Adaptive Boolean Monotonicity Testing in Total Influence Time

Deeparnab Chakrabarty[1]
Dartmouth College, Hanover, NH 03755, USA
deeparnab@dartmouth.edu

C. Seshadhri[2]
University of California, Santa Cruz, CA 95064, USA
sesh@ucsc.edu

Abstract

Testing monotonicity of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is an important problem in the field of property testing. It has led to connections with many interesting combinatorial questions on the directed hypercube: routing, random walks, and new isoperimetric theorems. Denoting the proximity parameter by ε, the best tester is the non-adaptive $\widetilde{O}(\varepsilon^{-2}\sqrt{n})$ tester of Khot-Minzer-Safra (FOCS 2015). A series of recent results by Belovs-Blais (STOC 2016) and Chen-Waingarten-Xie (STOC 2017) have led to $\widetilde{\Omega}(n^{1/3})$ lower bounds for *adaptive* testers. Reducing this gap is a significant question, that touches on the role of adaptivity in monotonicity testing of Boolean functions.

We approach this question from the perspective of *parametrized property testing*, a concept recently introduced by Pallavoor-Raskhodnikova-Varma (ACM TOCT 2017), where one seeks to understand performance of testers with respect to parameters other than just the size. Our result is an adaptive monotonicity tester with one-sided error whose query complexity is $O(\varepsilon^{-2}\mathbf{I}(f)\log^5 n)$, where $\mathbf{I}(f)$ is the total influence of the function. Therefore, adaptivity provably helps monotonicity testing for low influence functions.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Monotonicity Testing, Influence of Boolean Functions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.20

1 Introduction

Monotonicity testing of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ is a classic problem in property testing that has been extensively studied over the last two decades [11, 8, 6, 2, 4, 3, 9, 1, 5]. Consider the coordinate-wise partial order over $\{0,1\}^n$, denoted by $\prec$. A function f is monotone if $\forall x, y \in \{0,1\}^n$ where $x \prec y$, $f(x) \le f(y)$. The distance between two functions f, g is $\Pr_x[f(x) \neq g(x)]$, where x is u.a.r in $\{0,1\}^n$. The distance of f to monotonicity is the minimum distance of f to a monotone g. Typically, we say that f is ε-far if the distance to monotonicity is at least ε.

The goal of monotonicity testing is to design efficient (poly(n)) randomized procedures that distinguish monotone functions from those that are far from monotone. Formally, given query access to f and a proximity parameter $\varepsilon > 0$, a monotonicity tester is a (randomized) procedure that accepts if f is monotone, and rejects if f is ε-far from being monotone. Both

[1] Supported by NSF CCF-1813165.
[2] Supported by NSF TRIPODS CCF-1740850 and NSF CCF-1813165.

the above guarantees should hold with probability at least 2/3. A tester has one-sided error if it always accepts a monotone function, or equivalently, always provides a certificate of rejection. A tester is called *non-adaptive* if all the queries can be made without seeing any of the answers. On the other hand, if the tester's queries depend on previous answers, the tester is adaptive. As we explain below, the power of adaptivity is the central open question in Boolean monotonicity testing.

Raskhodnikova [11], Goldreich et al. [8], and Dodis et al. [6] initiated the study of monotonicity testing by describing a $O(n/\varepsilon)$-query non-adaptive, one-sided tester. Their "edge tester" repeats this simple procedure $O(n/\varepsilon)$ times: sample a random edge of the hypercube and test for a monotonicity violation among its endpoints. Chakrabarty and Seshadhri [2] described the first $o(n)$-query tester, by proving connections between monotonicity testing and directed isoperimetry theorems. Their "path tester" performs random walks on the directed hypercube. Chen-Servedio-Tan [4] gave a tighter analysis to get an $\widetilde{O}(n^{5/6}\varepsilon^{-4})$ bound. In a remarkable result, Khot, Minzer, and Safra [9] (henceforth KMS) prove that the path tester works in $\widetilde{O}(\sqrt{n}/\varepsilon^2)$ queries. This was achieved by a deep isoperimetric result, a directed analog of Talagrand's isoperimetry theorem [12].

