Adaptive Lower Bound for Testing Monotonicity on the Line

Aleksandrs Belovs

Abstract

In the property testing model, the task is to distinguish objects possessing some property from the objects that are far from it. One of such properties is monotonicity, when the objects are functions from one poset to another. It is an active area of research.

Recently, Pallavoor, Raskhodnikova and Varma (ITCS’17) proposed an $\varepsilon$-tester for monotonicity of a function $f : [n] \rightarrow [r]$, whose complexity depends on the size of the range as $O\left(\frac{\log r}{\varepsilon^2}\right)$. In this paper, we prove a nearly matching lower bound of $\Omega\left(\frac{\log r \log \log r}{\varepsilon^2}\right)$ for adaptive two-sided testers. Additionally, we give an alternative proof of the $\Omega\left(\varepsilon^{-d} \log n - \varepsilon^{-d} \log \varepsilon^{-1}\right)$ lower bound for testing monotonicity on the hypergrid $[n]^d$ due to Chakrabarty and Se-shadhri (RANDOM’13).

1 Introduction

The framework of property testing was formulated by Rubinfeld and Sudan [19] and Goldreich et al. [16]. A property testing problem is specified by a property $\mathcal{P}$, which is a class of function mapping some finite set $D$ into some finite set $R$, and a real number $0 < \varepsilon < 1$. An $\varepsilon$-tester is a bounded-error randomised query algorithm which, given oracle access to a function $f : D \rightarrow R$, distinguishes between the case when $f$ belongs to $\mathcal{P}$ and the case when $f$ is $\varepsilon$-far from $\mathcal{P}$. The latter means that any function $g \in \mathcal{P}$ differs from $f$ on at least $\varepsilon$ fraction of the points in the domain $D$. The complexity measure is the number of queries to the function $f$. A tester is with 1-sided error if it always accepts $f \in \mathcal{P}$. A tester is non-adaptive if its queries do not depend on the responses received to previous queries. The most general tester is adaptive with 2-sided error, which we will implicitly assume in this paper.

When both the domain $D$ and the range $R$ are partially ordered sets, a natural property to consider is that of monotonicity. A function $f : D \rightarrow R$ is called monotone if $x \leq y$ implies $f(x) \leq f(y)$ for all $x, y \in D$. In this case, the property $\mathcal{P}$ consists of all monotone functions from $D$ to $R$. The problem of testing monotonicity was first studied by Goldreich et al. [15] in the case when $D$ is the Boolean hypercube $\{0, 1\}^d$ and $R = \{0, 1\}$. This is an active area of research as exemplified by the papers [11, 14, 6, 5, 8, 17, 1, 2, 10].

In this paper, however, we study an arguably simpler problem of testing monotonicity on the line. The line in this context is a totally ordered set $[n] = \{0, 1, \ldots, n - 1\}$, and we are interested in the functions from $[n]$ to $[r]$ for some positive integers $n$ and $r$. A related problem is that of testing monotonicity on the hypergrid. A hypergrid is a set $[n]^d$ of $d$-tuples with elements in $[n]$. For two $d$-tuples $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$, we have $x \leq y$ iff $x_i \leq y_i$ for all $i$. We are interested in functions from $[n]^d$ to $[r]$ for some positive integers $n, d$ and $r$. Clearly, this is a generalisation of both monotonicity testing on the line (when $d = 1$) and on the hypercube (when $n = 2$). These problems are closely related, so it is important to consider all of them when discussing prior work.

*Faculty of Computing, University of Latvia. Raina bulvaris 19, Riga, Latvia.
Let us start with the upper bounds. The problem of testing monotonicity on the line was first considered by Ergün et al. [12], who gave an ε-tester with complexity \( O(\varepsilon^{-1} \log n) \). Concerning the hypercubes \( \{0,1\}^d \), Goldreich et al. [15] proposed the edge tester, and Chakrabarty and Seshadhri [6] proved that it has query complexity \( O(d/\varepsilon) \). In the latter paper, an ε-tester for testing monotonicity on the hypergrid \([n]^d\) with complexity \( O(\varepsilon^{-1} \log \log n) \) was also constructed.

