] \tag{4.3}$$

where the last inequality relies on the identity

$$\binom{t}{i} = \binom{t-1}{i-1} + \binom{t-1}{i}.$$

For any fixed chain $\mathcal{C}$, because g alternates at most k times, there are at least $t-k$ choices of $i \in [t]$ such that $g(x^{i-1}) = g(x^i)$, which implies $\mathbb{1}_{\{h(x^i)\neq g(x^i)\}} + \mathbb{1}_{\{h(x^{i-1})\neq g(x^{i-1})\}} \geq 1$ (due to $h(x^{i-1}) \neq h(x^i)$). Thus

$$\sum_{i=1}^{t} \binom{t-1}{i-1} \cdot \left(\mathbb{1}_{\{h(x^i)\neq g(x^i)\}} + \mathbb{1}_{\{h(x^{i-1})\neq g(x^{i-1})\}}\right)$$

is at least the sum of smallest $t-k$ binomials among $\binom{t-1}{0}, \dots, \binom{t-1}{t-1}$ which is

$$\sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor} \binom{t-1}{i} + \sum_{i=0}^{\lfloor\frac{t-k-2}{2}\rfloor} \binom{t-1}{t-1-i} = \sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor} \binom{t-1}{i} + \sum_{i=0}^{\lfloor\frac{t-k-2}{2}\rfloor} \binom{t-1}{i} \geq \sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor} \binom{t}{i},$$

which implies

$$\Pr_x[h(x) \neq g(x)] \geq \frac{1}{2^t} \sum_{i=0}^{\lfloor\frac{t-k-1}{2}\rfloor} \binom{t}{i}$$

by combining the above with (4.3). □

□

Remark 4.5. One may observe that our lower bound is suboptimal (by a quadratic factor) for the case $k = 1$, i. e., for monotonicity. In this case, the construction and argument of [26] indeed give an $\Omega(d^{1/2})$ lower bound, while instantiating Theorem 1.1 only yields $\Omega(d^{1/4})$. The reason is that, in their proof, Fischer et al. apply a "spanning tree argument" on the set of violating edges to obtain this quadratic improvement: however, it is not clear how this argument, particularly well-suited for their construction (basically, for antidictator functions), would apply to our functions.

4.2.2 Two-sided lower bounds

The following theorem gives a construction that enables us to convert monotone functions into k-monotone functions, and functions that are far from monotone into functions that are far from k-monotone.

Theorem 4.6. *There exists an efficiently computable function $h\colon \{0,1\}^{d/2} \to \{0,1\}$ such that, for any $g\colon \{0,1\}^{d/2} \to \{0,1\}$, $g||h\colon \{0,1\}^d \to \{0,1\}$ is a Boolean function (defined as $g||h(x,y) \overset{\text{def}}{=} g(x) \oplus h(y)$ for any $x,y \in \{0,1\}^{d/2}$) satisfying the following.*

- *if g is monotone, then $g||h$ is a k-monotone function;*
- *if g is ε-far from monotone, then $g||h$ is $\Omega(\varepsilon/k)$-far from being a k-monotone function.*

The above theorem reduces testing monotonicity to testing k-monotonicity (for arbitrary constant k) with the same number of queries to the input function. As we will see, it carries any lower bound for testing monotonicity over to testing k-monotonicity, while preserving the characteristics (two-sidedness, adaptivity) of the original lower bound. In particular, combining it with the recent of [19, 21], we obtain the following corollary.

Corollary 4.7. *For any $c > 0$ and $k \geq 1$, there exists $\varepsilon = \varepsilon(k,c) > 0$ such that any two-sided non-adaptive algorithm for testing whether f is k-monotone or ε-far from it requires $\Omega(d^{1/2-c})$ queries. Any two-sided adaptive algorithm requires $\widetilde{\Omega}(d^{1/3})$ queries.*

To prove Theorem 4.6, we prove the following three claims. Claim 4.8 and Claim 4.9 assume the existence of a $(k-1)$-monotone function h which satisfies an interesting structural property. This property is then exploited to establish Theorem 4.6. Finally in Claim 4.10, we establish the existence of such h, and Theorem 4.6 follows.

Claim 4.8. *Let $h\colon \{0,1\}^d \to \{0,1\}$ be a $(k-1)$-monotone function. Then for any monotone $g\colon \{0,1\}^d \to \{0,1\}$, $f = g||h$ is a k-monotone function.*

Claim 4.9. *Suppose there exists a $h\colon \{0,1\}^d \to \{0,1\}$ such that the following holds. There exists at least M paths of length $k-1$ such that (i) all paths are vertex disjoint and (ii) for every path $y_1 \preceq \cdots \preceq y_k$, $h(y_1) = 0$ and $h(y_i) \neq h(y_{i+1})$ for $1 \leq i \leq k-1$. Then, for any $g\colon \{0,1\}^d \to \{0,1\}$ which is ε-far from being a monotone function, the function $f = g||h$ is $\Omega((M/2^d)\cdot\varepsilon)$-far from being a k-monotone function.*

Claim 4.10. *For any constant k, there exists an efficiently computable $h\colon \{0,1\}^d \to \{0,1\}$ such that h is a $(k-1)$-monotone function and h contains at least $(1-o_d(1))2^d/k$ paths of length $k-1$ such that all paths are vertex disjoint and for every path $y_1 \preceq \cdots \preceq y_k$, $h(y_1) = 0$ and $h(y_i) \neq h(y_{i+1})$ for $1 \leq i \leq k-1$.*

Proof of Claim 4.8. Suppose $f = g||h$ is not k-monotone, then there exist $(x_1,y_1),\ldots,(x_{k+1},y_{k+1})$ such that $(x_1,y_1) \preceq \cdots \preceq (x_{k+1},y_{k+1})$, $f(x_1,y_1) = 1$ and $f(x_i,y_i) \neq f(x_{i+1},y_{i+1})$ for any $1 \leq i \leq k$. Because g is monotone, either g is constant on $x_1,\ldots,x_{k+1}$, or there exists an index $1 < j \leq k+1$ such that $g(x_i) = 0$ for $i < j$ and $g(x_i) = 1$ for $i \geq j$. In the first case, if g is the constant 0 on $x_1,\ldots x_{k+1}$ (*resp. constant* 1) either $h(y_i) = f(x_i,y_i)$ (*resp. or* $h(y_i) = 1 - f(x_i,y_i)$) for any i and h alters exactly k times on points $y_1 \preceq \cdots \preceq y_{k+1}$. For the second case, $h(y_1) = f(x_1,y_1) = 1$, and h alters exactly $k-1$ times on $y_1 \preceq \cdots \preceq y_{j-1} \preceq y_{j+1} \preceq \cdots \preceq y_{k+1}$. Both cases contradict h being $(k-1)$-monotone. ☐

Proof of Claim 4.9. Let M_h be the maximal set of paths of length k such that all paths are vertex disjoint and for every path $y_1 \preceq \cdots \preceq y_k$, $h(y_1) = 0$ and $h(y_i) \neq h(y_{i+1})$ for $1 \leq i \leq k-1$. Let M_g be the maximal set of pairs such that all pairs are vertex disjoint and every pair $x_1 \preceq x_2$ is a violation for g, i. e., $g(x_1) = 1$ and $g(x_2) = 0$. For each path $(y_1 \preceq \cdots \preceq y_k) \in M_h$ and any pair $(x_1 \preceq x_2) \in M_g$, it is easy to see the following path is a violation for $f||g$ being k-monotone :

$$(x_1,y_1),\ldots,(x_1,y_k),(x_2,y_k)\,.$$

Let f' be the closest k-monotone function to f. For each violating path, f and f' differ on at least 1 point. Because both M_h and M_g are vertex disjoint, violating paths constructed by taking every path in M_h and every pair in M_g are vertex disjoint. Thus f and f' differ on at least $|M_h| \times |M_g|$ points and f is $(|M_h|\cdot|M_g|/2^{2d})$-far from f'. It is known ([28]) that for any $g\colon \{0,1\}^d \to \{0,1\}$ which is ε-far from monotone, $|M_g| \geq 2^{d-1}\varepsilon$. The desired conclusion follows. ☐

Proof of Claim 4.10. Let $B_1, B_2, \ldots, B_k$ be consecutive blocks each consisting of consecutive Hamming levels of the hypercube. Letting $|B_i|$ denote the number of points in the i^{th} block, note that it is possible to have all blocks satisfy,

$$\forall i \in [k] \left(1 - \frac{k}{\sqrt{d}}\right)\frac{2^d}{k} \leq |B_i| \leq \left(1 + \frac{k}{\sqrt{d}}\right)\frac{2^d}{k}\,.$$

Because k is a constant and every layer contains at most $2^d/\sqrt{d}$ points, we can always greedily find $B_1, \ldots, B_k$ one by one. Let h be a function such that h is constant over every block and alternates on successive blocks, and which evaluates to 0 over all of B_1. Note that h is $(k-1)$-monotone. Next we inductively argue that for any $1 \leq j \leq k$, $B_1, \ldots, B_j$ contain at least $(1 - o_d(1))2^d/k$ vertex disjoint paths of length $j-1$ such the ith point on every path is in B_i. Claim 4.10 follows from the case $j = k$.

For $j = 1$, the statement holds by taking all points in B_1. For $j > 1$, assume that $B_1, \ldots, B_{j-1}$ contain a set P_{j-1} of such $(1 - o_d(1))2^d/k$ vertex disjoint paths of length $j-2$. Let M be the maximal matching between B_{j-1} and B_j and let P_j be the set of paths of length $j-1$ constructed in following way: for each path in P_{j-1} with an endpoint y_{j-1}, if there exists y_j such that $(y_{j-1}, y_j) \in M$, we add the path appended with y_j into P_j. Because no points in B_j will be added into two different paths in P_j and P_{j-1} are vertex disjoint, paths in P_j are vertex disjoint.

Now we show $|M| = \min(|B_{j-1}|, |B_j|)$ which implies $|P_j| \geq (1 - o_d(1))2^d/k$. Suppose $|B_{j-1}| \leq |B_j|$ (the argument is analogous in the other case). For any subset S of B_{j-1}, let f_S be the indicator function of the upper closure of S denoted as $N(S)$. It is not hard to check that f_S is monotone and thus

$$\Pr[f_S(x) = 1 \mid x \in B_j] \geq \Pr[f_S(x) = 1 \mid x \in B_{j-1}]\,.$$

It follows

$$\left|N(S) \cap B_j\right| = \left|B_j\right| \Pr[f_S(x) = 1 \mid x \in B_j] \geq \left|B_{j-1}\right| \Pr[f_S(x) = 1 \mid x \in B_{j-1}] \geq |S|\,.$$

By Hall's theorem, $|M| = |B_{j-1}|$. By similar argument, we can show $|M| = |B_j|$ when $|B_{j-1}| > |B_j|$. Thus $|M| = \min(|B_{j-1}|, |B_j|)$. □

5 On the line

In this section we prove our results on testing k-monotonicity on the line, that is of functions $f\colon [n] \to \{0,1\}$. We start with Theorem 1.4, which establishes that this can be done non-adaptively with one-sided error, with only $O(k/\varepsilon)$ queries; we then turn to Theorem 1.3, which shows that this is the best one can hope for if we insist on one-sidedness. The last result of this section is Theorem 1.5, where we show that—perhaps unexpectedly—*two-sided* algorithms, even non-adaptive, can break this barrier and test k-mononicity with *no* dependence on k.

5.1 Upper and lower bounds for one-sided testers

We first prove the upper bound, restated below:

Theorem 1.4. *There exists a one-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n, \varepsilon, k) = O(k/\varepsilon)$.*

Proof. We assume that $\varepsilon n/(50k)$ is an integer,[6] and partition the domain into $K \overset{\text{def}}{=} 50k/\varepsilon$ intervals of size $\varepsilon n/(50k)$, the consecutive "blocks" $B_1,\dots,B_K$. We then define $g\colon [n] \to \{0,1,*\}$ as the function constant on each block $B_i = \{b_i,\dots,b_{i+1}-1\}$, such that

- If $f(b_i) = f(b_{i+1}-1)$, then $g(j) = f(b_i)$ for all $j \in B_i$;
- otherwise, $g(j) = *$ for all $j \in B_i$.

We say that a block B_i such that $g|_{B_i} = *$ is a *changepoint block* for g. Clearly, given query access to f one can obtain the value of g on any point $j \in [n]$ with only two queries to f. Moreover, defining $\widetilde{g}\colon [n] \to \{0,1\}$ to be the function obtained from g by replacing $*$ by 0, we observe the following:

- If f is k-monotone, then (i) so is $\widetilde{g}$, and (ii) f and $\widetilde{g}$ differ in at most k blocks (note that any block—whether changepoint block or otherwise—whenever they differ flip count of f goes up by 1), so that $\operatorname{dist}(f,g) \le k\cdot(1/K) = \varepsilon/50$;
- If f is ε-far from k-monotone, then either (i) $\widetilde{g}$ is not k-monotone, or (ii) $\operatorname{dist}(f,\widetilde{g}) > \varepsilon$.

We first discuss a natural idea, why it fails, and how to fix it. The above suggests a very simple algorithm: first, learn the function $\widetilde{g}$ (exactly) using $O(k/\varepsilon)$ queries, and then distinguish between $\operatorname{dist}(f,\widetilde{g}) \le \varepsilon/50$ and $\operatorname{dist}(f,\widetilde{g}) > \varepsilon$ by random sampling, using $O(1/\varepsilon)$ queries. This immediately results in a non-adaptive testing algorithm with query complexity $O(k/\varepsilon)$; however, this tester is inherently two-sided, due to the second (distance estimation) step. To obtain one-sidedness, we thus take a slightly different route, and consider the number of blocks on which f and $\widetilde{g}$ differ instead of estimating the distance. As finding at least $k+1$ such blocks can only happen when f is not k-monotone, this will never result in rejecting a k-monotone function; but this one-sidedness comes at the price of a slightly less straightforward analysis.

As mentioned above, we start by learning g (and thus $\widetilde{g}$) exactly, using $2K = O(k/\varepsilon)$ non-adaptive queries. Setting $m \overset{\text{def}}{=} C\cdot(k/\varepsilon)$, we also sample $m' \sim \text{Poisson}(m)$ points[7] $j_1,\dots,j_{m'}$ independently and uniformly from $[n]$, where $C > 0$ is a constant to be determined in the course of the analysis, and query the value of f (and g) on all of them. Then, we reject if either (i) $\widetilde{g}$ is not k-monotone; or (ii) there exist at least $k+1$ distinct blocks which contain a sample s_j such that $\widetilde{g}(s_j) \ne f(s_j)$.

By definition, this tester is non-adaptive; and it is not difficult to see it accepts any k-monotone function with probability 1, since in that case f and g (and a fortiori $\widetilde{g}$) differ in at most k blocks.

It remains to argue soundness: we will show that if f is ε-far from k-monotone, the tester will reject with probability at least $2/3$. By the first check made, (i), we can assume in the following that $\widetilde{g}$ is k-monotone—as otherwise f is rejected with probability 1—and we need to show that (ii) will reject with

[6]If not, we consider instead

$$\varepsilon' \overset{\text{def}}{=} \frac{50k}{n}\left\lfloor\frac{\varepsilon n}{50k}\right\rfloor > \varepsilon - \frac{50k}{n} > \frac{\varepsilon}{2}$$

if $\varepsilon > 100k/n$; while if $\varepsilon \le 100k/n$ we query the entire function, for a total of $n = O(k/\varepsilon)$ queries.