Fischer et al. [7] had proven an $\Omega(\sqrt{n})$ lower bound for non-adaptive, one-sided testers. Using sophisticated techniques, Chen-De-Servedio-Tan [3] proved an $\Omega(n^{1/2-\delta})$ (for any fixed $\delta > 0$) lower bound for non-adaptive, two-sided testers. All in all, this nearly resolves the non-adaptive complexity of monotonicity testing.

In a major recent advance, Belovs-Blais [1] proved the first polynomial lower bound of $\widetilde{\Omega}(n^{1/4})$ for any *adaptive* tester. This was further improved by Chen-Waingarten-Xie [5] to $\widetilde{\Omega}(n^{1/3})$. These are highly non-trivial results. Belovs-Blais also gave a $O(\log n)$ query adaptive monotonicity tester for the class of regular linear threshold functions (LTFs), the hard functions of Chen et al. [3]. This underscores the challenge of proving adaptive lower bounds and leaves a tantalizing open question. *Can adaptivity always help in Boolean monotonicity testing over the hypercube?*

We approach this question from the perspective of *parametrized property testing*, a concept recently introduced by Pallavoor-Raskhodnikova-Varma [10]. One seeks to understand the performance of testers with respect to parameters other than just the size of the domain. If one can beat existing lower bounds (for some settings of the parameters), this gives insight into the hardest functions for monotonicity testing.

Our main result is the existence of adaptive monotonicity testers parametrized by the total influence, $\mathbf{I}(f)$. Letting $\mathcal{D}$ be the uniform distribution over all pairs (x, y) at Hamming distance 1, $\mathbf{I}(f) := n \cdot \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq f(y)]$.

▶ **Theorem 1.** *Consider the class of functions $f : \{0,1\}^n \to \{0,1\}$ with total influence at most I. There exists a one-sided, adaptive monotonicity tester for this class with query complexity $O(I\varepsilon^{-2}\log^5 n)$.*

A claim of KMS (Theorem 9.1 in [9]) implies that if $\mathbf{I}(f) > 6\sqrt{n}$, then the edge tester is itself a $O(n/\mathbf{I}(f)))$ tester. Combined with Theorem 1, one gets an adaptive $\widetilde{O}(\min(\mathbf{I}(f), n/\mathbf{I}(f))\varepsilon^{-2})$-query tester. The trade-off point is basically $\sqrt{n}$, the maximum possible influence of a monotone function.

1.1 Perspective on Theorem 1

- When $\mathbf{I}(f) \ll \sqrt{n}$, Theorem 1 beats the lower bounds of [7, 3]. Indeed, the lower bound of Fischer et al. [7] holds for *constant* influence functions, and so adaptivity is crucial for a result like Theorem 1. As mentioned earlier, it was known that adaptivity helps for the structured class of regular LTFs [1]. What is intriguing about Theorem 1 is that no strong structural assumption is required to get $o(\sqrt{n})$ query testers.

- All adaptive lower bound constructions have functions with influence $\Theta(\sqrt{n})$ [1, 5]. In light of Theorem 1, this is perhaps no accident. Theorem 1 shows that this is the hard regime for monotonicity testing.
- The $o(n)$ non-adaptive testers are obtained by directed random walks in the hypercube. One starts at a random x and walks "up" (consistent with the partial order) the hypercube to reach some y. Then, the tester compares $f(x), f(y)$ to detect non-monotonicity. (The intermediate vertices in the random walk are not queried, and this is crucial for getting the $o(n)$ bound.) The algorithm of Theorem 1 is fundamentally different; it performs standard, *undirected* random walks on the hypercube. The endpoints might not even be comparable, so only querying these is of no use. This is exactly where adaptivity helps, since we can perform binary search to find violations in this path. This leads to a better monotonicity tester for functions with influence $o(\sqrt{n})$.