Now let us turn to the lower bounds. Ergün et al. [12] proved a lower bound of \( \Omega(\log n) \) for testing monotonicity on the line in the so-called comparison-based model. In this model, each query of the tester may depend only on the order relations between the responses to the previous queries, but not on the values of the responses themselves. Fischer [13] proved that any lower bound for a monotonicity testing problem in the comparison-based model can be converted into a lower bound in the usual value-based model. This immediately gives an \( \Omega(\log n) \) lower bound for testing monotonicity on the line, matching the upper bound by Ergün et al. [12]. Chakrabarty and Seshadhri [7] used the same technique to prove a lower bound of \( \Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1}) \) for testing monotonicity on hypergrids. Unfortunately, Fischer’s construction is based on Ramsey theory, which means that, in order for this construction to work, the size of the range, \( r \), has to be really huge.

Extreme values of \( r \) in the above lower bounds motivated Pallavoor et al. [18] to analyse the complexity of monotonicity testing with respect to the size of the range, \( r \), rather than the size of the domain, \( n \), as it was done previously. Let us briefly summarise what is known in this regard. It is easy to see that if \( r = 2 \), then \( O(1/\varepsilon) \) queries suffice to ε-test monotonicity on the line. Using similar observations, Pallavoor et al. constructed an ε-tester for monotonicity on the line with complexity \( O(\frac{1}{\varepsilon} \log r) \) for general \( r \), as well as an ε-tester for monotonicity on hypergrids with complexity \( O(\frac{d}{\varepsilon} \log \frac{d}{\varepsilon} \log r) \).

The only prior technique capable of proving strong lower bounds for functions with small range is that of communication complexity. Blais et al. introduced this technique in [3], where it was proven that \( \Omega(\min\{d,r^2\}) \) queries are required to test a function \( f: \{0,1\}^d \rightarrow [r] \) for monotonicity. In a subsequent paper [4] a non-adaptive lower bound of \( \Omega(d \log n) \) was proven for functions \( f: [n]^d \rightarrow [nd] \) on the hypergrid.

Our main contribution is the following adaptive lower bound for testing monotonicity on the line:

**Theorem 1.** Every adaptive bounded-error 1/2-tester for monotonicity of a function \( f: [2^k] \rightarrow [k^{3k}] \) has query complexity \( \Omega(k) \).

Unlike [13] [7], we bypass Ramsey’s theory and construct two explicit distributions of functions that are hard to distinguish by an adaptive algorithm. This gives the same lower bound of \( \Omega(\log n) \) as in Fischer’s result [13], but with vastly reduced range size. A more direct construction can be beneficial for generalisation to other models, like quantum testers, since it is not known how to quantise Fischer’s technique.

In terms on the size of the range, \( r \), this gives a lower bound of \( \Omega(\frac{\log r}{\log \log r}) \), thus nearly matching the upper bound by Pallavoor et al. [18]. Finally, we get a slightly worse estimate (in terms of the range size) than that of Blais et al. [4] but for adaptive testers.

We prove Theorem 1 in Section 3. If Section 4, we show how our construction can be used to obtain the lower bound \( \Omega(\varepsilon^{-1} d \log n - \varepsilon^{-1} \log \varepsilon^{-1}) \) of Chakrabarty and Seshadhri [7] but with greatly reduced range size.

\(^1\) Note that this does not mean that the tester asks queries of the form \( f(x) \leq f(y) \). The query is still an input \( x \), and the tester learns about the order relations between \( f(x) \) and \( f(y) \) for all previously queried \( y \)'s.
2 Preliminaries

It will be convenient for us to assume that \([n] = \{0, 1, \ldots, n - 1\}\). Also, \([a..b]\) stands for \(\{a, a+1, \ldots, b-1\}\).

An assignment \(\alpha: S \rightarrow [r]\) is a function defined on a subset \(S \subseteq [n]\). The weight of \(\alpha\) is the size of \(S\). We say that \(f\) agrees with \(\alpha\) is \(f(x) = \alpha(x)\) for all \(x \in S\). This is notated by \(f \sim \alpha\).

3 Proof of Theorem 1

We will define a probability distribution \(\mu\) on monotone functions \(f: [2^k] \rightarrow [k^3]\) and a probability distribution \(\nu\) on functions \(g: [2^k] \rightarrow [k^3]\) that are \(1/2\)-far from monotone. It will be impossible to distinguish these two distributions using fewer than \(\Omega(k)\) queries. Let \(m = k^3\), so that \(r = m^k\).

