THEORY OF COMPUTING, Volume 10 (17), 2014, pp. 453–464
www.theoryofcomputing.org

An Optimal Lower Bound for Monotonicity Testing over Hypergrids

Deeparnab Chakrabarty C. Seshadhri*

Received April 2, 2014; Revised December 3, 2014; Published December 24, 2014

Abstract: For positive integers n, d, the hypergrid $[n]^d$ is equipped with the coordinatewise product partial ordering denoted by $\prec$. A function $f : [n]^d \to \mathbb{N}$ is monotone if $\forall x \prec y$, $f(x) \leq f(y)$. A function f is ε-far from monotone if at least an ε fraction of values must be changed to make f monotone. Given a parameter ε, a *monotonicity tester* must distinguish with high probability a monotone function from one that is ε-far.

We prove that any (adaptive, two-sided) monotonicity tester for functions $f : [n]^d \to \mathbb{N}$ must make $\Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ queries. Recent upper bounds show the existence of $O(\varepsilon^{-1} d \log n)$ query monotonicity testers for hypergrids. This closes the question of monotonicity testing for hypergrids over arbitrary ranges. The previous best lower bound for general hypergrids was a non-adaptive bound of $\Omega(d \log n)$.

ACM Classification: F.1.2, F.2.2

AMS Classification: 68Q17, 68W20

Key words and phrases: lower bounds, property testing, monotonicity testing

1 Introduction

Given query access to a function f, the area of *property testing* [21, 17] deals with the problem of determining properties of f without accessing all its inputs. Monotonicity testing [16] is a classic problem

A conference version of this paper appeared in the Proceedings of the 17th Internat. Workshop on Randomization and Computation (RANDOM 2013) [11].

*This work was funded by the DARPA Graph-theoretic Research in Algorithms and the Phenomenology of Social Networks (GRAPHS) program and by the DOE ASCR Complex Interconnected Distributed Systems (CIDS) program, and Sandia's Laboratory Directed Research & Development (LDRD) program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

DOI: 10.4086/toc.2014.v010a017

in property testing. Consider a function $f : \mathbf{D} \to \mathbf{R}$, where $\mathbf{D}$ is a finite set equipped with a partial order given by "$\prec$," and $\mathbf{R}$ is a set equipped with a total order. The function f is monotone if for all $x \prec y$ (in $\mathbf{D}$), $f(x) \le f(y)$. The *distance to monotonicity* of f is the minimum fraction of values that need to be modified to make f monotone. More precisely, let the distance between functions $d(f,g)$ be $|\{x : f(x) \neq g(x)\}|/|\mathbf{D}|$, and let $\mathcal{M}$ be the set of all monotone functions. Then the distance to monotonicity of f is $\min_{g\in\mathcal{M}} d(f,g)$. (This minimum always exists since $\mathbf{D}$ is finite.)

A function is called ε-far from monotone if the distance to monotonicity is strictly greater than ε. A *property tester for monotonicity* is a, possibly randomized, algorithm that takes as input a distance parameter $\varepsilon \in (0,1)$, error parameter $\delta \in [0,1]$, and query access to an arbitrary f. If f is monotone, then the tester must accept with probability $> 1-\delta$. If it is ε-far from monotone, then the tester rejects with probability $> 1-\delta$. If neither, then the tester is allowed to do anything. The aim is to design a property tester making as few queries as possible to the function. A tester is called *one-sided* if it always accepts a monotone function. A tester is called *non-adaptive* if the queries made do not depend on function values returned in the previous queries. The most general tester is an adaptive, two-sided tester.

Monotonicity testing has a rich history and the hypergrid domain, $[n]^d$, has received special attention. The boolean hypercube ($n = 2$) and the total order ($d = 1$) are special instances of hypergrids. Following a long line of work [13, 16, 12, 19, 15, 1, 14, 18, 20, 2, 3, 4], previous work of the authors [10] shows the existence of $O(\varepsilon^{-1} d \log n)$-query monotonicity testers. The result in this paper is a matching lower bound that is optimal in all parameters for functions of unbounded range.

Theorem 1.1. *Any adaptive, two-sided monotonicity tester for functions $f : [n]^d \to \mathbb{N}$ requires*

$$\Omega\left(\frac{d\log n - \log \varepsilon^{-1}}{\varepsilon}\right)$$

queries, assuming $\varepsilon > n^{-d}$.

1.1 Previous work

The problem of monotonicity testing was introduced by Goldreich et al. [16], who demonstrated a $O(n/\varepsilon)$ tester for functions $f : \{0,1\}^n \to \{0,1\}$. The first tester for general hypergrids was given by Dodis et al. [12]. The upper bound of $O(\varepsilon^{-1} d \log n)$ for monotonicity testing was recently proven in [10]. We refer the interested reader to the introduction of [10] for a more detailed history of previous upper bounds.