1.2 Proof Idea

The $o(n)$ testers of [2, 4, 9] perform random walks on the directed hypercube with orientation corresponding to the partial order, and query the function at the endpoints of this walk. Their analysis shows that if the function is ε-far from being monotone, then there is a significant probability of encountering a violation. At a high level, one of the main observations of [2, 4, 9] is that "longer" walks lead to higher chances of catching a violation. However, the vast majority of vertices in the hypercube have Hamming weight $n/2 \pm \Theta(\sqrt{n})$. For the purposes of monotonicity testing, one can assume that vertices outside these middle layers do not participate in any violations to monotonicity. Each step of a directed walk increments the Hamming weight, and thus the walk length can be at most $\Theta(\sqrt{n})$. This $\sqrt{n}$ is precisely what shows up in the final bound of KMS.

Our insight is that one can perform an analogous analysis as above for random walks on the undirected hypercube H_n. These walks can have length $\omega(\sqrt{n})$. Suppose we perform an ℓ-step random walk (on H_n) from x that ends at y. Note that with high probability, x and y will not be comparable. Nonetheless, if $f(x) \neq f(y)$, then the walk has passed through an influential edge. The power of adaptivity is that we can find such an influential edge through binary search. This idea of using binary search is indeed also present in a $O(\log n)$-query algorithm of Belovs and Blais [1] for adaptive monotonicity testers for regular linear threshold functions.

However, we are not interested in finding an influential edge but rather a *violated* edge. Fix a violated edge (u, v). Our insight is to lower bound the probability that (u, v) is the *unique* influential edge in the undirected random walk. In this scenario, binary search is guaranteed to find a violation. There are two opposing forces here. First, the probability that (u, v) at all appears in the random walk grows as ℓ/n. On the other hand, the probability that this is the unique influential edge decreases with ℓ; indeed, this probability depends on the influence of f. A fairly simple calculation shows that for all but an $\ell\mathbf{I}(f)/n$ fraction of "bad" vertices, an ℓ-length from a vertex encounters no influential edge with constant probability. Putting these together, we see that setting $\ell \sim n/\mathbf{I}(f)$ would lead to a good probability of catching a violation; note that if $\mathbf{I}(f) \ll \sqrt{n}$, the desired length of the walk would indeed be $\gg \sqrt{n}$.

The only fly in the above ointment is if all the violated edges were concentrated on the bad vertices. This is where we invoke the connection between distance to monotonicity and isoperimetry; the directed Talagrand inequality of KMS essentially precludes this scenario by showing that violated edges need to be "spread apart". The actual math is more subtle. KMS gives a trade-off between the concentration of the violated edges and the number of violated edges. If all violated edges are incident only on a few vertices, then there must be a lot of violated edges. The latter is where the edge-tester, which is nothing but a length 1 random walk, works. This analysis is similar to what is done in KMS.

1.3 Preliminaries

We use H_n to denote the standard (undirected) hypercube graph on $\{0,1\}^n$, where all pairs at Hamming distance 1 are connected. An edge (x,y) of H_n is influential, if $f(x) \neq f(y)$. The number of influential edges is precisely $\mathbf{I}(f) \cdot 2^{n-1}$. An influential edge (x,y) is a violating edge if $x \prec y$, $f(x) = 1$, and $f(y) = 0$. Our tester will perform random walks of H_n. Note that H_n is regular, so this is a symmetric Markov Chain.

We crucially use the central result of KMS, which is obtained as a consequence of the directed analogue of Talagrand's isoperimetric theorem.

▶ **Lemma 2** (Lemma 7.1 in [9], paraphrased). *Given any Boolean function $f : \{0,1\}^n \to \{0,1\}$ that is ε-far from being monotone, there exists a subgraph $G = (A,B,E)$ of the hypercube and parameters $\sigma \in (0,1)$, $d \in \mathbb{N}$ such that*

- *Each edge $(a,b) \in E$ with $a \in A$ and $b \in B$ is a violating edge.*
- *The degree of each vertex in B is exactly d and the degree of each vertex in A is at most $2d$.*
- *$|B| = \sigma \cdot 2^n$.*
- *$\sigma^2 d = \Omega(\varepsilon^2/\log^4 n)$.*

2 Tester and Analysis

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function over the hypercube with total influence $\mathbf{I}(f)$.