[7]The fact that we sample $\text{Poisson}(m)$ instead of m is for ease of the analysis; note that due to the tight concentration of Poisson random variables, with probability $1-o(1)$ we will have $m' \le 2m$. If this does not happen, the tester can output ACCEPT, incurring only a small additional error probability (and not affecting the one-sidedness).

probability at least $2/3$. For each block B_i (where $i \in [K]$), let $p_i \in [0,1/K]$ be defined as the (normalized) number of points in B_i on which f and $\widetilde{g}$ differ (we henceforth refer to such a point as a *giveaway point*):

$$p_i \stackrel{\text{def}}{=} \frac{1}{n} \sum_{j \in B_i} \mathbb{1}_{\{f(j) \neq \widetilde{g}(j)\}} \,.$$

Since f is ε-far from the k-monotone function $\widetilde{g}$, we have $\sum_{i=1}^{K} p_i \geq \varepsilon$. Now, letting Z_i be the indicator of the event that among the m' samples, at least one is a giveaway point from B_i, and $Z = \sum_{i=1}^{K} Z_i$, we can write $Z_i = \mathbb{1}_{\{Y_i \neq 0\}}$, where the $(Y_i)_{i \in [K]}$ are *independent* Poisson random variables with $Y_i \sim \text{Poisson}(mp_i)$. The expected number of blocks in which a giveaway point is sampled is then

$$\mathbb{E}Z = \sum_{i=1}^{K} \mathbb{E}Z_i = \sum_{i=1}^{K} \Pr[Y_i \neq 0] = \sum_{i=1}^{K} (1 - \mathrm{e}^{-mp_i})\,.$$

Since for every $i \in [K]$ it holds that $\mathrm{e}^{-mp_i} \leq 1 - (m/2)p_i$ (the inequality holding since $0 \leq mp_i \leq 1$, which is verified for $m \leq K$), we get

$$\mathbb{E}Z = \sum_{i=1}^{K} \mathbb{E}Z_i \geq \sum_{i=1}^{K} \frac{m}{2} p_i \geq \frac{m\varepsilon}{2} \geq \frac{C}{2} k\,.$$

Moreover, by a Chernoff bound, we get that

$$\Pr\left[Z < \frac{C}{4} k \right] \leq \mathrm{e}^{-\frac{Ck}{16}} \leq \mathrm{e}^{-\frac{C}{16}}$$

which is less that $1/4$ for $C \geq 23$. Setting $C \stackrel{\text{def}}{=} 30$ satisfies both conditions that $m \leq K$ and $C \geq 23$, and results in a one-sided non-adaptive tester which rejects functions far from k-monotone with probability at least $1 - 1/4 + o(1) \geq 2/3$. □

Turning to the lower bounds against one-sided testers, we show the following:

Theorem 1.3. *Any one-sided (possibly adaptive) tester for k-monotonicity of functions* $f\colon [n] \to \{0,1\}$ *must have query complexity* $\Omega(k/\varepsilon)$.

Proof. Since a lower bound of $\Omega(1/\varepsilon)$ straightforwardly holds, we can restrict ourselves to $k \geq 8$, and $\varepsilon < 1/12$; moreover, we assume without loss of generality that $\varepsilon n/k$ is an integer, and partition the domain into $K \stackrel{\text{def}}{=} k/\varepsilon$ intervals of size $\varepsilon n/k$, the consecutive "blocks" $B_1, \dots, B_K$. For $v \in \{0,1\}^{K/2}$, we define $g_v\colon [n] \to \{0,1\}$ as the function which has constant value v_i on block B_{2i-1} and has constant value 1 on the remaining blocks.

Consider the distribution over $\{g_v\}_v$ where each coordinate of v is independently set to 0 with probability $p \stackrel{\text{def}}{=} 6\varepsilon$, and 1 otherwise. We next show that g_v is at least ε-far from any k-monotone function with very high probability over the choice of v. By a Chernoff bound, with probability at least $1 - \mathrm{e}^{-pK/16} = 1 - \mathrm{e}^{-3k/8}$, g_v has at least $pK/4 = 3k/2$ blocks that are 0 blocks. Conditioned on this, it is easy to see that g_v is ε-far from k-monotone: indeed, to make it k-monotone one has to flip its value on at least k blocks, and each block contains an ε/k fraction of the domain.

Fix any deterministic adaptive algorithm with query complexity $q \le k/(24\varepsilon)$ queries, and denote by $x_1,\dots,x_q$ the sequence of queries made (when given query access to some function g_v). Without loss of generality, we assume that $x_1,\dots,x_q$ are made into distinct odd blocks. $x_1,\dots,x_q$ reveals a violation if and only if number of 0's in corresponding answers is at least $k/2+1$.

If $x_1,\dots,x_q$ are non-adaptive queries, the probability that the number of 0's in $f(x_1),\dots,f(x_q)$ is at least $k/2+1$ can be upper bounded by Chernoff Bound in a straightforward way. Because each odd block is sampled independently, the same bound can be obtained in the adaptive setting. Note, for arbitrary $a \in \{0,1\}^q$,

$$\Pr[f(x_1)=a_1,\dots,f(x_q)=a_q] = \Pr[f(x_1)=a_1]\cdot\prod_{i=2}^{q}\Pr[f(x_i)=a_i \mid f(x_1)=a_1,\dots,f(x_{i-1})=a_{i-1}].$$

Because x_i is determined by $f(x_1)=a_1,\dots,f(x_{i-1})=a_{i-1}$ and each odd block is sampled independently, for every i it holds that $\Pr[f(x_i)=a_i \mid f(x_1)=a_1,\dots,f(x_{i-1})=a_{i-1}] = 6\varepsilon$ if $a_i=0$ and $\Pr[f(x_i)=a_i \mid f(x_1)=a_1,\dots,f(x_{i-1})=a_{i-1}] = 1-6\varepsilon$ if $a_i=1$. Thus.

$$\Pr[f(x_1)=a_1,\dots,f(x_q)=a_q] = (1-6\varepsilon)^{|a|}(6\varepsilon)^{q-|a|}. \tag{5.1}$$

Let X_i be the indicator that $f(x_i)=0$. We get that, writing $F(i,N,p)$ for the cumulative distribution function of a Binomial with parameters N and p,

$$\begin{aligned}\Pr\left[\sum_{i=1}^{q} X_i \ge \frac{k}{2}+1\right] &\le \sum_{a\in\{0,1\}^q:\ |\bar{a}|\ge k/2} (1-6\varepsilon)^{|a|}(6\varepsilon)^{r-|a|}\\ &= \sum_{\ell=0}^{q-k/2}\binom{r}{\ell}(1-6\varepsilon)^{\ell}(6\varepsilon)^{q-\ell} = F\left(q-\frac{k}{2},q,1-6\varepsilon\right)\\ &= F(q(1-x),q,1-6\varepsilon) && (x \overset{\text{def}}{=} \tfrac{k}{2q}\in(12\varepsilon,1))\\ &\le \mathrm{e}^{-qD(1-x\|1-6\varepsilon)} && (\text{relative entropy Chernoff bound}^{8})\end{aligned}$$

where

$$D(p\,||\,p') \overset{\text{def}}{=} p\ln\frac{p}{p'} + (1-p)\ln\frac{1-p}{1-p'},$$

and we used the fact that $k/2+1 \le q \le k/(24\varepsilon)$. Rewriting slightly the right-hand-side, we obtain

$$\Pr\left[\sum_{i=1}^{q} X_i \ge \frac{k}{2}+1\right] \le \mathrm{e}^{-\frac{k}{2}\Phi(x)}$$

for

$$\Phi(x) \overset{\text{def}}{=} \frac{1}{x}\left((1-x)\ln\frac{1-x}{1-6\varepsilon} + x\ln\frac{x}{6\varepsilon}\right).$$

[8]Recall that the relative entropy version of the Chernoff bound states that $F(m,N,p) \le \mathrm{e}^{-mD(\frac{m}{N}||p)}$ as long as $0 \le \frac{m}{N} \le p$.

It is not hard to see that Φ is increasing on $[6\varepsilon, 1)$ (indeed, for $x \in [6\varepsilon, 1)$, we have

$$\Phi'(x) = -\frac{1}{x^2} \ln \frac{1-x}{1-6\varepsilon}$$

and hence $\Phi'(x) \geq 0$), and since $x \geq 12\varepsilon$ the right-hand-side is at most $\mathrm{e}^{-(k/2)\Phi(12\varepsilon)}$. It then suffices to observe that, for $\varepsilon \leq 1/12$, it holds that $\Phi(12\varepsilon) \geq \Phi(0) = \ln 2 - 1/2 > 1/8$ to conclude that

$$\Pr\left[\sum_{i=1}^{q} X_i \geq \frac{k}{2} + 1\right] \leq \mathrm{e}^{-\frac{k}{16}}$$

and therefore obtain

$$\Pr[f(x_1), \ldots, f(x_q) \text{ contains at least } (k/2+1) \text{ zeros}] \leq \mathrm{e}^{-\frac{k}{16}}.$$

Combining the two, this shows that the probability that $x_1, \ldots, x_q$ does not reveal a violation for g_v while g_v is ε-far from k-monotone is at least $1 - \mathrm{e}^{-k/16} - \mathrm{e}^{-3k/8} > 1/3$ (since $k \geq 8$). By Yao's principle, for any (possibly randomized) non-adaptive algorithm $\mathcal{A}$ making at most $k/(24\varepsilon)$ there exists a fixed v such that g_v is ε-far from k-monotone yet $\mathcal{A}$ rejects g_v with probability less than $2/3$. The desired conclusion follows. □

5.2 Upper bound for two-sided testers: proof of Theorem 1.5

In this section, we prove the two-sided non-adaptive upper bound of Theorem 1.5, restated below:

Theorem 1.5. *There exists a two-sided non-adaptive tester for k-monotonicity of functions $f\colon [n] \to \{0,1\}$ with query complexity $q(n,\varepsilon,k) = \widetilde{O}(1/\varepsilon^7)$, independent of k.*

In what follows, we assume that $k > 20/\varepsilon$, as otherwise we can use for instance the $O(k/\varepsilon)$-query (non-adaptive, one-sided) tester of Theorem 1.4 to obtain an $O(1/\varepsilon^2)$ query complexity.

5.2.1 Testing k-monotonicity over $[Ck]$

In this section, we give a poly(C/ε)-query tester for k-monotonicity over the domain $[Ck]$, where C is a parameter to be chosen (for our applications, we will eventually set $C = \text{poly}(1/\varepsilon)$).

Lemma 5.1. *There exists a two-sided non-adaptive tester for k-monotonicity of functions $f\colon [Ck] \to \{0,1\}$ with query complexity $O(C^3/\varepsilon^3)$.*

The tester proceeds by reducing to support size estimation and using (a slight variant of) an algorithm of Canonne and Rubinfeld [15]. Let $f\colon [Ck] \to \{0,1\}$, and suppose f is s-monotone but not $(s-1)$-monotone. Then there is a unique partition of $[Ck]$ into $s+1$ disjoint intervals $I_1, I_2, \ldots, I_{s+1}$ such that f is constant on each interval; note that this constant value alternates in consecutive intervals. We can then define a distribution D_f over $[s+1]$ such that $D_f(i) = |I_i|/(Ck)$.

Our next claims, Claim 5.2 and Observation 5.3, provide the basis for the reduction (from testing k-monotonicity of f to support size estimation of D_f).

Claim 5.2. *If f is ε-far from k-monotone, then it is not $(1+\varepsilon)k$-monotone, and in particular $\left|\operatorname{supp}(D_f)\right| > (1+\varepsilon)k+1$.*

Also, note that if f is k-monotone, then $\left|\operatorname{supp}(D_f)\right| \leq k+1$. This together with Claim 5.2 furnishes the following (for $k > 20/\varepsilon$):

Observation 5.3. In order to ε-test k-monotonicity of f, it suffices to estimate $\left|\operatorname{supp}(D_f)\right|$ to within $\varepsilon k/10$.

The proofs of the above results are relatively straightforward. So we defer these proofs to the end of this section and now proceed to use algorithm of [15] to do support size estimation. The algorithm of [15] uses "dual access" to D; an oracle that provides a random sample from D, and an oracle that given an element of D, returns the probability mass assigned to this element by D.

Theorem 5.4 ([15, Theorem 14 (rephrased)])**.** *In the dual access model described above, there exists an algorithm that, on input a threshold $n \in \mathbb{N}^*$ and a parameter $\varepsilon > 0$, and given access to a distribution D (over an arbitrary set) satisfying*

$$\min_{x\in\operatorname{supp}(D)} D(x) \geq \frac{1}{n}$$

estimates the support size $|\operatorname{supp}(D)|$ up to an additive εn, with query complexity $O(1/\varepsilon^2)$.

Note however that we only have access to D_f through query access to f, and thus have to manage to simulate (efficiently) access to the former. One difficulty is that, to access $D_f(i)$, we need to determine where I_i lies in f. For example, finding $D_f(k/2)$ requires finding $I_{k/2}$, which might require a large number of queries to f. We circumvent this by weakening the "dual access" model in two ways, arguing for each of these two relaxations that the algorithm of [15] can still be applied:

- we rewrite the support size as in [15], as $|\operatorname{supp}(D_f)| = \mathbb{E}_{x\sim D_f}[1/D_f(x)]$. We want to estimate it to within $\pm O(\varepsilon k)$ which we can do by random sampling;
- the quantity inside the expectation depends on $D_f(x)$ but not x itself, so "labels" are unnecessary for our random sampling. Thus, it will be sufficient to be able to compute $D_f(x)$ (and thus $1/D_f(x)$) for a random $x \sim D_f$, even if not actually knowing x itself;
- actually, even calculating $D_f(x)$ may possibly too expensive, so instead we will estimate

$$\mathbb{E}_{x\sim D_f}[1/\widetilde{D}_f(x)] \quad \text{where} \quad \widetilde{D}_f(x) = \min\left(\frac{20}{\varepsilon k}, D_f(x)\right).$$

  Note that $\widetilde{D}_f$ might no longer define a probability distribution; but this expectation is only off by at most $\varepsilon k/20$, since $1/D'_f(x) = \max(\varepsilon k/20, 1/D_f(x))$ and $1/D_f(x)$ is positive.

More details follow.

First, we note as discussed above that the algorithm does not require knowing the "label" of any element in the support of the distribution: the only access required is being able to randomly sample elements according to D_f, and evaluate the probability mass on the sampled points. This, in turn, can be done, as the following two lemmata explain:

Lemma 5.5 (Sampling from D_f)**.** *Let $i \in [n]$ be chosen uniformly at random, and let j be such that $i \in I_j$. Then, the distribution of j is exactly D_f.*

Lemma 5.6 (Evaluating $D_f(j)$)**.** *Suppose $I_j = \{a, a+1, \dots, b\}$. Given i such that $i \in I_j$, we can find I_j by querying $f(i+1) = f(i+2) = \cdots = f(b)$ and $f(b+1) \neq f(b)$, as well as $f(i-1) = f(i-2) = \cdots = f(a)$ and $f(a-1) \neq f(a)$. The number of queries to f is $b-a+3 = |I_j| + 3$.*

Here comes the second difficulty: if we straightforwardly use these approaches to emulate the required oracles to estimate the support size of D_f, the number of queries is potentially very large. For instance, if we attempt to query $D_f(j)$ where $|I_j| = \Omega(k)$, we will need $\Omega(k)$ queries to f. This is where comes the second relaxation: specifically, we shall argue that it will be enough for us to "cap" the size of the interval (as per our next lemma).

Observation 5.7 (Evaluating $D_f(j)$ with a cap)**.** Given i such that $i \in I_j$, we will query f on every point in $[i - 20C/\varepsilon, i + 20C/\varepsilon]$. If $|I_j| \leq 20C/\varepsilon$, then I_j will be determined by these queries. If these queries do not determine I_j, we know $|I_j| > 20C/\varepsilon$. Beyond querying i, this requires $40C/\varepsilon$ (nonadaptive) queries.