The distribution \(\mu\) is defined as the last member in an inductively-defined family \(\mu_0, \mu_1, \ldots, \mu_k\) of distributions, where \(\mu_i\) is supported on functions \([2^i] \rightarrow [m^i]\). The distribution \(\mu_0\) is supported on the only function that maps 0 to 0. Assume that \(\mu_i\) is already defined and let us define \(\mu_{i+1}\). In order to do that, we independently sample \(f_0\) and \(f_1\) from \(\mu_i\) and \(a\) from \([m-1]\).

The corresponding function \(f\) in \(\mu_{i+1}\) is given by

\[
\begin{aligned}
f(x) &= \begin{cases}
a \cdot m^i + f_0(x), & \text{if } 0 \leq x < 2^i; \\
(a+1)m^i + f_1(x-2^i), & \text{if } 2^i \leq x < 2^{i+1}.
\end{cases}
\end{aligned}
\]

For an alternative way of defining this distribution, let us assume that the argument \(x\) is written in binary and the value \(f(x)\) in \(m\)-ary. We prepend leading zeroes if necessary so that each number has exactly \(k\) digits. We enumerate the digits from left to right with the elements of \([k]\), so that the 0-th digit is the most significant one, and the \((k-1)\)-st digit is the least significant one. For each binary string \(s\) of length strictly less than \(k\), sample an element \(a_s\) from \([m-1]\) independently and uniformly at random. The \(i\)-th digit of \(f(x)\) is defined as \(a_s + b\), where \(s\) is the prefix of \(x\) of length \(i\) and \(b\) is the \(i\)-th bit of \(x\). It is easy to see that any function \(f\) from the support of \(\mu\) is monotone.

The distribution \(\nu\) is defined as the uniform mixture of the distributions \(\nu^j\) as \(j \in [k]\). The function \(g \sim \nu^j\) is defined as \(g(x) = f(x \oplus 2^{k-1-j})\) when \(f\) is sampled from \(\mu\). Here \(\oplus\) denotes the bit-wise XOR function. In other words, the \(j\)th bit of the argument is flipped before applying \(f\). Alternatively, we may say that \(g \sim \nu^j\) is defined as in the case of \(\mu\) with the exception that the \(j\)-th digit of \(g(x)\) is \(a_s + (1-b)\) instead of \(a_s + b\).

Claim 2. Any function \(g\) in the support of \(\nu^j\) is \(1/2\)-far from monotone.

Proof. Consider two input strings \(x < y\) that differ only in the \(j\)-th bit. By the definition of \(\nu^j\) we have \(g(x) > g(y)\), thus, \(\{x, y\}\) is a monotonicity-violating pair. We have \(2^{k-1}\) such disjoint pairs, and every monotone function differs from \(g\) on at least one element of each pair.

Now we are ready to start with the proof of Theorem 1. Assume towards contradiction that there exists a randomised \(1/2\)-tester \(A\) with query complexity \(o(k)\). Using standard error reduction, we may assume that \(A\) errs with probability at most \(1/8\) on each input. Let \(\lambda\) be the uniform mixture of \(\mu\) and \(\nu\). Clearly, \(A\) errs with probability at most \(1/8\) on \(\lambda\). Using a standard argument, there exists a deterministic query algorithm \(A\) with the same query complexity, which errs with probability at most \(1/8\) on \(\lambda\). This means that

\[
\Pr_{f \sim \mu}[A\ accepts\ f] \geq \frac{3}{4} \quad \text{and} \quad \Pr_{g \sim \nu}[A\ accepts\ g] \leq \frac{1}{4}.
\]
Also, we may assume that \( k \) is large enough so that the query complexity of the deterministic query algorithm \( A \) is less than \( k/2 \).