There have been numerous lower bounds for monotonicity testing. Following the work of Ergun et al. [13] who demonstrated an $\Omega(\log n)$ lower bound for *non-adaptive* monotonicity testers, for the total order $\mathbf{D} = [n]$, Fischer [14] gave an $\Omega(\log n)$ lower bound for adaptive monotonicity testers as well over $[n]$. For the hypercube domain, Fischer et al. [15] proved a $\Omega(\sqrt{d})$ lower bound for non-adaptive, one-sided testers (this lower bound holds even for $\{0,1\}$-ranged functions), which was improved to a $\Omega(d/\varepsilon)$ lower bound by Brïet et al. [8]. Using an ingenious reduction from communication complexity, Blais, Brody and Matulef [4] proved an $\Omega(d)$ lower bound for adaptive, two sided testers. Honing this reduction, Brody [9] improved it to an $\Omega(d/\varepsilon)$ lower bound. For the hypergrid domain, the only lower bound known was an $\Omega(d \log n)$ for non-adaptive testers by Blais, Raskhodnikova, and Yaroslavtsev [5] using communication complexity techniques.

We note that our theorem only holds when the range is $\mathbb{N}$, while some previous results hold for restricted ranges. The results of [4, 9] provide lower bounds for range $[\sqrt{d}]$ and that of Blais et al. [5] hold for the range $[nd]$. For these settings, the communication complexity reductions provide stronger lower bounds than our result.

1.2 Preliminaries and main ideas

We start with a formal definition of a tester. Consider the family of functions $f : \mathbf{D} \to \mathbf{R}$, where $\mathbf{D}$ is some partial order, and $\mathbf{R} \subseteq \mathbb{N}$. We assume that f always takes distinct values, so $\forall x, y, f(x) \neq f(y)$. Since we are proving lower bounds, this is no loss of generality.

Definition 1.2. An algorithm $\mathcal{A}$ is a (t, ε, δ)-*monotonicity tester* if $\mathcal{A}$ has the following properties. For any $f : \mathbf{D} \to \mathbf{R}$, the algorithm $\mathcal{A}$ makes t (possibly randomized) queries to f and then outputs either "accept" or "reject." If f is monotone, then $\mathcal{A}$ accepts with probability $> 1 - \delta$. If f is ε-far from monotone, then $\mathcal{A}$ rejects with probability $> 1 - \delta$.

Given a positive integer s, let $\mathbf{D}^s$ be the s-fold Cartesian product of $\mathbf{D}$. We define two symbols `acc` and `rej`, and denote $\mathbf{D}' = \mathbf{D} \cup \{\texttt{acc}, \texttt{rej}\}$. Any (t, ε, δ)-tester can be completely specified by the following family of functions. For all $s \leq t$, $\mathbf{x} \in \mathbf{D}^s$, $y \in \mathbf{D}'$, we consider a function $p^y_{\mathbf{x}} : \mathbf{R}^s \to [0, 1]$, with the semantics that for any $\mathbf{a} \in \mathbf{R}^s$, $p^y_{\mathbf{x}}(\mathbf{a})$ denotes the probability the tester queries y as the $(s+1)$th query, given that the first s queries are $\mathbf{x}_1, \ldots, \mathbf{x}_s$ and $f(\mathbf{x}_i) = \mathbf{a}_i$ for $1 \leq i \leq s$. These functions satisfy the following properties.

$$\forall s \leq t,\ \forall \mathbf{x} \in \mathbf{D}^s,\ \forall \mathbf{a} \in \mathbf{R}^s, \quad \sum_{y \in \mathbf{D}'} p^y_{\mathbf{x}}(\mathbf{a}) = 1\,, \tag{1.1}$$

$$\forall \mathbf{x} \in \mathbf{D}^t,\ \forall y \in \mathbf{D},\ \forall \mathbf{a} \in \mathbf{R}^t, \quad p^y_{\mathbf{x}}(\mathbf{a}) = 0\,. \tag{1.2}$$

(1.1) ensures the decisions of the tester at step $(s+1)$ must form a probability distribution. (1.2) implies that the tester makes at most t queries. Any adaptive tester can be specified by these functions. The important point to note is that these are finitely many functions; their number is at most $t|\mathbf{D}|^{t+1}$.

The starting point of this work is the result of Fischer [14] who proved an adaptive lower bound for monotonicity testing for functions $f : [n] \to \mathbb{N}$. He shows that adaptive testers can be reduced to what we call *comparison-based testers* ([14] calls them order-based testers). In plain English, comparison-based testers are adaptive testers whose decision on where to query at time $s+1$ depends only on the *order* of the function values at the s-query points so far, and not on the value themselves. Such a reduction is done using Ramsey theory arguments, in turn inspired by the work of Breslauer et al. [7]. Our starting point is an observation that Fischer's proof goes through for *every* partial order, and not just the total order $[n]$. To define comparison-based testers formally, we need some notation.

For any positive integer s, let $\mathbf{R}^{(s)}$ denote the set of *unordered* subets of $\mathbf{R}$ of cardinality s. We introduce new functions as follows. With each s, $\mathbf{x} \in \mathbf{D}^s$, $y \in \mathbf{D}'$, and *each permutation* $\sigma : [s] \to [s]$, we associate functions $q^y_{\mathbf{x},\sigma} : \mathbf{R}^{(s)} \to [0, 1]$, with the semantics

$$\text{For any set } S = (a_1 < a_2 < \cdots < a_s) \in \mathbf{R}^{(s)}, \qquad q^y_{\mathbf{x},\sigma}(S) := p^y_{\mathbf{x}}(a_{\sigma(1)}, \ldots, a_{\sigma(s)})\,.$$

That is, $q^y_{\mathbf{x},\sigma_s}(S)$ sorts the answers in S in increasing order, permutes them according to σ, and passes the permuted ordered tuple to $p^y_{\mathbf{x}}$. These q-functions allow us to formally define comparison-based testers.