We now can put all the above pieces together and give the proof for Lemma 5.1:

Proof of Lemma 5.1. As previously discussed, we use the algorithm of [15] for estimating support size. Inspecting their algorithm, we see that our cap of $20C/\varepsilon$ for interval length (and therefore $20/(\varepsilon k)$ for maximum probability reported) might result in further error of the estimate. The algorithm interacts with the unknown function by estimating the expected value of $1/D_f(j)$ over random choices of j with respect to D_f. Our cap can only decrease this expectation by at most $(\varepsilon k)/20$. Indeed, the algorithm works by estimating the quantity

$$\mathbb{E}_{x \sim D_f}\left[\frac{1}{D_f(x)} \mathbb{1}_{\{D_f(x) > \tau\}}\right],$$

for some suitable parameter $\tau > 0$. By capping the value of $1/D_f(x)$ to $20/(\varepsilon k)$, we can therefore only decrease the estimate, and by at most $20/(\varepsilon k) \cdot D_f(\{\, x : D_f(x) > (\varepsilon k)/20 \,\}) \leq 20/(\varepsilon k)$.

The condition for their algorithm to estimate support size to within $\pm \varepsilon m$ is that all elements in the support have a probability mass of at least $1/m$. Since each nonempty interval has length at least 1, we have $\min_j D_f(j) \geq (1/Ck)$. In order for their algorithm to report an estimate within $\pm \varepsilon k/20$ of support size, we set $\varepsilon' = (\varepsilon/20C)$ in their algorithm.

The total error in support size is at most $\varepsilon k/20 + \varepsilon k/20 = \varepsilon k/10$. By Observation 5.3, this suffices to test ε-test k-monotonicity of f.

Using the algorithm of [15], we need $O(1/\varepsilon'^2) = O\big((C/\varepsilon)^2\big)$ queries to D_f. For every query to D_f, we need to make $O(C/\varepsilon)$ queries to f, so the overall query complexity is $O\big(C^3/\varepsilon^3\big)$. □

Proof of Claim 5.2. The last part of the statement is immediate from the first, so it suffices to prove the first implication. We show the contrapositive: assuming f is $(1+\varepsilon)k$-monotone, then f can be fixed into a k-monotone function by changing at most εn points.

Let ℓ^* be the minimum integer ℓ for which f is ℓ-monotone: we can assume $\varepsilon k \geq 1$ and $k < \ell^* \leq (1+\varepsilon)k$ (as if $\ell^* \leq k$ we are done.) Consider as above the maximal consecutive monochromatic intervals $I_1, \dots, I_{\ell^*}$. Let S be the set of the $(\ell^* - k)$ shortest intervals. Note that intervals in S occupy at most an

$(\ell^* - k)/\ell^* \le \varepsilon k/k = \varepsilon$ fraction of the domain. Thus it suffices to show modifying intervals in S will produce a k-monotone function.

Observe that the partial function formed by intervals not in S is k-monotone (as only k intervals are involved). Let I_j be the first interval which is not in S. We go over intervals in S in ascending order and for each interval, if its index i is smaller than j, then we set the points in I_i to have the same value as the points in I_j, otherwise, we set them to have the same value as the points in I_{i-1}. As each modification does not increase the number of maximal consecutive monochromatic intervals, this procedure produces a k-monotone function. □

5.2.2 Reducing $[n] \to \{0,1\}$ to $[Ck] \to \{0,1\}$

Now we show how to reduce ε-testing k-monotonicity of $f\colon [n] \to \{0,1\}$ to ε'-testing k-monotonicity of a function $g\colon [Ck] \to \{0,1\}$ for $C = \mathrm{poly}(1/\varepsilon)$ and $\varepsilon' = \mathrm{poly}(\varepsilon)$, resulting in a $\mathrm{poly}(1/\varepsilon)$-query algorithm for ε-testing k-monotonicity.

The first step is (as before) to divide $[n]$ into blocks (disjoint intervals) of size $\varepsilon n/(4k)$ if $\varepsilon > 8k/n$ (again assuming without loss of generality that $\varepsilon n/(4k)$ is an integer), and blocks of size 1 otherwise (in which case $n \le 8k/\varepsilon$ and we can directly apply the result of Claim 5.1, with $C = n/k \le 8/\varepsilon$). Let $m = 4k/\varepsilon$ be the number of resulting blocks, and define $f_m\colon [n] \to \{0,1\}$ as the *m-block-coarsening* of f: namely, for any $j \in B_i$, we set

$$f_m(j) = \mathrm{argmax}_{b\in\{0,1\}} \Pr_{k\in B_i}[f(k) = b]\,. \qquad \text{(majority vote)}$$

Ordering the blocks $B_1, B_2, \ldots, B_m$, we also define $g\colon [m] \to \{0,1\}$ such that $g(i) = f_m(a)$ where $g(i)$ is set to be the constant value f_m takes over the i^{th} block.

It is easy to see that if f is k-monotone, then f has at most k non-constant blocks, and f_m is k-monotone. Because the function f only changes values k times; for a block to be non-constant, the block must contain a pair of points with a value change. We call a block *variable* if the minority points comprise at least an $\varepsilon/100$-fraction of the block; formally, B is variable if $\min_{b\in\{0,1\}} \Pr_{j\in B}[f(j) = b] \ge \varepsilon/100$.

We need the following claims (their proofs are at the end of the section) to prove Theorem 1.5.

Claim 5.8. *Suppose f has s variable blocks. Then* $\mathrm{dist}(f, f_m) \le s/m + \varepsilon/100$.

Claim 5.9. *Suppose f is promised to be either (i) k-monotone or (ii) such that f_m has more than $\frac{5}{4}k$ variable blocks. Then we can determine which with $O((1/\varepsilon^2)\log(1/\varepsilon))$ queries, and probability* $9/10$.

Proof of Theorem 1.5. We use the estimation/test from the previous claim as the first part of our tester. In more detail, note that in case (ii) in Claim 5.9 above, the test simply rejects f. Otherwise in case (i), either f is k-monotone or it has at most $5k/4$ variable blocks. If f passes, we can assume that f_m has less than $(5/4)k$ variable blocks. By Claim 5.8,

$$\mathrm{dist}(f, f_m) \le \frac{5k}{4} \Big/ m + \frac{\varepsilon}{100} = \frac{5\varepsilon}{16} + \frac{\varepsilon}{100} \le \frac{\varepsilon}{3}\,.$$

This part takes $O\big((1/\varepsilon^2)\log(1/\varepsilon)\big)$ queries.

Now, we apply the tester of Claim 5.1 (with probability of success amplified to $9/10$ by standard arguments) to $(\varepsilon/6)$-test k-monotonicity of $g\colon [m] \to \{0,1\}$, where $g(i)$ is the constant value of f_m on B_i, and $m = (4k)/\varepsilon$. Let q be the query complexity of the tester, and set $\delta = 1/(10q)$; to query $g(i)$, we randomly query f on $O((1/\varepsilon)\log(1/\delta))$ points in B_i and take the majority vote. With probability at least $1-\delta$, we get the correct value of $g(i)$, and by a union bound all q simulated queries have the correct value with probability at least $9/10$.

Therefore, to get a single query to g, we use $O((\log q)/\varepsilon)$ queries. In the context of our previous section, we have $C = 4/\varepsilon$, so $q = O(C^3/\varepsilon^3) = O\big(1/\varepsilon^6\big)$ and the overall query complexity of this part is $O((q\log q)/\varepsilon) = O((1/\varepsilon^7)\log(1/\varepsilon))$. This dominates the query complexity of the other part of the tester, from Lemma 5.9, which is $O((1/\varepsilon^2)\log(1/\varepsilon))$. By a union bound over the part from Lemma 5.9, the simulation of g, and the call to the tester of Claim 5.1, the algorithm is correct with probability at least $1-3/10 > 2/3$. □

Proof of Lemma 5.8. We will estimate the error of f_m in computing f on variable blocks and non-variable blocks separately. Each non-variable block B can contribute error on at most $\varepsilon\,|B|\,/100$ points. Each variable block B can contribute error on at most $|B| = n/m$ points. The total number of errors is at most $\varepsilon n/100 + s(n/m) = n(\varepsilon/100 + s/m)$, yielding the upper bound on $\operatorname{dist}(f, f_m)$. □

Proof of Lemma 5.9. We first note that given any fixed block B, it is easy to detect whether it is variable (with probability of failure at most δ) by making $O((1/\varepsilon)\log(1/\delta))$ uniformly distributed queries in B. Doing so, a variable block will be labelled as such with probability at least $1-\delta$, while a constant block will never be marked as variable. (If a block is neither constant nor variable, then any answer will do.)

Letting s denote the number of variable blocks, we then want to non-adaptively distinguish between

$$s \geq \frac{5}{4}k = \frac{5\varepsilon}{16}m \quad \text{and} \quad s \leq k = \frac{\varepsilon}{4}m$$

(since if f were k-monotone, then f_m had at most k variable blocks). Doing so with probability at least $19/20$ can be done by checking only $q = O(1/\varepsilon)$ blocks chosen uniformly at random: by the above, setting $\delta = 1/(20q)$ all of the q checks will also yield the correct answer with probability no less than $9/10$, so by a union bound we will distinguish (i) and (ii) with probability at least $9/10$. We conclude by observing that all

$$O\left(q \cdot \frac{1}{\varepsilon}\log\frac{1}{q}\right) = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$$

queries are indeed non-adaptive. □

6 On the grid

We now turn to the grid, and consider k-monotonicity of functions defined on $[n]^2$. More specifically, in this section we prove Theorem 1.6, giving an adaptive tester for 2-mononicity with optimal query complexity, before discussing in Section 6.2 possible extensions of these ideas.

6.1 The case $k = 2$

Theorem 1.6. *There exists a two-sided adaptive tester for* 2*-monotonicity of functions* $f : [n]^2 \to \{0,1\}$ *with query complexity* $q(n,\varepsilon) = O(1/\varepsilon)$.

Proof. At a high-level, the algorithm relies on two key components: the first is the observation that testing 2-monotonicity of $f : [n]^2 \to \{0,1\}$ *under some suitable additional assumption on* f reduces to (tolerant) testing monotonicity of two *one-dimensional* functions (but with larger range), under the L_1 norm. The second is that, given access to an arbitrary f, one can efficiently provide query access to some function g which satisifies this additional assumption, and such that g will also be close to f whenever f is truly 2-monotone.

Combining the two then enables one to test this function g for 2-monotonicity, and then check whether it is also the case that f and g are sufficiently close. The first step, by the above, can be done efficiently by simulating query access to g, which (with some additional tricks) in turn allows to simulate access to the corresponding one-dimensional functions: and invoke on these two functions the L_1-tester of [8]. (The main challenges there lies in performing this two-level simulation while keeping the number of queries to f low enough; which we achieve by carefully amortizing the queries made overall.)

Details. Hereafter, we assume without loss of generality that f is identically 0 on the bottom and top rows, that is $f(1,j) = f(n,j) = 0$ for all $j \in [n]$. (Indeed, we can ensure this is the case by adding two extra rows, extending the domain of f to $[n+2] \times [n]$: note that f remains 2-monotone if it was already, and can only decrease its distance to 2-monotonicity by $O(1/n)$).[9] For the sake of the proof, we will require the notion of 2*-column-wise-monotonicity*, defined below:

Definition 6.1. A function $f : [n]^2 \to \{0,1\}$ is said to be 2*-column-wise-monotone* if, for every $j \in [n]$, its restriction $f_j : [n] \times \{j\} \to \{0,1\}$ is 2-monotone. Given such a function f, we define the two sequences $(\bar{\partial} f_j)_{j\in[n]}$ and $(\underline{\partial} f_j)_{j\in[n]}$ as the sequence of "changepoints" in the columns. More formally, we define

$$\underline{\partial} f_j = \min\{\, i \in [n] \,:\, f(i,j) \neq f(1,j) \,\} - 1, \qquad \bar{\partial} f_j = \max\{\, i \in [n] \,:\, f(i,j) \neq f(n,j) \,\} + 1$$

for every j such that $f|_{[n]\times\{j\}}$ is not constant. If the column $f|_{[n]\times\{1\}}$ is constant, let $\underline{\partial} f_1 = 1$ and $\bar{\partial} f_1 = n$; further, if for some $j > 1$ the column $f|_{[n]\times\{j\}}$ is constant, let $\underline{\partial} f_j = \underline{\partial} f_{j-1}$ and $\bar{\partial} f_j = \bar{\partial} f_{j-1}$. Note that $\underline{\partial} f_j \leq \bar{\partial} f_j$ for every column $j \in [n]$.

As it turns out, testing 2-monotonicity of functions *guaranteed to be* 2*-column-wise-monotone* reduces to testing monotonicity of these two specific subsequences:

Lemma 6.2. *Let* $f : [n]^2 \to \{0,1\}$ *be* 2*-column-wise-monotone. If* f *is* 2*-monotone, then both sequences* $(\underline{\partial} f_j)_{j\in[n]}$ *and* $(\bar{\partial} f_j)_{j\in[n]}$ *are non-increasing. Moreover, if* f *is* ε*-far from* 2*-monotone, then at least one of the two sequences is* $\varepsilon/2$*-far from non-increasing (in Hamming distance).*

Proof. The first part of the theorem is straightforward (by contrapositive, if at least one of the two sequences is not non-increasing then we can find a violation of 2-monotonicity). We thus turn to the second part, and show the contrapositive; for this purpose, we require the following result:

[9]This will be used in the proof of Lemma 6.2 and Lemma 6.4.

Claim 6.3. *If $f\colon [n]^2 \to \{0,1\}$ is a 2-column-wise monotone function such that (i) $f(1,j) = f(n,j) = 0$ for all $j \in [n]$ and (ii) both $(\underline{\partial} f_j)_{j\in[n]}, (\bar{\partial} f_j)_{j\in[n]} \subseteq [n]$ are non-increasing, then f is 2-monotone.*

Proof. By contradiction, suppose there exists a 2-column-wise monotone function f satisfying (i) and (ii), which is not 2-monotone. This last point implies there exists a triple of comparable elements $x = (i_x, j_x) \prec y = (i_y, j_y) \prec z = (i_z, j_z)$ constituting a violation, i. e., such that $(f(x), f(y), f(z)) = (1,0,1)$. Moreover, since (i) holds we must have $1 < i_x \le i_y \le i_z < n$; more precisely, $1 \le \underline{\partial} f_{j_x} < i_x \le i_y \le i_z < \bar{\partial} f_{j_z} \le n$. As $x \prec y \prec z$, we have $j_x \le j_y \le j_z$, which by the non-increasing assumption (ii) implies that $\underline{\partial} f_{j_x} \ge \underline{\partial} f_{j_y}$ and $\bar{\partial} f_{j_y} \ge \bar{\partial} f_{j_z}$. But this is not possible, as altogether this leads to $\underline{\partial} f_{j_y} < i_y < \bar{\partial} f_{j_y}$, i. e., $f(y) = 1$. □

Assume both sequences $(\underline{\partial} f_j)_{j\in[n]}, (\bar{\partial} f_j)_{j\in[n]} \subseteq [n]$ are $\varepsilon/2$-close to non-increasing, and let $L, H \subset [n]$ (respectively) be the set of indices where the two sequences need to be changed in order to become non-increasing. By assumption, $|L|, |H| \le \varepsilon n/2$, so $|L \cup H| \le \varepsilon n$. But to "fix" a value of $(\underline{\partial} f_j)_{j\in[n]}$ or $(\bar{\partial} f_j)_{j\in[n]}$ requires to change the values of the function f inside a single column—and this can be done preserving its 2-column-wise-monotonicity, so that changing the value of f on at most n points is enough. It follows that making both $(\underline{\partial} f_j)_{j\in[n]}$ and $(\bar{\partial} f_j)_{j\in[n]}$ non-increasing requires to change f on at most εn^2 points, and with Claim 6.3 this results in a function which is 2-monotone. Thus, f is ε-close to 2-monotone. □

It is possible to refine the above statement to obtain, in the second case, a more precise characterization in terms of the L_1 *distance* of these two sequences to monotonicity. For conciseness, in the rest of this section we denote by $\mathcal{M}_k^{(d)}$ the class of k-monotone functions on $[n]^d$, and will omit the subscript when $k = 1$ (i. e., for monotone functions: $\mathcal{M}^{(d)} = \mathcal{M}_1^{(d)}$).