The leaves of \( A \) are specified by assignments in the sense that \( A \) terminates its work on input \( f \) in a leaf given by assignment \( \alpha \) if and only if \( f \) agrees with \( \alpha \). Moreover, the weight of each such \( \alpha \) does not exceed the query complexity of \( A \). We partition all these assignments \( \alpha \) into two parts as follows. We say that an integer \( \ell \in \mathbb{[}r] \) is good if it contains no digit 0 and no digit \( m-1 \) (in the \( m \)-ary representation as before). Otherwise, \( \ell \) is bad. We say that an assignment \( \alpha \) is good if all the elements in its range are good. Otherwise, \( \alpha \) is bad. Finally, a leaf of \( A \) is good iff the corresponding assignment is good.

**Claim 3.** The probability that \( A \) ends its work in a bad leaf when run on an input \( f \) sampled from \( \mu \) is \( o(1) \).

**Proof.** For each \( x \) in the domain of \( f \), the value \( f(x) \) is bad only if one of the \( k \) elements \( a_{s_0}, a_{s_1}, \ldots, a_{s_{k-1}} \) has value 0 or \( m-2 \), where \( s_i \) is the prefix of \( x \) of length \( i \). Thus, the probability of \( A \) to terminate in a bad leaf is at most the probability of finding an element \( a_s \) with value in \( \{0, m-2\} \) with \( k/2 \) queries, where on each query it is allowed to test the values of \( k \) different \( a_s \)'s. Since the \( a_s \)'s are independent, and the probability of \( a_s \in \{0, m-2\} \) is \( 2/(m-1) \), we get, using the standard bound on search, that the probability of succeeding is at most \( k/2 \cdot k \cdot 2/(m-1) = o(1) \). \( \square \)

**Claim 4.** For each good assignment \( \alpha \) of weight at most \( k/2 \),

\[
\Pr_{f \sim \mu} [f \sim \alpha] \leq 2 \cdot \Pr_{g \sim \nu} [g \sim \alpha].
\]

Before we start with the proof of this claim, let us show how Theorem 1 follows from Claims 3 and 4. In the following, let \( C \) be the set of assignments which correspond to the accepting leaves of \( A \), let \( B \subseteq C \) be the subset of bad assignments, and \( G \subseteq C \) be the subset of good assignments. Then,

\[
\Pr_{f \sim \mu} [A \text{ accepts } f] = \sum_{\alpha \in B} \Pr_{f \sim \mu} [f \sim \alpha] + \sum_{\alpha \in G} \Pr_{f \sim \mu} [f \sim \alpha] \\
\leq o(1) + 2 \sum_{\alpha \in G} \Pr_{g \sim \nu} [g \sim \alpha] \leq o(1) + 2 \Pr_{g \sim \nu} [A \text{ accepts } g],
\]

which is in contradiction with (2) if \( k \) is large enough.

It remains to prove Claim 4. Consider a good assignment \( \alpha \) of weight at most \( k/2 \). Let \( S \) be the domain of \( \alpha \). We say that a pair \( x < y \) from \( S \) cuts an index \( j \in [k] \) iff their first \( j \) bits agree, and they disagree in the \( j \)-th bit. In other words, there exists \( a \in \mathbb{Z} \) such that

\[
2a \cdot 2^{k-j-1} \leq x < (2a + 1) \cdot 2^{k-j-1} \leq y < (2a + 2) \cdot 2^{k-j-1}.
\]

The assignment \( \alpha \) cuts all the indices cut by the pairs in \( S \). Claim 4 follows from the following two lemmata.

**Lemma 5.** If a good assignment \( \alpha \) does not cut an index \( j \), then

\[
\Pr_{f \sim \mu} [f \sim \alpha] = \Pr_{g \sim \nu} [g \sim \alpha].
\]
Proof. By induction on \(i\) in the definition \(\mu_i\) of \(\mu_i\). Let for brevity \(j' = k - j - 1\). We can define \(\nu_i'\) from \(\mu_i\) as \(g(x) = f(x \oplus 2^{j'})\) when \(f\) is sampled from \(\mu_i\) if \(j' < i\), and as \(\mu_i\) otherwise. We prove that

\[
\Pr_{f \sim \mu_i} [f \sim \alpha] = \Pr_{g \sim \nu_i'} [g \sim \alpha]
\]

for every good assignment \(\alpha\) from \([2^i]\) to \([m^i]\) that does not cut the index \(j\).