Definition 1.3. A monotonicity tester $\mathcal{A}$ is *comparison-based* for functions $f : \mathbf{D} \to \mathbf{R}$ if for all s, $\mathbf{x} \in \mathbf{D}^s, y \in \mathbf{D}'$, and permutations $\sigma : [s] \to [s]$, the function $q^y_{\mathbf{x},\sigma}$ is a constant function on $\mathbf{R}^{(s)}$. In other words, the $(s+1)$th decision of the tester given that the first s questions is $\mathbf{x}$, depends only on the *ordering* of the answers received, and not on the values of the answers.

It is not too hard to see that a comparison-based tester for the domain $[n]$ can be easily converted to a non-adaptive tester, for which an $\Omega(\log n)$ bound was previously known [13]. This is not true for the hypergrid domain in general. To circumvent this, we first focus on the hypercube domain. As is standard, we define a distribution over functions, one of which is monotone and the others ε-far from monotone, and show that any *deterministic* comparison-based tester making few queries cannot be correct most of the time. Our monotone function is in fact the "decimal notation" of the binary vector which "mimics" a total order from 0 to $2^d - 1$. This can now be used to argue that any comparison-based tester is essentially non-adaptive for which a lower bound follows easily. Finally, for hypergrids, we give an easy reduction to hypercubes.

2 The reduction to comparison-based testers

Theorem 2.1. *Suppose there exists a (t, ε, δ)-monotonicity tester for functions $f : \mathbf{D} \to \mathbb{N}$. Then there exists a comparison-based $(t, \varepsilon, 2\delta)$-monotonicity tester for functions $f : \mathbf{D} \to \mathbb{N}$.*

As stated in the previous section, the above theorem is implicit in the work of Fischer [14] who proved it only for $\mathbf{D} = [n]$. We provide a proof for completeness. Call a monotonicity tester *discrete* if the corresponding functions $p^y_{\mathbf{x}}$ satisfying constraints (1.1), (1.2) can only take values in $\{i/K \,:\, 0 \le i \le K\}$ for some finite K.

Lemma 2.2. *Suppose there exists a (t, ε, δ)-monotonicity tester $\mathcal{A}$ for functions $f : \mathbf{D} \to \mathbb{N}$. Then there exists a discrete $(t, \varepsilon, 2\delta)$-monotonicity tester for such functions.*

Proof. We do a rounding on the p-functions. Let $K = 100t|\mathbf{D}|^t/\delta^2$. Start with the p-functions of the (t, ε, δ)-tester $\mathcal{A}$. For $y \in \mathbf{D} \cup \texttt{acc}$, $\mathbf{x} \in \mathbf{D}^s$, $\mathbf{a} \in \mathbf{R}^s$, let $\hat{p}^y_{\mathbf{x}}(\mathbf{a})$ be the largest value in $\{i/K \mid 0 \le i \le K\}$ which is at most $p^y_{\mathbf{x}}(\mathbf{a})$. Set $\hat{p}^{\texttt{rej}}_{\mathbf{x}}(\mathbf{a})$ so that (1.1) is maintained.

Note that for $y \in \mathbf{D} \cup \texttt{acc}$, if $p^y_{\mathbf{x}}(\mathbf{a}) > 10t/(\delta K)$, then

$$\left(1 - \frac{\delta}{10t}\right) p^y_{\mathbf{x}}(\mathbf{a}) \le \hat{p}^y_{\mathbf{x}}(\mathbf{a}) \le p^y_{\mathbf{x}}(\mathbf{a}).$$

Furthermore, $\hat{p}^{\texttt{rej}}_{\mathbf{x}}(\mathbf{a}) \ge p^{\texttt{rej}}_{\mathbf{x}}(\mathbf{a})$.

The $\hat{p}$-functions describe a new discrete tester $\mathcal{A}'$ that makes at most t queries. We argue that $\mathcal{A}'$ is a $(t, \varepsilon, 2\delta)$-tester. Given a function f that is either monotone or ε-far from monotone, consider a sequence of queries $\mathbf{x} = (x_1, \ldots, x_s)$ after which $\mathcal{A}$ returns a *correct* decision μ. Call such a sequence good, and let

$p(\mathbf{x})$ denote the probability this occurs. We know that the sum of $p(\mathbf{x})$ over all good query sequences is at least $(1-\delta)$. Now,

$$p(\mathbf{x}) := p^{x_1} \cdot p^{x_2}_{x_1}(f(x_1)) \cdot p^{x_3}_{(x_1,x_2)}(f(x_1), f(x_2)) \cdots p^{\mu}_{(x_1,\ldots,x_s)}(f(x_1),\ldots,f(x_s)).$$