Lemma 6.4. *Let $f\colon [n]^2 \to \{0,1\}$ be 2-column-wise-monotone. If f is ε-far from 2-monotone then at least one of the two sequences is $\varepsilon/2$-far from non-increasing (in L_1 distance). More precisely:*

$$\operatorname{dist}\Big(f, \mathcal{M}_2^{(2)}\Big) \le L_1(\bar{\partial} f, \mathcal{M}^{(1)}) + L_1(\underline{\partial} f, \mathcal{M}^{(1)})\,. \tag{6.1}$$

Proof. Recall that we aim at establishing the following:

$$\operatorname{dist}\Big(f, \mathcal{M}_2^{(2)}\Big) \le L_1(\bar{\partial} f, \mathcal{M}^{(1)}) + L_1(\underline{\partial} f, \mathcal{M}^{(1)})\,. \tag{6.2}$$

For notational convenience, we will view in this proof the sequences $(\underline{\partial} f)_j, (\bar{\partial} f)_j$ as functions

$$\underline{\partial} f, \bar{\partial} f\colon [n] \to [n]\,.$$

Let $\ell, h\colon [n] \to [n]$ (for "low" and "high," respectively) be monotone functions achieving $L_1(\underline{\partial} f, \mathcal{M}^{(1)})$ and $L_1(\bar{\partial} f, \mathcal{M}^{(1)})$, respectively.

- As $\underline{\partial} f(j) \le \bar{\partial} f(j)$ for all $j \in [n]$, we will assume $\ell(j) \le h(j)$ for all j. Otherwise, one can consider instead the functions $\ell' = \min(\ell, h)$ and $h' = \max(\ell, h)$: both will still be monotone (non-increasing), and by construction

$$\left|\ell'(j) - \underline{\partial} f(j)\right| + \left|h'(j) - \bar{\partial} f(j)\right| \le |\ell(j) - \underline{\partial} f(j)| + \left|h(j) - \bar{\partial} f(j)\right|$$

  for all $j \in [n]$, so that $L_1(\bar{\partial} f, \ell') + L_1(\underline{\partial} f, h') \le L_1(\bar{\partial} f, \ell) + L_1(\underline{\partial} f, h)$.

- From ℓ and h, we can define a 2-column-wise monotone function $g\colon [n]^2 \to [n]$ such that $\underline{\partial} g = \ell$ and $\bar{\partial} g = h$: that is,
$$g(i,j) = \begin{cases} 0 & \text{if } i \geq h(j), \\ 1 & \text{if } \ell(j) < i < h(j), \\ 0 & \text{if } i \leq \ell(j), \end{cases}$$
for $(i,j) \in [n]^2$.

It is clear that g is 2-column-wise monotone with $g(1,j) = g(n,j) = 0$ for all $j \in [n]$; since by construction $\underline{\partial} g, \bar{\partial} g$ are non-increasing, we can invoke Claim 6.3 to conclude g is 2-monotone. It remains to bound the distance between f and g: writing $\Delta_j \in \{0,\dots,n\}$ for the number of points on which f and g differ in the j-th column, we have

$$\begin{aligned} \operatorname{dist}\left(f, \mathcal{M}_2^{(2)}\right) \leq \operatorname{dist}(f,g) &= \frac{1}{n^2}\sum_{j=1}^n \Delta_j \leq \frac{1}{n^2}\sum_{j=1}^n \left(|\ell(j) - \underline{\partial} f(j)| + \left|h(j) - \bar{\partial} f(j)\right|\right) \\ &= \frac{1}{n^2}\sum_{j=1}^n |\ell(j) - \underline{\partial} f(j)| + \frac{1}{n^2}\sum_{j=1}^n \left|h(j) - \bar{\partial} f(j)\right| = L_1(\underline{\partial} f, \ell) + L_1\left(\bar{\partial} f, h\right) \\ &\leq L_1(\underline{\partial} f, \mathcal{M}^{(1)}) + L_1(\bar{\partial} f, \mathcal{M}^{(1)}) \end{aligned}$$

which concludes the proof. □

Having established these two lemmata, we turn to the proof of Theorem 1.6. We first describe a non-optimal tester making $O(1/\varepsilon^2)$ queries; before explaining how to modify it in order to amortize the number of queries, to yield the desired $O(1/\varepsilon)$ query complexity.

Suppose we are given query access to an arbitrary function $f\colon [n]^2 \to \{0,1\}$. For simplicity, as before we assume without loss of generality that $\varepsilon n/16$ is an integer, and partition each column into $K \overset{\text{def}}{=} 16/\varepsilon$ intervals of size $\varepsilon n/16$, that is partition the domain $[n] \times [n]$ into a grid $[16/\varepsilon] \times [n]$ (each column being divided in K blocks $B_1,\dots,B_K$ of size $\varepsilon n/16$). This uniquely defines a function $g\colon [n]^2 \to \{0,1,*\}$: for any point $x = (i,j) \in [n]^2$, we let $\ell \in [K]$ be the index such that $i \in B_\ell = \{b_\ell,\dots,b_{\ell+1}-1\}$, and set:

- $g(x) = f(i,b_\ell)$, if $f(i,b_\ell) = f(i,b_{\ell+1}-1)$;
- $g(x) = *$, if $f(i,b_\ell) \neq f(i,b_{\ell+1}-1)$;

so that g is constant on any "block" $B_\ell \times \{i\}$. Note that we can provide query access to g, at the price of an overhead of 2 queries (to f) per query (to g).

However, this g may not itself be 2-column-wise-monotone; for this reason, we will instead work with a "fixed" version of g which will by construction be 2-column-wise-monotone. In more detail, we define $\widetilde{g}$ to be the 2-column-wise-monotone function obtained by the following process.

- First, we (arbitrarily) set the values $*$ to 0, so that g becomes a function $g\colon [n]^2 \to \{0,1\}$.

- Define $\widetilde{g}$ by its restriction on each column: letting $(\bar{\partial} g_j)_{j\in[n]}$ and $(\underline{\partial} g_j)_{j\in[n]}$ be as defined in Definition 6.1 (observing that the quantities are well and uniquely defined even if g is not 2-column-wise-monotone), set
$$\widetilde{g}(i,j) = \begin{cases} g(1,j) & \text{if } i \le \underline{\partial} g_j, \\ 1-g(1,j) & \text{if } \underline{\partial} g_j < i < \bar{\partial} g_j, \\ g(n,j) & \text{if } i \ge \bar{\partial} g_j. \end{cases}$$

From this construction, it is clear that $\widetilde{g}$ is 2-column-wise-monotone, and entirely and deterministically determined by g (and therefore by f); moreover we have $\widetilde{g} = g$ whenever g is itself 2-column-wise-monotone. Furthermore, any query to $\widetilde{g}$ can straightforwardly be answered by making at most queries $O(1/\varepsilon)$ to g, and hence to f.

- If f is 2-monotone, then g is $\varepsilon/8$-close to f and so is $\widetilde{g}$; moreover, $\widetilde{g}$ is 2-monotone as well. Therefore $\operatorname{dist}(f,\widetilde{g}) \le \varepsilon/8$ and $\operatorname{dist}\left(\widetilde{g},\mathcal{M}_2^{(2)}\right) = 0$.
- If f is ε-far from 2-monotone, then $\widetilde{g}$ is either (i) $\varepsilon/4$-far from f or (ii) $3\varepsilon/4$-far from 2-monotone, since if neither hold then by the triangle inequality f is ε-close to 2-monotone.

The tester (first take). The algorithm now proceeds as follows:

1. simulate query access to $\widetilde{g}$ (defined as above) to detect if
$$\operatorname{dist}\left(\widetilde{g},\mathcal{M}_2^{(2)}\right) \ge \varepsilon$$
using Equation (6.1), with probability of failure $1/6$. More precisely, test monotonicity in L_1 distance of both $\underline{\partial}\widetilde{g}$ and $\bar{\partial}\widetilde{g}$ with parameter $\varepsilon/64$, using the (non-adaptive) algorithm of [8]; and reject if any of these two tests rejects.
   - If $\operatorname{dist}\left(f,\mathcal{M}_2^{(2)}\right) = 0$ (and so in particular $\widetilde{g}$ is 2-monotone as well by the first bullet above), then
     (a) $L_1(\bar{\partial}\widetilde{g},\mathcal{M}^{(1)}) = L_1(\underline{\partial}\widetilde{g},\mathcal{M}^{(1)}) = 0$, And
     (b) Estimating that $\operatorname{dist}(f,\widetilde{g}) \le \frac{\varepsilon}{8}$ vs $\operatorname{dist}(f,\widetilde{g}) > \frac{\varepsilon}{4}$ reveals the correct answer (the former) with probability at least $5/6$

     by Lemma 6.2, and both tests will accept.
   - If $\operatorname{dist}\left(\widetilde{f},\mathcal{M}_2^{(2)}\right) > \varepsilon$, (and so in particular $\widetilde{g}$ *is possibly* $\frac{3\varepsilon}{4}$ far from 2-monotone) then
     (a) $\max(L_1(\bar{\partial}\widetilde{g},\mathcal{M}^{(1)}), L_1(\underline{\partial}\widetilde{g},\mathcal{M}^{(1)})) > \frac{3\varepsilon}{8}$, Or
     (b) Estimating that $\operatorname{dist}(f,\widetilde{g}) \le \frac{\varepsilon}{8}$ vs $\operatorname{dist}(f,\widetilde{g}) > \frac{\varepsilon}{4}$ reveals the correct answer (the latter) with probability at least $5/6$

     So, in this case, at least one of the two tests rejects.
2. return ACCEPT if both of the two tests above passed, REJECT otherwise.

By a union bound, the tester is then correct with probability at least $2/3$; its query complexity is

$$2t \cdot O\left(\frac{1}{\varepsilon}\right) + \left(t' \cdot O\left(\frac{1}{\varepsilon}\right) + O\left(\frac{1}{\varepsilon}\right)\right)$$

where t and t' are respectively the cost of simulating query access to $\bar{\partial} g, \underline{\partial} g$, and to $\widetilde{g}$; as the first step, testing in L_1 for for functions defined on the line $[n]$, has query complexity $O(1/\varepsilon)$ from [8]. Taking $t = t' = O(1/\varepsilon)$ as discussed above then results in a query complexity of $O\big(1/\varepsilon^2\big)$.

However, as mentioned previously this is not optimal: as we shall see, we can modify this to obtain instead an $O(1/\varepsilon)$ query complexity. In order to *amortize* the overall query complexity, we define the following process that specifies a 2-column-wise-monotone function $\mathring{g}$:

The Amortized Tester The idea here is to have our previous tester simulate query access to a new function $\mathring{g}$ instead $\widetilde{g}$. Simulating a single query $\widetilde{g}$ took $O(1/\varepsilon)$ queries to f and this is where amortizing queries using $\mathring{g}$ reduces the number of queries. Details follow.

- *(Initialization)* Let $j_1, \dots, j_{1/\varepsilon+1} \in [n]$ be the indices defined by $j_\ell = (\ell - 1) \cdot \varepsilon n + 1$ for $\ell \in [1/\varepsilon]$, and $j_{1/\varepsilon+1} = n$.
- *($\mathring{g}$ on j_1-st column)* Obtain all the values of $\widetilde{g}$ on the j_1-st column $[n] \times \{j_1\}$, at a cost of $O(1/\varepsilon)$ queries to f, to find $\bar{\partial}\mathring{g}_{j_1}, \underline{\partial}\mathring{g}_{j_1}$. Define $\mathring{g}$ on this column accordingly.
- *($\mathring{g}$ on remaining columns)* Assuming $\mathring{g}$ has been defined on the j_ℓ-th column, define it on the $j_{\ell+1}$-th column: starting at the "vertical" positions of the two changepoints $\bar{\partial}\mathring{g}_{j_\ell}, \underline{\partial}\mathring{g}_{j_\ell}$ of the previous column, start querying "downwards" the values of g on the $j_{\ell+1}$-th column until candidates values for $\bar{\partial}\mathring{g}_{j_{\ell+1}}, \bar{\partial}\mathring{g}_{j_{\ell+1}}$ consistent with g are found or the bottom of the column is reached (in which case the corresponding changepoint $\bar{\partial}\mathring{g}_{j_{\ell+1}}$ or $\underline{\partial}\mathring{g}_{\ell+1}$ is set to 1). After this, define $\mathring{g}$ on the $j_{\ell+1}$-th column to be consistent with these (at most) two changepoints.

Note that the queries made in this step are adaptive, and the (partial) function $\mathring{g}$ obtained at the end coincides with $\widetilde{g}$ if f (and therefore $\widetilde{g}$) is indeed 2-monotone. This is because in this case, by Lemma 6.2 the sequences $(\bar{\partial}\widetilde{g}_j)_j, (\underline{\partial}\widetilde{g}_j)_j$ will be non-decreasing, and therefore the process outlined above will result in $\bar{\partial}\widetilde{g}_{j_\ell} = \bar{\partial}\mathring{g}_{j_\ell}$ and $\underline{\partial}\widetilde{g}_{j_\ell} = \underline{\partial}\mathring{g}_{j_\ell}$ for all $1 \leq \ell \leq 1/\varepsilon + 1$. Moreover, it is not difficult to see that the total number of queries made to f in this initialization step will be $O(1/\varepsilon)$: this is because of the fact that we only search for the current changepoint $\bar{\partial}\mathring{g}_{j_\ell}$ (resp. $\underline{\partial}\mathring{g}_{j_\ell}$) starting at the position of the previous one $\bar{\partial}\mathring{g}_{j_{\ell-1}}$ (resp. $\underline{\partial}\mathring{g}_{j_{\ell-1}}$), going downwards only. Since to obtain a changepoint we only query the positions of g (and therefore f) at block endpoints, there are in total at most $1/\varepsilon$ positions to query where starting at one and then only going "down." Thus, the number of queries made for each column j_ℓ can be written as $O(1) + m_\ell$, where

$$\sum_{\ell=1}^{1/\varepsilon+1} m_\ell \leq K = O(1/\varepsilon)\,.$$

Query time. When querying the value of $\mathring{g}$ on a point (i, j), first let ℓ be the index such that $j_\ell \leq j < j_{\ell+1}$. Then define $\bar{\partial}\mathring{g}_j$ (resp. $\underline{\partial}\mathring{g}_j$) by querying the value of g on the j-th column for all at most $1/\varepsilon$

(block) indices between $\bar{\partial}\mathring{g}_{j_\ell}$ and $\bar{\partial}\mathring{g}_{j_{\ell+1}}$ (resp. $\underline{\partial}\mathring{g}_{j_\ell}$ and $\underline{\partial}\mathring{g}_{j_{\ell+1}}$) to find the corresponding candidate changepoints:

$$\underline{\partial}\mathring{g}_j = \min\left\{\, \underline{\partial}\mathring{g}_{j_{\ell+1}} \le i \le \underline{\partial}\mathring{g}_{j_\ell} \;:\; g(i,j) \neq g(1,j) \,\right\} - 1\,,$$
$$\bar{\partial}\mathring{g}_j = \max\left\{\, \bar{\partial}\mathring{g}_{j_{\ell+1}} \le i \le \bar{\partial}\mathring{g}_{j_\ell} \;:\; g(i,j) \neq g(n,j) \,\right\} + 1\,.$$

Again, note that after each query the (partial) function $\mathring{g}$ obtained so far will coincide with $\widetilde{g}$ if f (and therefore $\widetilde{g}$) is indeed 2-monotone; and $\mathring{g}$ is uniquely determined by f (and in particular does not depend on the actual queries made nor on their order). Finally, the function $\mathring{g}$ thus defined will always by construction be 2-column-wise-monotone.