The base case \(i = 0\) is trivial. Assume \((\ref{eq:induct})\) is proven for \(i\), and let us prove it for \(i + 1\). Let \(S\) be the domain of \(\alpha\). There are two cases.

First, assume both \(S \cap [2^i]\) and \(S \cap [2^i..2^{i+1}]\) are non-empty. This means that \(\alpha\) cuts \(k - i - 1\), hence, \(i \neq j'\). Also, we may assume there exists \(1 \leq a \leq m - 3\) such that \(\alpha([2^i]) \subseteq [am^i..(a+1)m^i]\) and \(\alpha([2^i..2^{i+1}]) \subseteq [(a+1)m^i..(a+2)m^i]\), since otherwise both sides of \((\ref{eq:induct})\) are 0. Under these assumptions,

\[
\Pr_{f \sim \mu_{i+1}} [f \sim \alpha] = \frac{1}{m-1} \Pr_{f_0 \sim \mu_i} [f_0 \sim \alpha_0] \Pr_{f_1 \sim \mu_i} [f_1 \sim \alpha_1],
\]

where \(f_0\) and \(f_1\) are obtained reversely from \((\ref{eq:mu})\), and \(\alpha_0\) and \(\alpha_1\) are defined similarly: \(\alpha_0(i) = \alpha(i) \mod m^i\) and \(\alpha_1(i) = \alpha(i+2^i) \mod m^i\) for all \(i \in [2^i]\). Both assignments are good, and they do not cut the index \(j\). Similarly, since \(i \neq j'\):

\[
\Pr_{g \sim \nu_{i+1}} [g \sim \alpha] = \frac{1}{m-1} \Pr_{g_0 \sim \nu_i} [g_0 \sim \alpha_0] \Pr_{g_1 \sim \nu_i} [g_1 \sim \alpha_1].
\]

By the inductive assumption, we have the required equality.

Now assume one of \(S \cap [2^i]\) and \(S \cap [2^i..2^{i+1}]\) is empty. We consider the case \(S \subseteq [2^i]\), the second one being similar. Again, we can assume there exists \(1 \leq a \leq m - 2\) such that \(\alpha([2^i]) \subseteq [am^i..(a+1)m^i]\). Then,

\[
\Pr_{f \sim \mu_{i+1}} [f \sim \alpha] = \frac{1}{m-1} \Pr_{f_0 \sim \mu_i} [f_0 \sim \alpha_0]
\]

and, no matter whether \(i = j'\) or not,

\[
\Pr_{g \sim \nu_{i+1}} [g \sim \alpha] = \frac{1}{m-1} \Pr_{g_0 \sim \nu_i} [g_0 \sim \alpha_0],
\]

and again we have the required equality by the inductive assumption.

Lemma 6. An assignment \(\alpha\) of weight \(t\) cuts at most \(t - 1\) indices in \([k]\).

Proof. Let \(S\) be the domain of \(\alpha\), and let \(J \subseteq [k]\) be the set of indices that \(\alpha\) cuts.

Let us construct a graph \(G\) as follows. Its vertex set is \(S\). For every index \(j \in J\) take one arbitrary pair of elements \(x, y \in S\) that cuts \(j\) and connect \(x\) and \(y\) by an edge. We say that the edge \(xy\) cuts \(j\).

We claim that the graph \(G\) is acyclic, from which the statement of the lemma follows. Assume that \(G\) contains a simple cycle. Consider an edge \(xy\) that cuts the minimal index \(j\) on this cycle. Then \(x\) and \(y\) disagree in the \(j\)-th bit. On the other hand, considering the remaining part of the cycle, we see that \(x\) and \(y\) agree in the \(j\)-th bit. A contradiction, hence, \(G\) is acyclic.

By Lemma \(4\) there are at least \(k/2\) indices not cut by \(\alpha\), and using Lemma \(3\) we have

\[
2 \cdot \Pr_{g \sim \nu} [g \sim \alpha] \geq \frac{2}{k} \sum_{\text{j-\alpha does not cut } j} \Pr_{g \sim \nu^j} [g \sim \alpha] \geq \frac{2}{k} \cdot \frac{k}{2} \Pr_{f \sim \mu} [f \sim \alpha] = \Pr_{f \sim \mu} [f \sim \alpha],
\]

proving Claim \(4\).
4 Some Easy Extensions

In this section, we briefly describe how the result of Theorem 1 can be extended to include dependence on $\varepsilon$ and hypergrids. We start with dependence on $\varepsilon$.