Here p^{x_1} is the probability that the first point queried is x_1. Two cases arise. Suppose all of the multiplier probabilities in the right-hand side above are $\geq 10t/\delta K$. Then, the probability of this good sequence arising in $\mathcal{A}'$ is at least $(1-\delta/10t)^t p(\mathbf{x}) \geq p(\mathbf{x})(1-\delta/10)$. Otherwise, suppose some probability in the right-hand side is $< 10t/\delta K$; call such good sequences deficient. The total probability mass of querying deficient good sequences is at most $10t/\delta K \cdot |\mathbf{D}|^t \leq \delta/2$. Therefore, the probability of querying a good sequence in $\mathcal{A}'$ is at least $(1-3\delta/2)(1-\delta/10) > 1-2\delta$, where the first term is the mass on non-deficient, good sequences for $\mathcal{A}$. Therefore, $\mathcal{A}'$ is a $(t,\varepsilon,2\delta)$ tester. □

We introduce some Ramsey theory terminology. For any positive integer i, a *finite* coloring of $\mathbb{N}^{(i)}$ is a function $\texttt{col}_i : \mathbb{N}^{(i)} \to \{1,\ldots,C\}$ for some finite number C. An infinite set $X \subseteq \mathbb{N}$ is called *monochromatic* with respect to $\texttt{col}_i$ if for all sets $A, B \in X^{(i)}$, $\texttt{col}_i(A) = \texttt{col}_i(B)$. A *k-wise* finite coloring of $\mathbb{N}$ is a collection of k colorings $\texttt{col}_1,\ldots,\texttt{col}_k$. (Note that each coloring is over different sized tuples.) An infinite set $X \subseteq \mathbb{N}$ is k-wise monochromatic if X is monochromatic with respect to all the $\texttt{col}_i$s.

The following is a simple variant of Ramsey's original theorem. (We closely follow the proof of Ramsey's theorem as given in Chap VI, Theorem 4 of [6].)

Theorem 2.3. *For any k-wise finite coloring of $\mathbb{N}$, there is an infinite k-wise monochromatic set $X \subseteq \mathbb{N}$.*

Proof. We proceed by induction on k. If $k=1$, then this is trivially true since C is finite. We now iteratively construct an infinite set of $\mathbb{N}$. Let $\texttt{col}_1, \texttt{col}_2, \ldots, \texttt{col}_k$ be a k-coloring of $\mathbb{N}$. Start with a_0 being the minimum element in $\mathbb{N}$. Consider the following $(k-1)$-wise coloring of $(\mathbb{N} \setminus \{a_0\})$ $\texttt{col}'_1, \ldots, \texttt{col}'_{k-1}$, where $\texttt{col}'_i(S)$ is defined to be $\texttt{col}_{i+1}(S \cup a_0)$. By the induction hypothesis, there exists an infinite $(k-1)$-wise monochromatic set $A_0 \subseteq \mathbb{N} \setminus \{a_0\}$ with respect to coloring $\texttt{col}'_i$s. That is, for $2 \leq i \leq k$, and any set $S, T \subseteq A_0$ with $|S| = |T| = i-1$, we have $\texttt{col}_i(a_0 \cup S) = \texttt{col}_i(a_0 \cup T)$. Call this color C^0_i. Denote the collection of these colors as a vector $\mathbf{C}_0 = (C^0_1, C^0_2, \ldots, C^0_k)$ where $C^0_1 = \texttt{col}_1(a_0)$.

Subsequently, let a_1 be the minimum element in A_0, and consider the $(k-1)$-wise coloring $\texttt{col}'$ of $(A_0 \setminus \{a_1\})$ where $\texttt{col}'_i(S) = \texttt{col}_{i+1}(S \cup \{a_1\})$ for $S \subseteq A_0 \setminus \{a_1\}$. Again, the induction hypothesis yields an infinite $(k-1)$-wise monochromatic set A_1 as before, and similarly the vector $\mathbf{C}_1$. Continuing this procedure, we get an infinite sequence $a_0, a_1, a_2, \ldots$ of natural numbers, an infinite sequence of vectors of k colors $\mathbf{C}_0, \mathbf{C}_1, \ldots$, and an infinite nested sequence of infinite sets $A_0 \supset A_1 \supset A_2 \ldots$. Every A_r contains $a_s, \forall s > r$ and by construction, any set $(\{a_r\} \cup S)$, $S \subseteq A_r$, $|S| = i-1$, has color C^i_r. Since there are only finitely many colors, some vector of colors occurs infinitely often as $\mathbf{C}_{r_1}, \mathbf{C}_{r_2}, \ldots$. The corresponding infinite sequence of elements $a_{r_1}, a_{r_2}, \ldots$ is k-wise monochromatic. □

Proof of Theorem 2.1. Suppose there exists a (t,ε,δ)-tester for functions $f : \mathbf{D} \to \mathbb{N}$. We need to show there is a comparison-based $(t,\varepsilon,2\delta)$-tester for such functions.