The tester previously described can then be slightly modified to simulate access to $\mathring{g}$ instead of $\widetilde{g}$: by the above discussion, for the completeness case we will have the same guarantees as then $\mathring{g} = \widetilde{g}$, while the soundness case stays unchanged:

- If f is 2-monotone, then $\mathring{g} = \widetilde{g}$ is $\frac{\varepsilon}{8}$-close to f; so $\operatorname{dist}(f, \mathring{g}) \le \frac{\varepsilon}{8}$ and $\operatorname{dist}\left(\mathring{g}, \mathcal{M}_2^{(2)}\right) \le \frac{\varepsilon}{8}$.
- If f is ε-far from 2-monotone, then $\mathring{g}$ is either (i) $\frac{\varepsilon}{4}$-far from f or (ii) $\frac{3\varepsilon}{4}$-far from 2-monotone.

Thus, the analysis of correctness of the tester carries through with this modification; it only remains to bound the query complexity. We will show that the *expected* number of queries made is $O(1/\varepsilon)$; a bound on the worst-case query complexity will then follow from standard arguments,[10] at the price of a constant factor in the $O(\cdot)$.

To give this bound on the expected number of queries, we first observe that the algorithm from [8] we rely on in the first stage of the tester is non-adaptive, and moreover all the queries it makes are uniformly distributed (as it works by a reduction, invoking the non-adaptive, one-sided, sample-based monotonicity tester for functions $[n] \to \{0,1\}$). Similarly, all the queries made in the second stage are uniformly distributed as well.

Therefore, the expected number of queries a_ℓ made to columns with indices $j_\ell \le j < j_{\ell+1}$ is the same for each $\ell \in [1/\varepsilon + 1]$, namely

$$a_\ell = \frac{q}{1/\varepsilon} = O(1)\,.$$

where $q = O(1/\varepsilon)$ is the total number of queries made to $\mathring{g}$ (and/or to $\bar{\partial}\mathring{g}, \underline{\partial}\mathring{g}$) during the second phase of "Query time." Now, letting $m_\ell = \underline{\partial}\mathring{g}_{j_\ell} - \underline{\partial}\mathring{g}_{j_{\ell+1}}$ and $m'_\ell = \bar{\partial}\mathring{g}_{j_\ell} - \bar{\partial}\mathring{g}_{j_{\ell+1}}$, we have that the total expected cost of simulating these queries (in terms of queries to f) is upper bounded by

$$\sum_{\ell=1}^{\frac{1}{\varepsilon}} a_\ell \cdot \left(O(1) + m_\ell + m'_\ell\right) = O\left(\frac{1}{\varepsilon}\right) + O(1) \cdot \sum_{\ell=1}^{\frac{1}{\varepsilon}} m_\ell + O(1) \cdot \sum_{\ell=1}^{\frac{1}{\varepsilon}} m'_\ell = O\left(\frac{1}{\varepsilon}\right).$$

Since the total number of queries made to f is the sum of the number of queries made to partly build $\mathring{g}$ during the "Initialization phase" (which is $O(1/\varepsilon)$ by the foregoing discussion), the number of queries made to simulate access to $\mathring{g}$ or $\bar{\partial}\mathring{g}, \underline{\partial}\mathring{g}$ during the "Query time" (which was just shown to be $O(1/\varepsilon)$ in expectation), and the number of queries directly made to f when testing the distance of f to $\mathring{g}$ (which is also $O(1/\varepsilon)$), the expected total number of queries is indeed $O(1/\varepsilon)$, as claimed. □

[10] Namely, stopping the algorithm and outputting REJECT if the number of queries made exceeds C/ε for some absolute constant $C > 0$, found by applying Markov's inequality.

6.2 Possible extensions

We now discuss two possible extensions of the techniques underlying Theorem 1.6, namely to (i) k-monotonicity testing of functions over $[n]^2$, for general k; and (ii) 2-monotonicity testing of functions over $[n]^d$, for general d. (Note that we do provide a different tester for general k and d in the next section, Section 7).

Extending to general k (for $d = 2$). A natural direction would be to first to generalize Definition 6.1 to *k-column-wise monotone functions* f, defining the k sequences

$$(\underline{\partial} f_j^{(1)})_{j\in[n]}, \dots, (\underline{\partial} f_j^{(k)})_{j\in[n]}$$

of column changepoints with $\underline{\partial} f_j^{(1)} \le \cdots \le \underline{\partial} f_j^{(k)}$ for all $j \in [n]$. The next step would then be to obtain an analogue of the key lemma of the previous section, Lemma 6.4 to this setting. An issue is that it appears necessary to consider now the L_1 distance to $(k-1)$*-monotonicity* of these k sequences, instead of monotonicity as before. Thus, taking this route requires to generalize the definition of k-monotonicity to real-valued functions, but also to develop L_1-testers *for k-monotonicity* over the line.

The testing algorithm now follows the same outline as in the previous section, with the same "amortizing" idea when invoking this newly obtained L_1-tester for k-monotonicity in parallel on the k subsequences, each with probability of failure $\delta = 1/(10k)$ (for a union bound) and approximation parameter $\varepsilon' = \varepsilon/(10k)$. (Note that some more optimizations may then help further reduce the query complexity, by "sharing" the same set of queries between the k instances of the L_1-testing algorithm.)

Extending to general d (for $k = 2$). At a very high-level, the tester of Section 6.1 works by reducing 2-monotonicity testing of $f\colon [n]^2 \to \{0,1\}$ to monotonicity L_1-testing of $\underline{\partial} f, \bar{\partial} f\colon [n] \to [0,1]$. More generally, one can hope to extend this approach to higher dimensions, reducing 2-monotonicity testing of $f\colon [n]^d \to \{0,1\}$ to monotonicity L_1-testing of $\underline{\partial} f, \bar{\partial} f\colon [n]^{d-1} \to [0,1]$: that is, testing monotonicity (in L_1) of the two $(d-1)$-dimensional "surfaces" of changepoints. This in turn could be done invoking the L_1 non-adaptive tester of [8] for monotonicity over $[n]^d$, which has query complexity $\widetilde{O}(d/\varepsilon)$: which may lead to a total query complexity of $\operatorname{poly}(d, 1/\varepsilon)$, that is *polynomial in the dimension*. We leave this possible extension as an interesting direction for future work.

7 On the high-dimensional grid

In this section, we give two algorithms for tolerant testing, that is testing whether a function $f\colon [n]^d \to \{0,1\}$ is ε_1-close to k-monotone vs. ε_2-far from k-monotone, establishing Theorem 1.8. The first has query complexity exponential in the dimension d and is *fully tolerant*, that is works for any setting of $0 \le \varepsilon_1 < \varepsilon_2 \le 1$. The second applies whenever $\varepsilon_2 > 3\varepsilon_1$, and has (incomparable) query complexity exponential in $\widetilde{O}(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2)$. Both of these algorithms can be used for non-tolerant ("regular") testing by setting $\varepsilon_1 = 0$ and $\varepsilon_2 = \varepsilon$, which implies Corollary 1.7.

Theorem 1.8. *There exist*

- *a non-adaptive (fully) tolerant tester for k-monotonicity of functions* $f\colon [n]^d \to \{0,1\}$ *with query complexity*
$$q(n,d,\varepsilon_1,\varepsilon_2,k) = \widetilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2-\varepsilon_1}\right)^d\right);$$
- *a non-adaptive tolerant tester for k-monotonicity of functions* $f\colon [n]^d \to \{0,1\}$ *with query complexity*
$$q(n,d,\varepsilon_1,\varepsilon_2,k) = \exp\Big(\widetilde{O}\Big(k\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\Big)\Big),$$
under the restriction that $\varepsilon_2 > 3\varepsilon_1$.

As a corollary, this implies Corollary 1.7, restated below:

Theorem 1.7. *There exists a non-adaptive two-sided tester for k-monotonicity of functions* $f\colon [n]^d \to \{0,1\}$ *with query complexity*
$$q(n,d,\varepsilon,k) = \min\left(\widetilde{O}\left(\frac{1}{\varepsilon^2}\left(\frac{5kd}{\varepsilon}\right)^d\right), \exp\Big(\widetilde{O}\Big(k\sqrt{d}/\varepsilon^2\Big)\Big)\right).$$

For convenience, we will view in this part of the paper the set $[n]$ as $[n] = \{0,1,\dots,n-1\}$. Assuming that m divides n, we let $\mathcal{B}_{m,n}\colon [n]^d \to [m]^d$ be the mapping such that $\mathcal{B}_{m,n}(y)_i = \lfloor my_i/n \rfloor$ for $1 \le i \le m$. For $x \in [m]^d$, we define the set $\mathcal{B}^{-1}_{m,n}(x)$ to be the inverse image of x. Specifically, $\mathcal{B}^{-1}_{m,n}(x)$ is the set of points of the form $m\cdot x + [n/m]^d$, with standard definitions for scalar multiplication and coordinate-wise addition. That is, $\mathcal{B}^{-1}_{m,n}(x)$ is a "coset" of $[n/m]^d$ points in $[n]^d$. To keep on with the notations of the other sections, we will call these cosets *blocks*, and will say a function $h\colon [n]^d \to \{0,1\}$ is an *m-block function* if it is constant on each block. Moreover, for clarity of presentation, we will omit the subscripts on $\mathcal{B}$ and $\mathcal{B}^{-1}$ whenever they are not necessary.

We first establish a lemma that will be useful for the proofs of correctness of both algorithms.

Lemma 7.1. *Suppose* $f\colon [n]^d \to \{0,1\}$ *is k-monotone. Then there is an m-block function* $h\colon [n]^d \to \{0,1\}$ *such that* $\operatorname{dist}(f,h) < kd/m$.

Proof. Fix any k-monotone function $f\colon [n]^d \to \{0,1\}$. We partition $[m]^d$ into chains of the form
$$C_x = \left\{\, x + \ell\cdot 1^d \;:\; \ell \in \mathbb{N}, x \in [m]^d \text{ and } x_i = 0 \text{ for some } i \,\right\}.$$
There are $m^d - (m-1)^d \le dm^{d-1}$ of these chains: we will show that f can only be nonconstant on at most k blocks of each chain.

By contradiction, suppose there exists $x \in [m]^d$ such that f is nonconstant on $k+1$ different blocks $\mathcal{B}^{-1}(z^{(i)})$, where $z^{(1)} \prec z^{(2)} \prec \cdots \prec z^{(k)} \prec z^{(k+1)}$, and each $z^{(i)} \in C_x$. By construction, we have $\mathcal{B}^{-1}(z^{(i)}) \prec \mathcal{B}^{-1}(z^{(j)})$ for $i<j$. For each $1 \le i \le k+1$, there are two points $v_*^{(i)}, v_{**}^{(i)} \in \mathcal{B}^{-1}(z_i)$ such that $v_*^{(i)} \prec v_{**}^{(i)}$ and $f(v_*^{(i)}) \ne f(v_{**}^{(i)})$. By construction $v_*^{(1)} \prec v_{**}^{(1)} \prec v_*^{(2)} \prec v_{**}^{(2)} \prec v_*^{(3)} \prec v_{**}^{(3)} \prec \cdots \prec v_*^{(k+1)} \prec v_{**}^{(k+1)}$, and there must be at least $k+1$ pairs of consecutive points with differing function values. Out of these $2k+2$

many points, there is a chain of points $\bar{v}^{(1)} \prec \bar{v}^{(2)} \prec \cdots \prec \bar{v}^{(k+1)}$ where $f(\bar{v}^{(i)}) \neq f(\bar{v}^{(i+1)})$ for $1 \leq i \leq k$, which is a violation of the k-monotonicity of f.

Thus, in each of the dm^{d-1} many chains of blocks, there can only be k nonconstant blocks. It follows that there are at most kdm^{d-1} nonconstant blocks in total. We now define $h(y)$ to be equal to $f(y)$ if f is constant on $\mathcal{B}(y)$, and arbitrarily set $h(y) = 0$ otherwise. Each set $\mathcal{B}^{-1}(y)$ contains $(n/m)^d = n^d \cdot m^{-d}$ many points, and f is not constant on at most kdm^{d-1} of these. It follows that $\operatorname{dist}(f,h) \leq kdm^{d-1} \cdot m^{-d} = kd/m$. □

7.1 Fully tolerant testing with $O(kd/(\varepsilon_2 - \varepsilon_1))^d$ queries

Our first algorithm (Algorithm 1) then proceeds by essentially brute-force learning an m-block function close to the unknown function, and establishes the first item of Theorem 1.8.

Algorithm 1 Fully tolerant testing with $O(kd/(\varepsilon_2 - \varepsilon_1))^d$ queries.

```
Require: Query access to $f\colon [n]^d \to \{0,1\}$, $\varepsilon_2 > \varepsilon_1 \geq 0$, a positive integer $k$
 1: $\alpha \leftarrow (\varepsilon_2 - \varepsilon_1), m \leftarrow \lceil 5kd/\alpha \rceil, t \leftarrow \lceil 25\ln(6m^d)/(2\alpha^2) \rceil$
 2: ▷ Define a distribution $D$ over $[m]^d \times \{0,1\}$.
 3: for $x \in [m]^d$ do
 4:     Query $f$ on $t$ random points $T_x \subseteq \mathcal{B}^{-1}(x)$.
 5:     $D(x,0) \leftarrow \Pr_{y \in T_x}[f(y) = 0]/m^d$
 6:     $D(x,1) \leftarrow \Pr_{y \in T_x}[f(y) = 1]/m^d$
 7: end for
 8: ▷ Define a distribution $D'$ over $[n]^d \times \{0,1\}$ such that $D'(y,b) = D(\mathcal{B}(y),b) \cdot m^d/n^d$.
 9: if there exists a $k$-monotone $m$-block function $h$ such that $\Pr_{(y,b)\sim D'}[h(y) \neq b] \leq \varepsilon_1 + \frac{\alpha}{2}$ then return
    ACCEPT
10: end if
11: return REJECT
```

Proposition 7.2. *Algorithm 1 accepts all functions ε_1-close to k-monotone functions, and rejects all functions ε_2-far from k-monotone (with probability at least $2/3$). Its query complexity is*

$$O\left(\frac{d}{(\varepsilon_2 - \varepsilon_1)^2}\left(\frac{5kd}{\varepsilon_2 - \varepsilon_1} + 1\right)^d \log \frac{kd}{\varepsilon_2 - \varepsilon_1}\right).$$

Proof. The algorithm first estimates $\Pr_{y \in \mathcal{B}^{-1}(x)}[f(y) = b]$ for every $x \in [m]^d$ and $b \in \{0,1\}$ to within $\pm\alpha/5$. We use $t = 25\ln(6m^d)/2\alpha^2$ points in each block to ensure (by an additive Chernoff bound) that each estimate is correct except with probability at most $m^{-d}/3$. By a union bound, the probability that all estimates are correct is at least $2/3$. Conditioning on this event, we have

$$\mathbb{E}_{(x,b)\sim D}\left[\Pr_{y \in \mathcal{B}^{-1}(x)}[f(y) \neq b]\right] = \Pr_{(y,b)\sim D'}[f(y) \neq b] \leq \frac{\alpha}{5}.$$

In this probability experiment, the marginal distribution of D' on y is uniform over $[n]^d$.