Algorithm 1 Adaptive Monotonicity Tester for Boolean Functions.

```
Input: A Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a parameter $\varepsilon \in (0,1)$
1. Choose $k \in_R \{0,1,2,\ldots,\lceil \log n \rceil\}$ uniformly at random. Set $\ell := 2^k$.
2. Choose $x \in \{0,1\}^n$ uniformly at random.
3. Perform an $\ell$-length random walk $p$ on $H_n$ starting from $x$ to reach $y \in \{0,1\}^n$.
4. If $f(x) \neq f(y)$:
   a. Perform binary search on $p$ to find an influential edge $(u,v) \in p$.
   b. REJECT if $(u,v)$ is a monotonicity violation.
5. ACCEPT.
```

Theorem 1 follows directly from the following theorem by repeating the above subroutine the appropriate number of times. Note that the subroutine above does not need to know $\mathbf{I}(f)$.

▶ **Theorem 3.** *If f is ε-far from being monotone, then the algorithm described in Algorithm 1 rejects with probability $\Omega\left(\frac{\varepsilon^2}{\mathbf{I}(f)\log^5 n}\right)$.*

▶ **Definition 4.** Given a positive integer ℓ, a vertex $x \in \{0,1\}^n$ is denoted *ℓ-sticky* if an ℓ-length random walk from x on H_n contains no influential edges with probability $\geq 1/2$. A vertex is called non-ℓ-sticky otherwise. An edge is ℓ-sticky if both endpoints are ℓ-sticky.

▶ **Observation 5.** *If a vertex x is ℓ-sticky and $\ell' < \ell$, then x is ℓ'-sticky as well.*

▶ **Lemma 6.** *The fraction of non-ℓ-sticky vertices of a hypercube is at most $\frac{2\ell \cdot \mathbf{I}(f)}{n}$.*

Proof. Given $x \in \{0,1\}^n$ and a positive integer $\ell > 0$, define the random variable $Z_{x,\ell}$ that is the number of influential edges in a random walk of length ℓ starting from x. Therefore, x is non-ℓ-sticky iff $\Pr[Z_{x,\ell} > 0] > 1/2$. Let N denote the set of non-ℓ-sticky vertices.

Since $Z_{x,\ell}$ is non-negative and integer valued we get $\Pr[Z_{x,\ell} > 0] \leq \mathbf{E}[Z_{x,\ell}]$.

$$|N|/2^n < \frac{2}{2^n}\sum_{x\in N}\Pr[Z_{x,\ell}>0] < \frac{2}{2^n}\sum_{x\in\{0,1\}^n}\Pr[Z_{x,\ell}>0] \;\leq\; \frac{2}{2^n}\sum_{x\in\{0,1\}^n}\mathbf{E}[Z_{x,\ell}] \tag{1}$$

The RHS above is precisely twice the expected number of influential edges encountered in an ℓ-length random walk starting from the uniform distribution on H_n. Let $\mathcal{P}_\ell$ denote the uniform distribution on ℓ-length paths in H_n. For $p \sim \mathcal{P}_\ell$, p_t denotes the tth edge in p, and let $\chi(e)$ be the indicator for edge e being influential. The RHS of (1) is equal to $2\mathbf{E}_{p\sim\mathcal{P}_\ell}[\sum_{t\leq\ell}\chi(p_t)] = 2\sum_{t\leq\ell}\mathbf{E}_{p\sim\mathcal{P}_\ell}[\chi(p_t)]$. Since the uniform distribution is stationary for random walks on H_n, the distribution induced on p_t is the uniform distribution on edges in H_n. Thus, $\mathbf{E}_{p\sim\mathcal{P}_\ell}[\chi(p_t)] = \mathbf{I}(f)/n$ and the RHS of (1) is $2\ell\cdot\mathbf{I}(f)/n$. ◀

For any integer $\ell > 0$, let F_ℓ be the set of ℓ-sticky violating edges. That is,

$$F_\ell := \{(x,y) \in H_n : (x,y) \text{ is violating and, } x,y \text{ are } \ell\text{-sticky}\}$$

▶ **Lemma 7.** *If ℓ is the length of the random walk chosen in Step 1, then Algorithm 1 rejects with probability $\Omega\left(\frac{\ell}{n}\cdot\frac{|F_\ell|}{2^n}\right)$.*

Proof. Fix an edge $(u,v) \in F_\ell$. Let $p = (e_1,\ldots,e_\ell)$ be the edges of a random walk p. Each edge e_t for $1 \leq t \leq \ell$ is a uniformly sampled random edge of H_n. Therefore, for any fixed edge (u,v), $\Pr[e_t = (u,v)] = \frac{1}{n\cdot 2^{n-1}}$.