By Lemma 2.2, there is a discrete $(t,\varepsilon,2\delta)$-tester $\mathcal{A}$. Equivalently, we have the functions $q^y_{\mathbf{x},\sigma}$ as described in the previous section. We now describe a t-wise finite coloring of $\mathbb{N}$. Consider $s \in [t]$. Given a set $A \subseteq \mathbb{N}^{(s)}$, $\texttt{col}_s(A)$ is a vector indexed by $(y, \mathbf{x}, \sigma)$, where $y \in \mathbf{D}'$, $\mathbf{x} \in \mathbf{D}^s$, and σ is a permutation of

$[s]$. The value of the vector at this entry is defined to be $q^y_{\mathbf{x},\sigma}(A)$. The domain is finite, so the number of dimensions is finite. Since the tester is discrete, the number of possible colors entries is also finite. Applying Theorem 2.3, we know the existence of a t-wise monochromatic infinite set $\mathbf{R} \subseteq \mathbb{N}$. By the monochromatic property, we get that for any $y, \mathbf{x}, \sigma$, and any two sets $A, B \in \mathbf{R}^{(s)}$, $s \le t$, we have $q^y_{\mathbf{x},\sigma}(A) = q^y_{\mathbf{x},\sigma}(B)$. That is, the algorithm $\mathcal{A}$ is a comparison-based tester for functions $f : \mathbf{D} \to \mathbf{R}$.

Consider the strictly monotone map $\phi : \mathbb{N} \to \mathbf{R}$, where $\phi(b)$ is the bth element of $\mathbf{R}$ in sorted order. Now given any function $f : \mathbf{D} \to \mathbb{N}$, consider the function $\phi \circ f : \mathbf{D} \to \mathbf{R}$. Consider an algorithm $\mathcal{A}'$ which on input f runs $\mathcal{A}$ on $\phi \circ f$. More precisely, whenever $\mathcal{A}$ queries a point x, it gets answer $\phi \circ f(x)$. Observe that if f is monotone (or ε-far from monotone), then so is $\phi \circ f$, and therefore, the algorithm $\mathcal{A}'$ is a $(t, \varepsilon, 2\delta)$-tester of $\phi \circ f$. Since the range of $\phi \circ f$ is $\mathbf{R}$, $\mathcal{A}'$ is comparison-based. □

3 Lower bounds

We assume that n is a power of 2 and set $\ell := \log_2 n$, and think of $[n]$ as $\{0, 1, \dots, n-1\}$. For any integer $0 \le z < n$, we think of the binary representation of z as an ℓ-bit vector $(z_1, z_2, \dots, z_\ell)$, where z_1 is the least significant bit (although, z_1 is leftmost in the way written).

We first start with a map which allows us to reduce functions on hypergrids from those on hypercubes. The map is the following natural one: $\phi : [n]^d \to \{0,1\}^{d\ell}$. For any $\vec{y} = (y_1, y_2, \dots, y_d) \in [n]^d$, we concatenate binary representations of the y_is in order to get a $d\ell$-bit vector $\phi(\vec{y})$. Hence, we can transform a function $f : \{0,1\}^{d\ell} \to \mathbb{N}$ into a function $\widetilde{f} : [n]^d \to \mathbb{N}$ by defining $\widetilde{f}(\vec{y}) := f(\phi(\vec{y}))$.

In Section 3.1, we describe a distribution of functions over the hypercube with equal mass on monotone and ε-far from monotone functions. The key property is that for a function drawn from this distribution, any deterministic comparison based algorithm errs in classifying it with non-trivial probability. This property will be used in conjunction with the above mapping to get our final lower bound Section 3.2.

3.1 The hard distribution

We focus on functions $f : \{0,1\}^m \to \mathbb{N}$. (Eventually, we set $m = d\ell$.) Given any $x \in \{0,1\}^m$, we let $\mathtt{val}(x) := \sum_{i=1}^m 2^{i-1} x_i$ denote the number for which x is the binary representation. Here, x_1 denotes the least significant bit of x.

For convenience, we let ε be a power of $1/2$. For $k \in \{1, \dots, 1/2\varepsilon\}$, we let

$$S_k := \left\{x : \mathtt{val}(x) \in [2(k-1)\varepsilon 2^m, 2k\varepsilon 2^m - 1]\right\}.$$

Note that the sets S_k partition the hypercube, with each $|S_k| = \varepsilon 2^{m+1}$. In fact, each S_k is a subhypercube of dimension $m' := m + 1 - \log(1/\varepsilon)$, with the minimal element having all zeros in the m' least significant bits, and the maximal element having all ones in those.

We describe a distribution $\mathcal{F}_{m,\varepsilon}$ on functions. The support of $\mathcal{F}_{m,\varepsilon}$ consists of $f(x) = 2\mathtt{val}(x)$ and $m'/(2\varepsilon)$ functions indexed as $g_{j,k}$ with $j \in [m']$ and $k \in [1/(2\varepsilon)]$, defined as follows.

$$g_{j,k}(x) = \begin{cases} 2\mathtt{val}(x) - 2^j - 1 & \text{if } x_j = 1 \text{ and } x \in S_k, \\ 2\mathtt{val}(x) & \text{otherwise.} \end{cases}$$

The distribution $\mathcal{F}_{m,\varepsilon}$ puts probability mass $1/2$ on the function $f = 2\texttt{val}$ and ε/m' on each of the $g_{j,k}$s. All these functions take distinct values on their domain. Note that $2\texttt{val}$ induces a total order on $\{0,1\}^m$.