Let $f^* \colon [n]^d \to \{0,1\}$ be a k-monotone function minimizing $\Pr[f(y) \neq f^*(y)]$. Lemma 7.1 ensures that there is a k-monotone m-block function $h \colon [n]^d \to \{0,1\}$ such that $\operatorname{dist}(f^*, h) < kd/m \leq \alpha/5$. Let $h^* \colon [n]^d \to \{0,1\}$ be a k-monotone m-block function minimizing $\operatorname{dist}(f^*, h^*)$.

Completeness. Suppose $\operatorname{dist}(f, f^*) \leq \varepsilon_1$. Then by the triangle inequality,

$$\Pr_{(y,b)\sim D'}[h^*(y) \neq b] \leq \Pr_{(y,b)\sim D'}[h^*(y) \neq f^*(y)] + \Pr_{(y,b)\sim D'}[f^*(y) \neq f(y)] + \Pr_{(y,b)\sim D'}[f(y) \neq b] \leq \varepsilon_1 + \frac{2\alpha}{5}\,.$$

where to bound the first term $\Pr_{(y,b)\sim D'}[h^*(y) \neq f^*(y)]$ by $\operatorname{dist}(f^*, h^*) \leq \alpha/5$ we used the fact that the marginal distribution of y is uniform when $(y,b) \sim D'$. Thus, the algorithm will find a k-monotone m-block function close to D (without using any queries to f) and accept.

Soundness. Suppose $\operatorname{dist}(f, f^*) \geq \varepsilon_2$. Then by the triangle inequality

$$\begin{aligned}\Pr_{(y,b)\sim D'}[h(y) \neq b] &\geq \Pr_{(y,b)\sim D'}[h(y) \neq f(y)] - \Pr_{(y,b)\sim D'}[f(y) \neq b] \\ &\geq \Pr_{(y,b)\sim D'}[f^*(y) \neq f(y)] - \Pr_{(y,b)\sim D'}[f(y) \neq b] \geq \varepsilon_2 - \frac{\alpha}{5}\end{aligned}$$

for every k-monotone m-block function h. Since $\varepsilon_2 - 2\alpha/5 \geq \varepsilon_1 + 3\alpha/5$, the algorithm never find a k-monotone m-block function h with low error with respect to D, and the algorithm will reject.

Query complexity. The algorithm only makes queries in constructing D; the number of queries required is

$$m^d \cdot t = O\left(\frac{d}{\alpha^2}\left(\frac{5kd}{\alpha}+1\right)^d \log\frac{kd}{\alpha}\right). \qquad \square$$

7.2 Tolerant testing *via* agnostic learning

We now present our second algorithm, Algorithm 2, proving the second item of Theorem 1.8. At its core is the use of an *agnostic learning algorithm* for k-monotone functions, which we first describe.[11]

Proposition 7.3. *There exists an agnostic learning algorithm for k-monotone functions over* $[r]^d \to \{0,1\}$ *with excess error* τ *with sample complexity* $\exp(\widetilde{O}(k\sqrt{d}/\tau^2))$.

We start the proof of Proposition 7.3 by formally presenting the learning algorithm.

To prove its correctness we will rely on tools from Fourier analysis. For this reason, it will be convenient in this section to view the range as $\{-1,1\}$ instead of $\{0,1\}$.

The idea of the algorithm is to first attempt to learn the function, before checking whether what was learned is indeed close to both the unknown function and the set of k-monotone functions. To perform the

[11] Recall that an *agnostic learner with excess error* τ for some class of functions $\mathcal{C}$ is an algorithm that, given an unknown distribution D, an unknown arbitrary function f, and access to random labelled samples $\langle x, f(x)\rangle$ where $x \sim D$, satisfies the following. It outputs a hypothesis function $\hat{h}$ such that $\Pr_{x\sim D}\left[f(x) \neq \hat{h}(x)\right] \leq \textsc{opt}_D + \tau$ with probability at least $2/3$, where $\textsc{opt}_D = \min_{h\in\mathcal{C}} \Pr_{x\sim D}[f(x) \neq h(x)]$ (i. e., it performs "almost as well as the best function in $\mathcal{C}$").

Algorithm 2 Multiplicative approximation with $\exp\left(\widetilde{O}\left(k\sqrt{d}/(\varepsilon_2-3\varepsilon_1)^2\right)\right)$ queries

```
Require: Query access to $f\colon [n]^d \to \{0,1\}$, $\varepsilon_2 > 3\varepsilon_1 \geq 0$, a positive integer $k$
 1: $\alpha \leftarrow (\varepsilon_2 - 3\varepsilon_1)$, $m \leftarrow \lceil 6kd/\alpha \rceil$, $t \leftarrow \lceil 3d(k+1)/\alpha \ln m + \ln 100 \rceil$
 2: Define $D$ to be the distribution over $[m]^d \times \{0,1\}$ such that $D(x,b) = \Pr_{y \in \mathcal{B}^{-1}(x)}[f(y) = b]$.
 3: $\triangleright$ $\mathcal{A}_D(\tau, f)$ denotes the output of an agnostic learner of $k$-monotone functions with respect to $D$,
    with excess error $\tau$ and probability of failure $1/10$
 4: $h\colon [m]^d \to \mathbb{R} \leftarrow \mathcal{A}_D(\alpha/12, f)$.
 5: Estimate $\Pr_{(x,b)\sim D}[h(x) \neq b]$ to within $\pm\alpha/7$ with probability of failure $1/10$, using $O(1/\alpha^2)$
    queries.
 6: if the estimate is more than $\varepsilon_1 + \frac{5\alpha}{12}$ then return REJECT
 7: end if
 8: if $\mathrm{dist}(h,\ell) = \Pr_{x\in[m]^d}[h(x) \neq \ell(x)] \leq 2\varepsilon_1 + \frac{5\alpha}{12}$ for some $k$-monotone $m$-block function $\ell$ then return
    ACCEPT
 9: else return REJECT
10: end if
```

first step efficiently enough, we need to design an agnostic learning algorithm for k-monotone functions, which is what most of this section will be dedicated to. In more detail, we will rely on a theorem of [34], which very broadly speaking states that any class of functions C that can be well-approximated by linear combinations of a small number of functions admits a good agnostic learner (see Theorem 7.10 for the formal statement). Given this, our goal is to establish the existence of a small set of functions whose linear combinations "cover" the class of k-monotone functions. To do so, we first generalize a theorem of Blais et al. [11], to show that k-monotone functions on the hypergrid have small influence (Lemma 7.6). We then use this in Lemma 7.8 to get that k-monotone functions have Fourier concentration on relatively small degree coefficients, which we leverage in Lemma 7.9 to obtain our small set of functions.

7.3 Preliminary tools from Fourier analysis

Definition 7.4. For a Boolean function $f\colon [r]^d \to \{-1,1\}$, we define the *influence* of variable i on f as

$$\mathbf{Inf}_i[f] = 2\Pr\left[f(x) \neq f(x^{(i)})\right]$$

where $x = (x_1, x_2, \ldots, x_d)$ is a uniformly random string over $[r]^d$, and

$$x^{(i)} = (x_1, x_2, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_d)$$

for x' drawn independently and uniformly from $[r]$. We also define the *total influence* of the variables on f as $\mathbf{Inf}[f] = \sum_{i=1}^{d} \mathbf{Inf}_i[f]$.

We first generalize the following result, due to Blais *et al.*, to more general domains:

Proposition 7.5 ([11])**.** *Let $f\colon \{0,1\}^d \to \{-1,1\}$ be a k-monotone function. Then* $\mathbf{Inf}[f] \leq k\sqrt{d}$.

Lemma 7.6 (Generalization)**.** *Let $f\colon [r]^d \to \{-1,1\}$ be a k-monotone function. Then* $\mathbf{Inf}[f] \le k\sqrt{d}$.

Proof. For any two strings $y^0, y^1 \in [r]^d$, let $f_{y^0,y^1}\colon \{0,1\}^d \to \{-1,1\}$ be the function obtained by setting $f_{y^0,y^1}(x) = f(y^x)$, where $y^x \in [r]^d$ is defined as

$$y_i^x = \begin{cases} \min\{y_i^0, y_i^1\} & \text{if } x_i = 0, \\ \max\{y_i^0, y_i^1\} & \text{if } x_i = 1. \end{cases}$$

Since f was a k-monotone function, so is f_{y^0,y^1}. Thus $\mathbf{Inf}[f_{y^0,y^1}] \le k\sqrt{d}$ for every choice of y^0 and y^1. It is not hard to see that for any fixed $i \in [d]$ the following two processes yield the same distribution over $[r]^d \times [r]^d$:

- Draw $z \in [r]^d$, $z_i' \in [r]$ independently and *uar*, set $z' \overset{\text{def}}{=} (z_1, \dots, z_{i-1}, z_i', z_{i+1}, \dots, z_d)$, and output (z, z');
- Draw $y^0, y^1 \in [r]^d, x \in \{0,1\}^d$ independently and *uar*, and output $(y^x, y^{x^{(i)}})$.

This implies that

$$\begin{aligned}\mathbf{Inf}[f] &= \sum_{i=1}^d \mathbf{Inf}_i[f] = \sum_{i=1}^d 2 \Pr_{z\in[r]^d}[f(z) \ne f(z^{(i)})] = \sum_{i=1}^d 2\mathbb{E}_{y^0,y^1\in[r]^d}\left[\Pr_{x\in\{0,1\}^d}\left[f(y^x) \ne f(y^{x^{(i)}})\right]\right] \\ &= \mathbb{E}_{y^0,y^1\in[r]^d}\left[\sum_{i=1}^d 2 \Pr_{x\in\{0,1\}^d}\left[f(y^x) \ne f(y^{x^{(i)}})\right]\right] \\ &= \mathbb{E}_{y^0,y^1\in[r]^d}\left[\sum_{i=1}^d 2 \Pr_{x\in\{0,1\}^d}\left[f_{y_0,y_1}(x) \ne f_{y_0,y_1}(x^{(i)})\right]\right] \\ &= \mathbb{E}_{y^0,y^1}[\mathbf{Inf}[f_{y^0,y^1}]] \le \mathbb{E}_{y^0,y^1}[k\sqrt{d}] = k\sqrt{d}\,. \end{aligned}$$

□

For two functions $f, g\colon [r]^d \to \mathbb{R}$, we define the inner product $\langle f, g\rangle = \mathbb{E}_x[f(x)g(x)]$, where the expectation is taken with respect to the uniform distribution. It is known that for functions $f\colon [r]^d \to \mathbb{R}$, there is a "Fourier basis" of orthonormal functions f. To construct such a basis, we can take any orthonormal basis $\{\phi_0 \equiv 1, \phi_1, \dots, \phi_{|r|-1}\}$ for functions $f\colon [r] \to \mathbb{R}$. Given such a basis, a Fourier basis is the collection of functions ϕ_α, where $\alpha \in [r]^d$, and $\phi_\alpha(x) = \prod_{i=1}^d \phi_{\alpha_i}(x_i)$. Then every $f\colon [r]^d \to \mathbb{R}$ has a unique representation $f = \sum_{\alpha\in[r]^d} \widehat{f}(\alpha)\phi_\alpha$, where $\widehat{f}(\alpha) = \langle f, \phi_\alpha\rangle \in \mathbb{R}$.

Many Fourier formulæ hold in arbitrary Fourier bases, an important example being Parseval's Identity: $\sum_{\alpha\in[r]^d} \widehat{f}(\alpha)^2 = 1$. We will use the following property:

Lemma 7.7 ([43, Proposition 8.23])**.** *For $\alpha \in [r]^d$, let $|\alpha|$ denote the number of nonzero coordinates in α. Then we have*

$$\mathbf{Inf}[f] = \sum_{\alpha\in[r]^d} |\alpha| \hat{f}(\alpha)^2\,.$$

Lemma 7.8. *If* $\mathbf{Inf}[f] \le k$, *then* $\sum_{\alpha:|\alpha|>k/\varepsilon} \widehat{f}(\alpha)^2 \le \varepsilon$.

Proof. If not, then

$$\mathbf{Inf}[f] = \sum_{\alpha} |\alpha| \widehat{f}(\alpha)^2 \geq \sum_{\alpha:|\alpha|>k/\varepsilon} |\alpha| \widehat{f}(\alpha)^2 \geq \frac{k}{\varepsilon} \sum_{\alpha:|\alpha|>k/\varepsilon} \widehat{f}(\alpha)^2 > \frac{k}{\varepsilon} \cdot \varepsilon = k,$$

a contradiction. □

Lemma 7.9. *Let p be the function $\sum_{\alpha:|\alpha|\leq t} \widehat{f}(\alpha)\phi_\alpha$. Then*

(i) $\|p-f\|_2^2 = \mathbb{E}_{x\in[r]^d}[(p(x)-f(x))^2] = \sum_{\alpha:|\alpha|>t} \widehat{f}(\alpha)^2$;

(ii) p is expressible as a linear combination of real-valued functions over $[r]^d$, each of which only depends on at most t coordinates;

(iii) p is expressible as a degree-t polynomial over the rd indicator functions $\mathbb{1}_{\{x_i=j\}}$ for $1 \leq i \leq d$ and $j \in [r]$.

Proof. The lemma follows immediately from Parseval's identity and from the definitions. □

Theorem 7.10 ([34, Theorem 5])**.** *Let $\mathcal{C}$ be a class of Boolean functions over $\mathcal{X}$ and $\mathcal{S}$ a collection of real-valued functions over $\mathcal{X}$ such that for every $f\colon \mathcal{X} \to \{-1,1\}$ in $\mathcal{C}$, there exists a function $p\colon \mathcal{X} \to \mathbb{R}$ such that p is expressible as a linear combination of functions from $\mathcal{S}$ and $\|p-f\|_2^2 \leq \tau^2$. Then there is an agnostic learning algorithm for $\mathcal{C}$ achieving excess error τ which has sample complexity* $\mathrm{poly}(|\mathcal{S}|, 1/\tau)$.

Importantly, this algorithm is still successful with inconsistent labelled samples (examples), as long as they come from a distribution on $\mathcal{X} \times \{-1,1\}$, where the marginal distribution on $\mathcal{X}$ is uniform.

7.4 Finishing the proof of Proposition 7.3

Now we put all the pieces together. To agnostically learn a k-monotone function, we simply perform the agnostic learning algorithm of [34] on the distribution D over $[m]^d \times \{-1,1\}$ defined by

$$D(x,b) = \Pr_{y\in\mathcal{B}^{-1}(x)}[f(y) = b].$$

To generate a sample (x,b) from D, we draw a uniformly random string in $x \in [m]^d$, and b is the result of a query for the value of $f(y)$ for a uniformly random $y \in \mathcal{B}^{-1}(x)$. From Lemma 7.9, we can take $\mathcal{S}$ to be the set of $(k\sqrt{d}/\tau^2)$-way products of rd indicator functions. It follows that

$$|\mathcal{S}| = \binom{rd}{k\sqrt{d}/\tau^2} = \exp\left(\widetilde{O}\left(k\sqrt{d}/\tau^2\right)\right)$$

(this holds because $r = O(k/(\varepsilon_2 - 3\varepsilon_1))$).

Proposition 7.11. *Algorithm 2 accepts all functions ε_1-close to k-monotone functions, and rejects all functions ε_2-far from k-monotone, when $\varepsilon_2 > 3\varepsilon_1$ (with probability at least $2/3$). Its query complexity is*

$$\exp\left(\widetilde{O}\left(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2\right)\right).$$

Proof. By a union bound, we have that with probability at least $8/10$ both Step 5 and Step 4 succeed. Let us condition on this.

Completeness. Suppose f is ε_1-close to k-monotone. Lemma 7.1 and the triangle inequality imply that there is a k-monotone m-block function g^* such that $\operatorname{dist}(f,g^*) \le \varepsilon_1 + \alpha/6$. The agnostic learning algorithm thus returns a hypothesis h such that $\operatorname{dist}(f,h) \le \varepsilon_1 + \alpha/4$. The algorithm estimates this closeness to within $\alpha/7$, so the estimate obtained in Step 5 is at most $\varepsilon_1 + \alpha/4 + \alpha/7 < \varepsilon_1 + 5\alpha/12$ and the algorithm does not reject in this step. By the triangle inequality, h is $(2\varepsilon_1 + 5\alpha/12)$-close to k-monotone, and the algorithm will accept. There is no estimation error here, since no queries to f are required.