Now, given $1 \leq t \leq \ell$, define $\mathcal{E}^t_{u,v}$ to be the event that the tth edge of p is (u,v) *and* no other edge of p is influential. Define $\mathcal{E}_{u,v}$ to be the disjoint union $\vee_{t=1}^{\ell}\mathcal{E}^t_{u,v}$. Observe that for two distinct edges (u,v) and (u',v') in F_ℓ, the events $\mathcal{E}_{u,v}$ and $\mathcal{E}_{u',v'}$ are disjoint. Therefore,

$$\Pr[\text{Algorithm 1 rejects for length } \ell] \geq \Pr[\bigvee_{(u,v)\in F_\ell}\mathcal{E}_{u,v}] = \sum_{(u,v)\in F_\ell}\Pr[\mathcal{E}_{u,v}] \tag{2}$$

The inequality follows since if $\mathcal{E}_{u,v}$ occurs then the end points of p must have differing values and binary search on p will return the violation (u,v). The equality follows since the events are mutually exclusive.

Consider the event $\mathcal{E}^t_{u,v}$. For this event to occur, e_t must be (u,v). Consider the conditional probability $\Pr[\mathcal{E}^t_{u,v} \mid e_t = (u,v)]$. Let $\mathcal{F}_u$ be the event that a $(t-1)$-length random walk from u contains no influential edges, and let $\mathcal{F}_v$ be the event that an *independent* $(\ell-t)$-length random walk from v contains no influential edges.

▶ **Claim 8.** $\Pr[\mathcal{E}^t_{u,v} \mid e_t = (u,v)] = \Pr[\mathcal{F}_u \wedge \mathcal{F}_v] = \Pr[\mathcal{F}_u]\cdot\Pr[\mathcal{F}_v]$

Proof. Conditioned on $e_t = (u,v)$, the distribution of the first $(t-1)$ steps of the random walk is the uniform distribution of $(t-1)$-length paths that end at u. This is the same distribution of the $(t-1)$-length random walk starting from u. The distribution of the last $(\ell-t)$ steps, conditioned on $e_t = (u,v)$, is the $(\ell-t)$-length random walk starting from v. ◀

Since (u,v) is an ℓ-sticky edge, by Obs 5 and Definition 4, $\Pr[\mathcal{F}_u] \geq 1/2$ and $\Pr[\mathcal{F}_v] \geq 1/2$. The proof is completed by plugging the following bound into (2).

$$\Pr[\mathcal{E}_{u,v}] = \sum_{t=1}^{\ell} \Pr[\mathcal{E}^t_{u,v}] = \sum_{t=1}^{\ell} \Pr[e_t = (u,v)] \cdot \Pr[\mathcal{E}^t_{u,v} \mid e_t = (u,v)]$$
$$= \frac{2}{n \cdot 2^n} \sum_{t=1}^{\ell} \Pr[\mathcal{E}^t_{u,v} \mid e_t = (u,v)] \geq \frac{\ell}{4n \cdot 2^n} \quad ◀$$

We complete the proof of Theorem 3.

Proof of Theorem 3. Let $G = (A,B,E), \sigma, d$ be as in Lemma 2.

Case 1: $\frac{n\sigma}{16\mathbf{I}(f)} \leq 1$.

In this case, we argue that the edge tester succeeds. More precisely, consider the setting of the algorithm described in Algorithm 1 that sets $\ell = 1$, that is, the tester checks whether a random edge of H_n is a violation. This setting occurs with probability $\Omega(1/\log n)$. The number of violated edges is at least $\sigma d 2^n$, the number of edges in $G = (A,B,E)$ (as defined in Lemma 2). Since $\sigma^2 d = \Omega(\varepsilon^2/\log^4 n)$, $|E| = \Omega(\frac{\varepsilon^2}{\log^4 n} \cdot \frac{2^n}{\sigma})$. Since the number of edges of the hypercube is $\Theta(n2^n)$, we get that the probability Algorithm 1 obtains a violation is $\Omega(\frac{\varepsilon^2}{\log^5 n} \cdot \frac{1}{n\sigma})$. By assumption, $\frac{n\sigma}{16\mathbf{I}(f)} \leq 1$, so this probability is $\Omega(\frac{\varepsilon^2}{\mathbf{I}(f)\log^5 n})$.