The distinguishing problem: Given query access to a random function f from $\mathcal{F}_{m,\varepsilon}$, we want a deterministic comparison-based algorithm that declares that $f = 2\texttt{val}$ or $f \neq 2\texttt{val}$. We refer to any such algorithm as a *distinguisher*. Naturally, we say that the distinguisher errs on f if its declaration is wrong. We first prove a lower bound for non-adaptive distinguishers.

Lemma 3.1. *Any deterministic,* non-adaptive*, comparison-based distinguisher $\mathcal{A}$ making fewer than $t \leq m'/(8\varepsilon)$ queries, errs with probability at least* $1/8$.

Proof. Let X be the set of points queried by the distinguisher. Set $X_k = X \cap S_k$; these form a partition of X. We say that a pair of points (x,y) *captures* the (unique) coordinate j, if j is the largest coordinate where $x_j \neq y_j$. (By largest coordinate, we refer to most significant bit.) For a set Y of points, we say Y captures coordinate j if there is a pair in Y that captures j. The main technical argument is encapsulated in the following two claims.

Claim 3.2. *For any j,k, if the algorithm distinguishes between* $\texttt{val}$ *and $g_{j,k}$, then X_k captures j.*

Proof. If the algorithm distinguishes between $\texttt{val}$ and $g_{j,k}$, there must exist $(x,y) \in X$ such that $\texttt{val}(x) < \texttt{val}(y)$ and $g_{j,k}(x) > g_{j,k}(y)$. We claim that x and y capture j; this will also imply they lie in the same $S_{k'}$ since the $m-j$ most significant bit of x and y are the same.

Firstly, observe that we must have $y_j = 1$ and $x_j = 0$; otherwise,

$$g_{j,k}(y) - g_{j,k}(x) \geq 2(\texttt{val}(y) - \texttt{val}(x)) > 0$$

contradicting the supposition. Now suppose (x,y) don't capture j implying there exists $i > j$ which is the largest coordinate at which they differ. Since $\texttt{val}(y) > \texttt{val}(x)$ we have $y_i = 1$ and $x_j = 0$. Therefore, we have

$$g_{j,k}(y) - g_{j,k}(x) \geq 2(\texttt{val}(y) - \texttt{val}(x)) - 2^j - 1 \geq (2^i + 2^j) - \sum_{1 \leq r < i} 2^r - 2^j - 1 > 0\,.$$

So, x,y capture j and lie in the same $S_{k'}$. If $k' \neq k$, then again $g_{j,k}(y) - g_{j,k}(x) = 2(\texttt{val}(y) - \texttt{val}(x)) > 0$. Therefore, X_k captures j. □

Claim 3.3. *A set Y of points captures at most $|Y| - 1$ coordinates.*

Proof. We apply induction on $|Y|$. When $|Y| = 2$, this is trivially true. Otherwise, pick the largest coordinate j captured by Y and let $Y_0 = \{y : y_j = 0\}$ and $Y_1 = \{y : y_j = 1\}$. By induction, Y_0 captures at most $|Y_0| - 1$ coordinates, and Y_1 captures at most $|Y_1| - 1$ coordinates. Pairs $(x,y) \in Y_0 \times Y_1$ only capture coordinate j. Therefore, the total number of captured coordinates is at most

$$|Y_0| - 1 + |Y_1| - 1 + 1 = |Y| - 1\,.$$ □

We now complete the proof of Lemma 3.1. If $|X| \leq m'/8\varepsilon$, then there exist at least $1/4\varepsilon$ values of k such that $|X_k| \leq m'/2$. By Claim 3.2 and Claim 3.3, each such X_k captures at most $m'/2$ coordinates. Therefore, there exist at least

$$\frac{1}{4\varepsilon} \cdot \frac{m'}{2} = \frac{m'}{8\varepsilon}$$

functions $g_{j,k}$ that are indistinguishable from the monotone function $2\texttt{val}$ to a comparison-based procedure that queries X. This implies the distinguisher must err (make a mistake on either these $g_{j,k}$s or $2\texttt{val}$) with probability at least

$$\min\left(\frac{\varepsilon}{m'} \cdot \frac{m'}{(8\varepsilon)}, \frac{1}{2}\right) = \frac{1}{8}. \qquad \square$$

A basic proposition reduces adaptive distinguishers to non-adaptive ones. This crucially uses the total order given by $\texttt{val}(x)$.