Soundness. Now suppose f is ε_2-far from k-monotone, where $\varepsilon_2 = 3\varepsilon_1 + \alpha$ for some $\alpha > 0$. Suppose the algorithm does not reject when estimating $\operatorname{dist}(f,h)$, where h is the hypothesis returned by the agnostic learning algorithm. Then $\operatorname{dist}(f,h) \le \varepsilon_1 + 5\alpha/12 + \alpha/7 < \varepsilon_1 + 7\alpha/12$. By the triangle inequality, if t is a k-monotone function, $\operatorname{dist}(h,t) \ge \operatorname{dist}(f,t) - \operatorname{dist}(f,h) > \varepsilon_2 - (\varepsilon_1 + 7\alpha/12) = 2\varepsilon_1 + 5\alpha/12$. The algorithm will thus reject in the final step.

Query complexity. The query complexity of the algorithm is dominated by the query complexity of the agnostic learning algorithm, which is

$$\exp\left(\widetilde{O}\left(k\sqrt{d}/\alpha^2\right)\right) = \exp\left(\widetilde{O}\left(k\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2\right)\right). \qquad \square$$

8 Tolerant testing and applications to L_1-testing

We now show how our techniques can be applied to make progress on an open problem on L_1 tolerant testing of monotonicity, asked at the Sublinear Algorithms Workshop 2016 [52]. In particular, as detailed in Theorem 1.9, we obtain tolerant ℓ_1 testers for monotonicity over the hypergrids with query complexity depending only on the parameters ε_1 and ε_2 for small (constant) d.

We start by describing a reduction lemma from L_1 distance to monotonicity of functions in $[0,1]^{\mathcal{X}}$ to Hamming distance to monotonicity of functions in $\{0,1\}^{\mathcal{X}\times[0,1]}$ (that is, "trading the range for a dimension"). We note that this idea appears in Berman *et al*. [8, Lemmata 2.1 and 2.3], although formulated in a slightly different way. For convenience and completeness, we state and prove here the version we shall use.

In what follows, we let $\mathcal{X}$ be a discrete partially ordered domain equipped with a measure μ,[12] that is a tuple $(\mathcal{X}, \preceq, \mu)$; and for a set $\mathcal{Y} \subseteq \mathbb{R}$ we denote by $\mathcal{M}^{(\mathcal{X}\to\mathcal{Y})} \subseteq \mathcal{Y}^{\mathcal{X}}$ the set of monotone functions from $\mathcal{X}$ to $\mathcal{Y}$.

Definition 8.1 (Analogue of [8, Definition 2.1])**.** For a function $f\colon \mathcal{X} \to [0,1]$, the *threshold function* $T \circ f\colon \mathcal{X} \times [0,1] \to \{0,1\}$ is defined by

$$T \circ f(x,t) = \mathbb{1}_{\{f(x)\ge 1-t\}} = \begin{cases} 1 & \text{if } f(x) \ge 1-t, \\ 0 & \text{otherwise.} \end{cases}$$

[12] We will only require that $(\mathcal{X},\mu)$ be a measurable space with finite measure, that is $\mu(\mathcal{X}) < \infty$, and shall henceforth only concern ourselves with *measurable* functions.

The next fact is immediate from this definition:

Fact 8.2. *For any $f\colon \mathcal{X} \to [0,1]$, it is the case that for every $x \in \mathcal{X}$*

$$f(x) = \int_0^1 T \circ f(x,t) dt\,.$$

Moreover, $f \in \mathcal{M}^{(\mathcal{X}\to[0,1])}$ if, and only if, $T \circ f \in \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}$.

We begin by the following characterization, which is immediately obtained from a corresponding theorem of Berman *et al.*; before stating a slightly modified version that we shall rely upon. For completeness, we give the proof of the former below.

Proposition 8.3 (Analogue of [8, Lemma 2.1])**.** *For any $f\colon \mathcal{X} \to [0,1]$,*

$$L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) = L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right) = \operatorname{dist}\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)\,.$$

(In particular, the RHS is well-defined, i. e., the mapping

$$t \in [0,1] \mapsto L_1\left(T\circ f(\cdot,t), \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)\,,$$

where $T\circ f(\cdot,t) \in \{0,1\}^{\mathcal{X}}$ is the function that maps x to $T\circ f(x,t)$, is measurable.)

Proof. We write $\nu \stackrel{\text{def}}{=} \mu \times \mathrm{Leb}_{[0,1]}$ for the product measure on $\mathcal{X}\times[0,1]$ induced by μ and the Lebesgue measure on $[0,1]$; so that $\nu(\mathcal{X}\times[0,1]) = \mu(\mathcal{X})\cdot 1 = \mu(\mathcal{X})$.

For any fixed $t\in[0,1]$, let $g_t \in \mathcal{M}^{(\mathcal{X}\to\{0,1\})}$ be any function with

$$L_1(T\circ f(\cdot,t), g_t) = L_1\left(T\circ f(\cdot,t), \mathcal{M}^{(\mathcal{X}\to\{0,1\})}\right)\,,$$

and define $g \in [0,1]^{\mathcal{X}}$ by

$$g'(x) = \int_0^1 dt g_t(x)$$

for all $x \in \mathcal{X}$: note that g is then monotone by construction.[13] Moreover, choose $h \in \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}$ as a function achieving

$$L_1(T\circ f, h) = L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)\,.$$

[13]Additionally, since we restrict ourselves to finite $\mathcal{X}$, there are only finitely many distinct functions $T\circ f(\cdot,t)$ (for $t\in[0,1]$, and therefore only finitely many distinct functions g_t).

Then we have

$$\begin{aligned}
L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) &\le L_1(f,g') = \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\left|\int_0^1 dt(T\circ f(x,t)-g_t(x))\right| \\
&\le \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\int_0^1 dt\,|T\circ f(x,t)-g_t(x)| \\
&= \int_0^1 dt\left(\frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\,|T\circ f(x,t)-g_t(x)|\right) = \int_0^1 dt L_1(T\circ f(\cdot,t),g_t) \\
&\le \int_0^1 dt L_1(T\circ f(\cdot,t),h(\cdot,t)) = \int_0^1 dt\left(\frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\,|T\circ f(x,t)-h(x,t)|\right) \\
&= \frac{1}{\nu(\mathcal{X}\times[0,1])}\int_{\mathcal{X}\times[0,1]}\nu(dx,dt)\,|T\circ f(x,t)-h(x,t)| \\
&= L_1(T\circ f,h) = L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)
\end{aligned}$$

where we applied Fact 8.2 (and the definition of $g' = \int_0^1 g_t$) for the first equality, and for the third inequality the fact that h induces (for every fixed $t\in[0,1]$) a monotone function $h(\cdot,t)\in\mathcal{M}^{(\mathcal{X}\to\{0,1\})}$: so that $L_1(T\circ f(\cdot,t),g_t)\le L_1(T\circ f(\cdot,t),h(\cdot,t))$ for all t.

For the other direction of the inequality, fix any $f\colon \mathcal{X}\to[0,1]$, and let $g\in\mathcal{M}^{(\mathcal{X}\to[0,1])}$ be (any) function achieving $L_1(f,g) = L_1\left(f,\mathcal{M}^{(\mathcal{X}\to[0,1])}\right)$. We can write, unrolling the definitions,

$$\begin{aligned}
L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) &= \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)|f(x)-g(x)| \\
&= \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\left|\int_0^1 dt(T\circ f(x,t)-T\circ g(x,t))\right| \\
&= \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\left|\int_0^1 dt(T\circ f(x,t)-T\circ g(x,t))\right| \\
&= \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\Big(\int_0^1 dt(T\circ f(x,t)-T\circ g(x,t))\mathbb{1}_{\{f(x)>g(x)\}} \\
&\qquad + (T\circ g(x,t)-T\circ f(x,t))\mathbb{1}_{\{g(x)>f(x)\}}\Big) \\
&= \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\int_0^1 dt\mu(dx)\Big((T\circ f(x,t)-T\circ g(x,t))\mathbb{1}_{\{f(x)>g(x)\}} \\
&\qquad + (T\circ g(x,t)-T\circ f(x,t))\mathbb{1}_{\{g(x)>f(x)\}}\Big) \\
&= \frac{1}{\nu(\mathcal{X}\times[0,1])}\int_{\mathcal{X}\times[0,1]}\nu(dx,dt)\,|T\circ f(x,t)-T\circ g(x,t)| = L_1(T\circ f,T\circ g) \\
&\ge L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)
\end{aligned}$$

where we applied Fact 8.2 for the second equality, the definition of L_1 distance for the second-to-last; and to handle the absolute values we used the fact that $|a-b| = (a-b)\mathbb{1}_{\{a>b\}} + (b-a)\mathbb{1}_{\{a>b\}}$, along with the observation that $T\circ f(x,t) > T\circ g(x,t)$ can only hold if $f(x) > g(x)$. Finally, we have $L_1(T\circ f, T\circ g) \geq L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)$ since $T\circ g \in \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}$, yielding the desired claim.

Finally, the fact that

$$L_1\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right) = \operatorname{dist}\left(T\circ f, \mathcal{M}^{(\mathcal{X}\times[0,1]\to\{0,1\})}\right)$$

is immediate from the Boolean range, as $|a-b| = \mathbb{1}_{\{a\neq b\}}$ for any $a.b \in \{0,1\}$. □

Proposition 8.4 (Rounding and Range-Dimension Tradeoff). *For any $f\colon \mathcal{X}\to[0,1]$ and parameter $m\geq 1$, let $R_m \stackrel{\text{def}}{=} \{1/m, 2/m, \dots, 1\}$. We define the m-rounding of f as $\Phi_m\circ f\colon \mathcal{X}\to R_m$ by*

$$\Phi_m\circ f(x) = \frac{\lceil mf(x)\rceil}{m}, \quad x\in\mathcal{X}.$$

Then we have

(i) $\left|L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) - L_1\left(\Phi_m\circ f, \mathcal{M}^{(\mathcal{X}\to R_m)}\right)\right| \leq \frac{1}{m}$;

(ii) $L_1\left(\Phi_m\circ f, \mathcal{M}^{(\mathcal{X}\to R_m)}\right) = \operatorname{dist}\left(T\circ\Phi_m\circ f, \mathcal{M}^{(\mathcal{X}\times R_m\to\{0,1\})}\right)$.

Proof of Proposition 8.4. Fix any $m\geq 1$. We start the proof of Proposition (i) by the simple observation that if $f\in\mathcal{M}^{(\mathcal{X}\to[0,1])}$, then $\Phi_m\circ f\in\mathcal{M}^{(\mathcal{X}\to R_m)}\subseteq\mathcal{M}^{(\mathcal{X}\to[0,1])}$, that is rounding preserves monotonicity; and that $\Phi_m\circ g = g$ for all $g\colon\mathcal{X}\to R_m$. This, along with the fact that for all $f\colon\mathcal{X}\to[0,1]$

$$L_1(f,\Phi_m\circ f) = \frac{1}{\mu(\mathcal{X})}\int_{\mathcal{X}}\mu(dx)\underbrace{|\Phi_m\circ f(x) - f(x)|}_{\leq 1/m} \leq \frac{1}{m}$$

implies by the triangle inequality, for any $g\in\mathcal{M}^{(\mathcal{X}\to R_m)}\subseteq\mathcal{M}^{(\mathcal{X}\to[0,1])}$, that

$$L_1\left(f,\mathcal{M}^{(\mathcal{X}\to[0,1])}\right) \leq L_1(f,g) \leq \frac{1}{m} + L_1(g,\Phi_m\circ g) + L_1(\Phi_m\circ f,\Phi_m\circ g) = \frac{1}{m} + 0 + L_1(\Phi_m\circ f, g)\ .$$

Taking $g\in\mathcal{M}^{(\mathcal{X}\to R_m)}$ that achieves $L_1(\Phi_m\circ f, g) = L_1\left(\Phi_m\circ f,\mathcal{M}^{(\mathcal{X}\to R_m)}\right)$, we get

$$L_1\left(f,\mathcal{M}^{(\mathcal{X}\to[0,1])}\right) \leq \frac{1}{m} + L_1\left(\Phi_m\circ f,\mathcal{M}^{(\mathcal{X}\to R_m)}\right)\ .$$

For the other direction, we first note that for any two functions $f,g\colon\mathcal{X}\to[0,1]$, it is the case that $L_1(f,g)\geq L_1(\Phi_m\circ f,\Phi_m\circ g) - 1/m$ (which is immediate from the definition of the rounding operator), and taking g to be the closest monotone function to f this readily yields

$$L_1\left(f,\mathcal{M}^{(\mathcal{X}\to[0,1])}\right) \geq L_1(\Phi_m\circ f,\Phi_m\circ g) - \frac{1}{m} \geq L_1\left(\Phi_m\circ f,\mathcal{M}^{(\mathcal{X}\to R_m)}\right) - \frac{1}{m}\,.$$

Finally, the proof of the second part, Proposition (ii), is identical to that of Proposition 8.3, replacing the Lebesgue measure on $[0,1]$ by the counting measure on R_m. (So that integrals over $[0,1]$ become sums over R_m, normalized by $|R_m| = m$.) □

Given Proposition 8.4, it is now easy to apply the results of Section 7 to obtain a tolerant L_1 tester for monotonicity of functions $f\colon [n]^d \to [0,1]$. Indeed, given parameters $0 < \varepsilon_1 < \varepsilon_2$, one can set the rounding parameter m to $\lceil 4/(\varepsilon_2 - \varepsilon_1)\rceil$; and from query access to $f\colon [n]^d \to [0,1]$, simulate query access to $\Phi_m \circ f$ and therefore to $g \overset{\text{def}}{=} T \circ \Phi_m \circ f\colon [n]^d \times R_m \to \{0,1\}$. By Proposition 8.4 and our choice of m, in order to distinguish

$$L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) \le \varepsilon_1 \quad \text{vs.} \quad L_1\left(f, \mathcal{M}^{(\mathcal{X}\to[0,1])}\right) \ge \varepsilon_2$$

it is enough to distinguish

$$\operatorname{dist}\left(g, \mathcal{M}^{(\mathcal{X}\times R_m\to\{0,1\})}\right) \le \varepsilon_1 + \frac{1}{m} \quad \text{vs.} \quad \operatorname{dist}\left(g, \mathcal{M}^{(\mathcal{X}\times R_m\to\{0,1\})}\right) \ge \varepsilon_2 - \frac{1}{m}.$$

By our choice of m, we also have

$$\left(\varepsilon_2 - \frac{1}{m}\right) - \left(\varepsilon_1 + \frac{1}{m}\right) \ge \frac{\varepsilon_2 - \varepsilon_1}{2}.$$

The last step is to observe that one can view equivalently g as a function $g\colon [n]^d \times [m] \to \{0,1\}$; by Proposition 8.4 and our choice of m, so that the algorithms of Corollary 1.7 apply.

Theorem 1.9. *There exists a non-adaptive tolerant L_1-tester for monotonicity of functions $f\colon [n]^d \to [0,1]$ with query complexity*

- $\widetilde{O}\left(\frac{1}{(\varepsilon_2-\varepsilon_1)^2}\left(\frac{5d}{\varepsilon_2-\varepsilon_1}\right)^d\right)$, *for any* $0 \le \varepsilon_1 < \varepsilon_2 \le 1$;
- $\exp\left(\widetilde{O}\left(\sqrt{d}/(\varepsilon_2 - 3\varepsilon_1)^2\right)\right)$, *for any* $0 \le 3\varepsilon_1 < \varepsilon_2 \le 1$.