Case 2: $\frac{n\sigma}{16\mathbf{I}(f)} > 1$.

In this case, there exists a non-negative power of 2 (call it ℓ^*) such that $\frac{\sigma}{16} < \frac{\ell^* \cdot \mathbf{I}(f)}{n} \leq \frac{\sigma}{8}$. Let A' and B' be the subset of A and B that are ℓ^*-sticky. Let $E' \subseteq E$ be the edges with end points in A' and B'. Note that any edge of $E' \subseteq F_{\ell^*}$. By Lemma 6, we get that the fraction of non-ℓ^*-sticky vertices is at most $2\ell^* \cdot \mathbf{I}(f)/n \leq \sigma/4$. Since the degree of any vertex in G in Lemma 2 is $\leq 2d$,

$$|F_{\ell^*}| \geq |E'| \geq |E| - (2d) \cdot \frac{\sigma \cdot 2^n}{4} = \frac{\sigma d 2^n}{2}.$$

The probability the algorithm chooses $\ell = \ell^*$ is $1/\log n$. Lemma 7 gives us

$$\begin{aligned}
\Pr[\text{Algorithm rejects}] &\geq \frac{1}{\log n} \cdot \Pr[\text{Algorithm rejects} | \ell = \ell^*] \\
&\geq \frac{1}{\log n} \cdot \left(\frac{\ell^*}{n} \cdot \frac{|F_{\ell^*}|}{2^n}\right) && \text{(by Lemma 7)} \\
&\geq \frac{1}{\log n} \cdot \frac{\ell^*}{n} \cdot \frac{\sigma d}{2} \\
&\geq \frac{1}{\mathbf{I}(f)} \cdot \left(\frac{\sigma^2 d}{32 \log n}\right) && \text{(plugging } \ell^* \geq \sigma n/(16 \cdot \mathbf{I}(f))\text{)} \\
&= \Omega\left(\frac{\varepsilon^2}{\mathbf{I}(f)\log^5 n}\right) && \text{(by Lemma 2)} \quad ◀
\end{aligned}$$

References

1 Aleksandrs Belovs and Eric Blais. A Polynomial Lower Bound for Testing Monotonicity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2016.

2 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ Monotonicity Tester for Boolean Functions over the Hypercube. *SIAM Journal on Computing (SICOMP)*, 45(2):461–472, 2014.

3 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean Function Monotonicity Testing Requires (Almost) $O(n^{1/2})$ Non-adaptive Queries. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2015.

4 Xi Chen, Rocco A. Servedio, and Li-Yang. Tan. New Algorithms and Lower Bounds for Monotonicity Testing. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.

5 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand: New Lower Bounds for Testing Monotonicity and Unateness. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2017.

6 Yevgeny Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved Testing Algorithms for Monotonicity. *Proceedings, International Workshop on Randomization and Computation (RANDOM)*, 1999.

7 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, and Ronitt Rubinfeld. Monotonicity Testing over General Poset Domains. *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2002.

8 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samordinsky. Testing Monotonicity. *Combinatorica*, 20:301–337, 2000.

9 Subhash Khot, Dor Minzer, and Muli Safra. On Monotonicity Testing and Boolean Isoperimetric Type Theorems. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.

10 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Parameterized Property Testing of Functions. *ACM Transactions on Computation Theory (TOCT)*, 9(4), 2017.

11 Sofya Raskhodnikova. Monotonicity Testing. *Masters Thesis, MIT*, 1999.

12 Michel Talagrand. Isoperimetry, Logarithmic Sobolev inequalities on the Discrete Cube, and Margulis' Graph Connectivity Theorem. *Geom. Func. Anal.*, 3(3):295–314, 1993.