Proposition 3.4. *Suppose there exists a deterministic comparison-based distinguisher $\mathcal{A}$ that makes at most t queries for inputs drawn from distribution $\mathcal{F}_{m,\varepsilon}$. Then there exists a deterministic* non-adaptive *comparison-based distinguisher $\mathcal{A}'$ making at most t queries whose probability of error on inputs from $\mathcal{F}_{m,\varepsilon}$ is at most that of $\mathcal{A}$.*

Proof. We represent $\mathcal{A}$ as a comparison tree. For any path in $\mathcal{A}$, the total number of distinct domain points involved in comparisons is at most t. Note that $2\texttt{val}(x)$ is a total order, since for any x, y either $\texttt{val}(x) < \texttt{val}(y)$ or vice versa. We say that a comparison between $f(x)$ and $f(y)$ is *inconsistent* with $\texttt{val}$ if $f(x) < f(y)$, $\texttt{val}(x) > \texttt{val}(y)$ or vice versa. We construct a comparison tree $\mathcal{A}'$ where we simply reject whenever a comparison is inconsistent with the total order, and otherwise mimics $\mathcal{A}$. The comparison tree of $\mathcal{A}'$ has an error probability at most that of $\mathcal{A}$ since it never errs when $\mathcal{A}$ doesn't err. Furthermore, the tree is just a path and thus can be modeled as a non-adaptive distinguisher as follows. We simply query upfront all the points involving points on this path, and make the relevant comparisons for the output. $\square$

Our main lemma is a direct consequence of Proposition 3.4 and Lemma 3.1.

Lemma 3.5. *Any deterministic comparison-based distinguisher that makes less than $m'/(8\varepsilon)$ queries errs with probability at least $1/8$ on a function drawn from $\mathcal{F}_{\varepsilon,m}$.*

3.2 The final bound

Recall, given function $f : \{0,1\}^{d\ell} \to \mathbb{N}$, we have the function $\widetilde{f} : [n]^d \to \mathbb{N}$ by defining $\widetilde{f}(\vec{y}) := f(\phi(\vec{y}))$. We start with the following observation.

Proposition 3.6. *The function $\widetilde{2\texttt{val}}$ is monotone and every $\widetilde{g_{j,k}}$ is $\varepsilon/2$-far from being monotone.*

Proof. Let $\vec{u}$ and $\vec{v}$ be elements in $[n]^d$ such that $\vec{u} \prec \vec{v}$. We have $\texttt{val}(\phi(\vec{u})) < \texttt{val}(\phi(\vec{v}))$, so $\widetilde{2\texttt{val}}$ is monotone. For the latter, it suffices to exhibit a matching of violated pairs of cardinality $\varepsilon 2^{d\ell}$ for $\widetilde{g_{j,k}}$. This is given by pairs $(\vec{u}, \vec{v})$ where $\phi(\vec{u})$ and $\phi(\vec{v})$ only differ in their jth coordinate, and are both contained in S_k. Note that these pairs are comparable in $[n]^d$ and are violations. $\square$

Theorem 3.7. *Any* $(t, \varepsilon/2, 1/16)$*-monotonicity tester for* $f : [n]^d \to \mathbb{N}$*, must have*

$$t \geq \frac{d \log n - \log(1/\varepsilon)}{8\varepsilon}.$$

Proof. By Theorem 2.1, it suffices to show this for comparison-based $(t, \varepsilon/2, 1/8)$ testers. By Yao's minimax lemma, it suffices to produce a distribution $\mathcal{D}$ over functions $f : [n]^d \to \mathbb{N}$ such that any deterministic comparison-based $(t, \varepsilon/2, 1/8)$-monotonicity tester for $\mathcal{D}$ must have $t \geq s$, where

$$s := \frac{d \log n - \log(1/\varepsilon)}{8\varepsilon}.$$

Consider the distribution $\mathcal{D}$ where we generate f from $\mathcal{F}_{m,\varepsilon}$ and output $\widetilde{f}$. Suppose $t < s$. By Proposition 3.6, the deterministic comparison based monotonicity tester acts as a deterministic comparison-based distinguisher for $\mathcal{F}_{m,\varepsilon}$ making fewer than s queries, contradicting Lemma 3.1. □

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 and ECCC. [doi:10.1016/j.ic.2006.06.001] 454

[2] NIR AILON, BERNARD CHAZELLE, S. COMANDUR, AND DING LIU: Estimating the distance to a monotone function. *Random Structures Algorithms*, 31(3):1704–1711, 2007. Preliminary version in RANDOM'04. [doi:10.1002/rsa.20167] 454

[3] TUĞKAN BATU, RONITT RUBINFELD, AND PATRICK WHITE: Fast approximate *PCP*s for multidimensional bin-packing problems. *Inform. and Comput.*, 196(1):42–56, 2005. Preliminary version in APPROX'99. [doi:10.1016/j.ic.2004.10.001] 454

[4] 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 and ECCC. [doi:10.1007/s00037-012-0040-x] 454, 455

[5] ERIC BLAIS, SOFYA RASKHODNIKOVA, AND GRIGORY YAROSLAVTSEV: Lower bounds for testing properties of functions on hypergrid domains. In *Proc. 29th IEEE Conf. on Computational Complexity (CCC'14)*, pp. 309–320, 2014. Preliminary version in ECCC. [doi:10.1109/CCC.2014.38] 454, 455

[6] BÉLA BOLLOBÁS: *Modern Graph Theory*. Springer, 2000. 457

[7] DANY BRESLAUER, ARTUR CZUMAJ, DEVDATT P. DUBHASHI, AND FRIEDHELM MEYER AUF DER HEIDE: Transforming comparison model lower bounds to the parallel-random-access-machine. *Inform. Process. Lett.*, 62(2):103–110, 1997. [doi:10.1016/S0020-0190(97)00032-X] 455