Acknowledgments. We would like to thank Eric Blais for helpful remarks on an earlier version of this paper, and the anonymous reviewers for very detailed and insightful comments.

References

[1] NIR AILON AND BERNARD CHAZELLE: Information theory in property testing and monotonicity testing in higher dimension. *Inform. and Comput.*, 204(11):1704–1717, 2006. Preliminary version in STACS'05. [doi:10.1016/j.ic.2006.06.001] 4

[2] NIR AILON, BERNARD CHAZELLE, SESHADHRI COMANDUR, AND DING LIU: Estimating the distance to a monotone function. *Random Structures Algorithms*, 31(3):371–383, 2007. Preliminary version in RANDOM'04. [doi:10.1002/rsa.20167] 14

[3] KAZUYUKI AMANO AND AKIRA MARUOKA: A superpolynomial lower bound for a circuit computing the Clique function with at most $(1/6)\log\log n$ negation gates. *SIAM J. Comput.*, 35(1):201–216, 2005. Preliminary version in MFCS'98. [doi:10.1137/S0097539701396959] 9

[4] DANA ANGLUIN: Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988. [doi:10.1007/BF00116828] 2

[5] MARIA-FLORINA BALCAN, ERIC BLAIS, AVRIM BLUM, AND LIU YANG: Active property testing. In *Proc. 53rd FOCS*, pp. 21–30. IEEE Computer Society, 2012. [doi:10.1109/FOCS.2012.64] 4

[6] TUĞKAN BATU, RONITT RUBINFELD, AND PATRICK WHITE: Fast approximate PCPs for multidimensional bin-packing problems. *Inform. and Comput.*, 196(1):42–56, 2005. Preliminary version in RANDOM'99. [doi:10.1016/j.ic.2004.10.001] 4

[7] ALEKSANDRS BELOVS AND ERIC BLAIS: A polynomial lower bound for testing monotonicity. In *Proc. 48th STOC*, pp. 1021–1032. ACM, 2016. [doi:10.1145/2897518.2897567, arXiv:1511.05053] 2, 3, 10, 18

[8] PIOTR BERMAN, SOFYA RASKHODNIKOVA, AND GRIGORY YAROSLAVTSEV: L_p-testing. In *Proc. 46th STOC*, pp. 164–173. ACM, 2014. [doi:10.1145/2591796.2591887] 4, 6, 8, 10, 11, 32, 35, 36, 37, 38, 45, 46, 53

[9] ARNAB BHATTACHARYYA, ELENA GRIGORESCU, KYOMIN JUNG, SOFYA RASKHODNIKOVA, AND DAVID P. WOODRUFF: Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012. Preliminary version in SODA'09. [doi:10.1137/110826655, arXiv:0808.1787] 9

[10] ERIC BLAIS, JOSHUA BRODY, AND KEVIN MATULEF: Property testing lower bounds via communication complexity. *Comput. Complexity*, 21(2):311–358, 2012. Preliminary version in CCC'11. [doi:10.1007/s00037-012-0040-x] 4

[11] ERIC BLAIS, CLÉMENT L. CANONNE, IGOR C. OLIVEIRA, ROCCO A. SERVEDIO, AND LI-YANG TAN: Learning circuits with few negations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (RANDOM 2015)*, volume 40, pp. 512–527. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.APPROX-RANDOM.2015.512, arXiv:1410.8420] 2, 3, 8, 9, 42

[12] 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. Preliminary version in RANDOM'10. [doi:10.1007/s00493-012-2765-1] 2, 3

[13] NADER H. BSHOUTY AND CHRISTINO TAMON: On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996. Preliminary version in STOC'95. [doi:10.1145/234533.234564] 2

[14] CLÉMENT L. CANONNE, ELENA GRIGORESCU, SIYAO GUO, AKASH KUMAR, AND KARL WIMMER: Testing k-monotonicity. In *Proc. 8th Symp. Innovations in Theoretical Computer Science (ITCS'17)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.ITCS.2017.29, arXiv:1609.00265] 1

[15] CLÉMENT L. CANONNE AND RONITT RUBINFELD: Testing probability distributions underlying aggregated data. In *Proc. 41st Internat. Colloq. on Automata, Languages and Programming*

(ICALP'14), volume 8572 of *Lecture Notes in Computer Science*, pp. 283–295. Springer, 2014. [doi:10.1007/978-3-662-43948-7_24, arXiv:1402.3835, ECCC:TR14-021] 7, 27, 28, 29

[16] DEEPARNAB CHAKRABARTY AND C. SESHADHRI: Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proc. 45th STOC*, pp. 419–428, 2013. [doi:10.1145/2488608.2488661, arXiv:1204.0849] 2, 4, 11

[17] DEEPARNAB CHAKRABARTY AND C. SESHADHRI: An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10(17):453–464, 2014. Preliminary version in RANDOM'13. [doi:10.4086/toc.2014.v010a017] 2, 4, 11

[18] DEEPARNAB CHAKRABARTY AND C. SESHADHRI: An $o(n)$ monotonicity tester for Boolean functions over the hypercube. *SIAM J. Comput.*, 45(2):461–472, 2016. Preliminary version in STOC'13. [doi:10.1137/13092770X, arXiv:1302.4536] 2, 3, 4, 10

[19] XI CHEN, ANINDYA DE, ROCCO A. SERVEDIO, AND LI-YANG TAN: Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proc. 47th STOC*, pp. 519–528. ACM, 2015. [doi:10.1145/2746539.2746570, arXiv:1412.5657] 2, 3, 10, 18, 22

[20] XI CHEN, ROCCO A. SERVEDIO, AND LI-YANG TAN: New algorithms and lower bounds for monotonicity testing. In *Proc. 55th FOCS*, pp. 286–295. IEEE Computer Society, 2014. [doi:10.1109/FOCS.2014.38, arXiv:1412.5655] 3, 10

[21] XI CHEN, ERIK WAINGARTEN, AND JINYU XIE: Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proc. 49th STOC*, pp. 523–536, 2017. [doi:10.1145/3055399.3055461, arXiv:1702.06997] 3, 10, 22

[22] YEVGENIY DODIS, ODED GOLDREICH, ERIC LEHMAN, SOFYA RASKHODNIKOVA, DANA RON, AND ALEX SAMORODNITSKY: Improved testing algorithms for monotonicity. In *Proc. 3rd Internat. Workshop on Randomization and Computation (RANDOM'99)*, volume 1671 of *Lecture Notes in Computer Science*, pp. 97–108. Springer, 1999. [doi:10.1007/978-3-540-48413-4_10] 2, 3, 7

[23] FUNDA ERGÜN, SAMPATH KANNAN, RAVI KUMAR, RONITT RUBINFELD, AND MAHESH VISWANATHAN: Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000. Preliminary version in STOC'98. [doi:10.1006/jcss.1999.1692] 4, 14

[24] SHAHAR FATTAL AND DANA RON: Approximating the distance to monotonicity in high dimensions. *ACM Trans. Algorithms*, 6(3):52:1–37, 2010. [doi:10.1145/1798596.1798605] 6, 7, 11

[25] ELDAR FISCHER: On the strength of comparisons in property testing. *Inform. and Comput.*, 189(1):107–116, 2004. [doi:10.1016/j.ic.2003.09.003] 4

[26] ELDAR FISCHER, ERIC LEHMAN, ILAN NEWMAN, SOFYA RASKHODNIKOVA, RONITT RUBINFELD, AND ALEX SAMORODNITSKY: Monotonicity testing over general poset domains. In *Proc. 34th STOC*, pp. 474–483. ACM, 2002. [doi:10.1145/509907.509977] 2, 3, 9, 10, 14, 15, 18, 19, 21

[27] PETER GEMMELL, RICHARD J. LIPTON, RONITT RUBINFELD, MADHU SUDAN, AND AVI WIGDERSON: Self-testing/correcting for polynomials and for approximate functions. In *Proc. 23rd STOC*, pp. 32–42. ACM, 1991. [doi:10.1145/103418.103429] 12

[28] ODED GOLDREICH, SHAFI GOLDWASSER, ERIC LEHMAN, DANA RON, AND ALEX SAMORODNITSKY: Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. Preliminary version in FOCS'98. [doi:10.1007/s004930070011] 2, 3, 4, 10, 22

[29] ELENA GRIGORESCU, AKASH KUMAR, AND KARL WIMMER: Flipping out with many flips: Hardness of testing k-monotonicity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (RANDOM 2018)*, pp. 40:1–17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.APPROX-RANDOM.2018.40] 8

[30] SIYAO GUO AND ILAN KOMARGODSKI: Negation-limited formulas. *Theoret. Comput. Sci.*, 660:75–85, 2017. Preliminary version in RANDOM'15. [doi:10.1016/j.tcs.2016.11.027] 2, 9

[31] SIYAO GUO, TAL MALKIN, IGOR C. OLIVEIRA, AND ALON ROSEN: The power of negations in cryptography. In *12th Theory of Cryptography Conf. (TCC'15)*, volume 9014 of *Lecture Notes in Computer Science*, pp. 36–65. Springer, 2015. [doi:10.1007/978-3-662-46494-6_3] 2, 9

[32] SHIRLEY HALEVY AND EYAL KUSHILEVITZ: Testing monotonicity over graph products. *Random Structures Algorithms*, 33(1):44–67, 2008. Preliminary version in ICALP'04. [doi:10.1002/rsa.20211] 4, 14

[33] STASYS JUKNA: *Boolean Function Complexity*. Springer, 2012. [doi:10.1007/978-3-642-24508-4] 2, 9

[34] ADAM TAUMAN KALAI, ADAM R. KLIVANS, YISHAY MANSOUR, AND ROCCO A. SERVEDIO: Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. Preliminary version in FOCS'05. [doi:10.1137/060649057] 8, 42, 44

[35] MICHAEL J. KEARNS AND DANA RON: Testing problems with sublearning sample complexity. *J. Comput. System Sci.*, 61(3):428–456, 2000. Preliminary version in COLT'98. [doi:10.1006/jcss.1999.1656] 4

[36] MICHAEL J. KEARNS AND LESLIE G. VALIANT: Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994. Preliminary version in STOC'89. [doi:10.1145/174644.174647] 2

[37] SUBHASH KHOT, DOR MINZER, AND MULI SAFRA: On monotonicity testing and Boolean isoperimetric type theorems. In *Proc. 56th FOCS*, pp. 52–58. IEEE Computer Society, 2015. [doi:10.1109/FOCS.2015.13] 2, 3, 10, 18

[38] PRAVESH KOTHARI, AMIR NAYYERI, RYAN O'DONNELL, AND CHENGGANG WU: Testing surface area. In *Proc. 25th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'14)*, pp. 1204–1214. SIAM, 2014. [doi:10.1137/1.9781611973402.89] 4

[39] CHENGYU LIN AND SHENGYU ZHANG: Sensitivity conjecture and log-rank conjecture for functions with small alternating numbers. In *Proc. 44th Internat. Colloq. on Automata, Languages and Programming (ICALP'17)*, pp. 51:1–13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. [doi:10.4230/LIPIcs.ICALP.2017.51, arXiv:1602.06627] 2, 9

[40] ANDREI A. MARKOV: On the inversion complexity of systems of functions. *Doklady Akademii Nauk SSSR, N.S.*, 116:917–919, 1957. English translation in [41]. 2

[41] ANDREI A. MARKOV: On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958. [doi:10.1145/320941.320945] 53

[42] JOE NEEMAN: Testing surface area with arbitrary accuracy. In *Proc. 46th STOC*, pp. 393–397. ACM, 2014. [doi:10.1145/2591796.2591807] 4

[43] RYAN O'DONNELL: *Analysis of Boolean Functions*. Cambridge University Press, 2014. [doi:10.1017/CBO9781139814782] 43

[44] RYAN O'DONNELL AND ROCCO A. SERVEDIO: Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007. Preliminary version in CCC'06. [doi:10.1137/060669309] 2

[45] RYAN O'DONNELL AND KARL WIMMER: KKL, Kruskal-Katona, and monotone nets. *SIAM J. Comput.*, 42(6):2375–2399, 2013. Preliminary version in FOCS'09. [doi:10.1137/100787325] 2

[46] MICHAL PARNAS, DANA RON, AND RONITT RUBINFELD: Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 72(6):1012–1042, 2006. [doi:10.1016/j.jcss.2006.03.002] 5, 14

[47] RAN RAZ AND AVI WIGDERSON: Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992. Preliminary version in STOC'90. [doi:10.1145/146637.146684] 2

[48] ALEXANDER A. RAZBOROV: Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR, N.S.*, 281(4):798–801, 1985. English translation in Soviet Math. Doklady, 31:354–357, 1985. 2

[49] BENJAMIN ROSSMAN: Correlation bounds against monotone NC^1. In *Proc. 30th IEEE Conf. on Computational Complexity (CCC'15)*, pp. 392–411. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.CCC.2015.392] 2, 9

[50] ROCCO A. SERVEDIO: On learning monotone DNF under product distributions. *Inform. and Comput.*, 193(1):57–74, 2004. Preliminary version in COLT'01. [doi:10.1016/j.ic.2004.04.003] 2

[51] LESLIE G. VALIANT: A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. Preliminary version in STOC'84. [doi:10.1145/1968.1972] 13

[52] GRIGORY YAROSLAVTSEV: List of open problems in sublinear algorithms: Problem 70. `http://sublinear.info/70`, 2016. Originally posed in [8]. 6, 8, 45

AUTHORS

Clément L. Canonne
Graduate student
Columbia University, New York, NY
ccanonne@cs.columbia.edu
http://www.cs.columbia.edu/~ccanonne/

Elena Grigorescu
Assistant professor
Purdue University
elena-g@purdue.edu
https://www.cs.purdue.edu/homes/egrigore/

Siyao Guo
Assistant professor
NYU Shanghai
siyao.guo@nyu.edu
https://sites.google.com/site/siyaoguo/

Akash Kumar
Graduate student
Purdue University
akumar@purdue.edu
https://www.cs.purdue.edu/homes/akumar/

Karl Wimmer
Associate professor
Duquesne University
wimmerk@duq.edu
http://www.mathcs.duq.edu/~wimmer/

ABOUT THE AUTHORS

CLÉMENT L. CANONNE is currently a Motwani Postdoctoral Fellow at Stanford University. He graduated from Columbia University in 2017, where his advisor was Rocco Servedio. His research focuses on the fields of property testing and sublinear algorithms; specifically, on understanding the strengths and limitations of the standard models in property and distribution testing, as well as in related areas. He also really likes elephants.

ELENA GRIGORESCU is an Assistant Professor in the Computer Science Department of Purdue University. She graduated from EECS MIT in 2010; her advisor was Madhu Sudan. Her research interests span many aspects of Theoretical Computer Science, with a focus on sublinear algorithms, coding theory, and complexity theory. She likes elephants too, but prefers Boo.

SIYAO GUO is an Assistant Professor in the Computer Science Department of NYU Shanghai. She graduated from the Chinese University of Hong Kong in 2014 where she was advised by Andrej Bogdanov. Her research interests include pseudorandomness, complexity and cryptography. She loves her softball teams BNU Phoenix and CUHK Phoenix.

AKASH KUMAR is a Ph. D. student at Purdue University. He is interested in Property Testing and Hardness of Approximation. He likes to imagine that he is good at the game of cricket. But really, he is a terrible player.

KARL WIMMER is an Associate Professor at Duquesne University. He graduated from CMU in 2009; he had the privilege of being the first Ph. D. graduate advised by Ryan O'Donnell. His research interest is, like, math and stuff, with a current focus on property testing and Fourier analysis. He enjoys spending time with his wonderful wife Cassondra and their three children. His family also greatly enjoys pets, but they have not yet taken in any elephants.