[8] JOP BRIËT, SOURAV CHAKRABORTY, DAVID GARCÍA-SORIANO, AND ARJEH MATSLIAH: Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012. Preliminary version in RANDOM'99 and ECCC. [doi:10.1007/s00493-012-2765-1] 454

[9] JOSHUA BRODY: Personal communication, 2013. 454, 455

[10] DEEPARNAB CHAKRABARTY AND C. SESHADHRI: Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proc. 45th STOC*, pp. 419–428. ACM Press, 2013. Preliminary version in ECCC. [doi:10.1145/2488608.2488661] 454

[11] DEEPARNAB CHAKRABARTY AND C. SESHADHRI: An optimal lower bound for monotonicity testing over hypergrids. In *Proc. 16th Internat. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'13)*, pp. 425–435, 2013. Preliminary version in ECCC. [doi:10.1007/978-3-642-40328-6_30] 453

[12] YEVGENIY DODIS, ODED GOLDREICH, ERIC LEHMAN, SOFYA RASKHODNIKOVA, DANA RON, AND ALEX SAMORODNITSKY: Improved testing algorithms for monotonicity. In *Proc. 2nd Internat. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'99)*, pp. 97–108, 1999. Available at ACM-DL. Preliminary version in ECCC. 454

[13] 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] 454, 456

[14] ELDAR FISCHER: On the strength of comparisons in property testing. *Inform. and Comput.*, 189(1):107–116, 2004. Preliminary version in ECCC. [doi:10.1016/j.ic.2003.09.003] 454, 455, 456

[15] ELDAR FISCHER, ERIC LEHMAN, ILAN NEWMAN, SOFYA RASKHODNIKOVA, RONITT RUBINFELD, AND ALEX SAMORODNITSKY: Monotonicity testing over general poset domains. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 474–483. ACM Press, 2002. [doi:10.1145/509907.509977] 454

[16] ODED GOLDREICH, SHAFI GOLDWASSER, ERIC LEHMAN, DANA RON, AND ALEX SAMORODNITSKY: Testing monotonicity. *Combinatorica*, 20:301–337, 2000. Preliminary version in FOCS'98. [doi:10.1007/s004930070011] 453, 454

[17] ODED GOLDREICH, SHAFI GOLDWASSER, AND DANA RON: Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. Preliminary version in FOCS'96 and ECCC. [doi:10.1145/285055.285060] 453

[18] 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] 454

[19] ERIC LEHMAN AND DANA RON: On disjoint chains of subsets. *J. Combin. Theory Ser. A*, 94(2):399–404, 2001. [doi:10.1006/jcta.2000.3148] 454

[20] MICHAL PARNAS, DANA RON, AND RONITT RUBINFELD: Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 6(72):1012–1042, 2006. Preliminary vesion in ECCC. [doi:10.1016/j.jcss.2006.03.002] 454

[21] RONITT RUBINFELD AND MADHU SUDAN: Robust characterization of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):647–668, 1996. [doi:10.1137/S0097539793255151] 453

AUTHORS

Deeparnab Chakrabarty
Researcher
Microsoft Research
9 Lavelle Road, Bangalore, India, 560001
deeparnab@gmail.com
http://research.microsoft.com/en-us/um/people/dechakr/

C. Seshadhri
Assistant Professor
Department of Computer Science
University of California, Santa Cruz
scomandu@ucsc.edu
csesha@gmail.com
https://users.soe.ucsc.edu/~sesh/

ABOUT THE AUTHORS

DEEPARNAB CHAKRABARTY received his B. Tech. from IIT Bombay in 2003 and Ph. D. from Georgia Institute of Technology in 2008. His graduate adviser was Vijay V. Vazirani. After stints at the universities of Waterloo and Pennsylvania, he crossed the pond again to join Microsoft Research in Bangalore where he has been a researcher since 2011. He is interested in understanding efficient algorithms using the lens of optimization and has worked on approximation algorithms, property testing, and on algorithmic questions arising in game theory and economics.

C. SESHADHRI (Seshadhri Comandur according to his passport, and Sesh according to his friends) follows the common naming style of South India, the native land of his parents. In three of his early papers, including one cited in this bibliography, his name appears as S. Comandur. The interested reader is invited to consult the OAQ (Occasionally Asked Questions) section of his website for more details. Sesh got his B. Tech. from IIT Kanpur in 2003 and his Ph. D. from Princeton University in 2008. His Ph. D. advisor was Bernard Chazelle. He spent two years at IBM Almaden, and in 2010 he became a member of technical staff at Sandia National Laboratories, Livermore, California. In 2015 he joins the faculty of the University of California at Santa Cruz. This paper was written during his time at Sandia. His research focuses on how random sampling can be used for algorithms for massive inputs. In theory, this manifests itself as work in property testing. In practice, this has led to research on algorithms for massive real-world graphs. He is increasingly interested in algorithmics beyond the worst-case, and bridging the gap between theory and practice for big data applications.