Theorem 7. The complexity of $\varepsilon$-testing a function $f: [n] \rightarrow [r]$ for monotonicity is $\Omega\left(\frac{\log(nr)}{\varepsilon^2}\right)$ and $\Omega\left(\frac{\log(nr)}{\varepsilon^2 \log\log(nr)}\right)$.

Proof. Let $\ell$ be a positive integer and assume $\varepsilon = 1/(2\ell)$. We will construct probability distributions $\tilde{\mu}$ and $\tilde{\nu}$ on functions $f: [\ell^2k] \rightarrow [\ell^3k]$ such that all functions in the support of $\tilde{\mu}$ are monotone, functions in the support of $\tilde{\nu}$ are $\varepsilon$-far from monotone, and it takes $\Omega(\ell k)$ queries to distinguish $\tilde{\mu}$ and $\tilde{\nu}$.

Let $\mu$ and $\nu$ be as in Section 3. For $s \in [\ell]$, independently sample $f_s$ from $\mu$. Define $f \sim \tilde{\mu}$ as

$$f(s \cdot 2^k + x) = s \cdot k^3 + f_s(x)$$

(4)

for all $x \in [2^k]$ and $s \in [\ell]$. The distribution $\tilde{\nu}$ is defined as a uniform mixture of $\nu^{t,j}$ as $t$ ranges over $[\ell]$ and $j$ over $[k]$. The corresponding function $f \sim \tilde{\nu}^{t,j}$ is defined as in (4) with exception that $f_t$ is sampled from $\nu^{t,j}$ instead of $\mu$. It is easy to see that functions in the support of $\tilde{\mu}$ are monotone, and, using Claim 2, that functions in the support of $\tilde{\nu}$ are $\varepsilon$-far from monotone.

Informally, it takes $\Omega(\ell k)$ queries to distinguish $\tilde{\mu}$ and $\tilde{\nu}$ because we are searching for one non-monotone distribution $\nu^{t,j}$ among $\ell$ independent distributions. Formally, we may proceed as follows. Assume towards contradiction that there exists a deterministic query algorithm $A$ that makes less than $\ell k/4$ queries, accepts $\tilde{\mu}$ with probability at least $3/4$ and accepts $\tilde{\nu}$ with probability at most $1/4$.

Using the same reasoning as in Claim 3, the expected number of bad elements found by $A$ when run on $\tilde{\mu}$ is $O(\ell k^2/m) = o(\ell)$. We call an assignment $\alpha$ on $[\ell^2k]$ bad if it has more than $\ell/4$ bad elements in its range. Otherwise, we call $\alpha$ good. By Markov’s inequality, the probability $A$ terminates in a bad assignment when executed on $\tilde{\mu}$ is $o(1)$.

Now consider a good assignment $\alpha$ of weight at most $\ell k/4$. It corresponds to $\ell$ sub-assignments $\alpha_s$ on $[2^k]$ defined by $\alpha_s(x) = \alpha(s \cdot 2^k + x)$. Using Claim 4 we get that

$$\Pr_{f \sim \tilde{\mu}}[f \sim \alpha] = \Pr_{g \sim \nu^{t,j}}[g \sim \alpha]$$

(5)

if the sub-assignment $\alpha_t$ is good and does not cut $j$. Using that there are at most $\ell/4$ bad $\alpha_s$ and Lemma 6 we have that there are at least $\ell k/2$ pairs $(t,j)$ satisfying (5). Hence,

$$\Pr_{f \sim \tilde{\mu}}[f \sim \alpha] \leq 2 \cdot \Pr_{g \sim \nu^{t,j}}[g \sim \alpha].$$

Now we finish the proof as in Section 3.

Theorem 8. The complexity of $\varepsilon$-testing a function $f: [n]^d \rightarrow [r]$ for monotonicity is $\Omega(\varepsilon^{-1}d \log n - \varepsilon^{-1} \log \varepsilon^{-1})$ and $\Omega\left(\frac{\log(nr)}{\varepsilon^2 \log\log(nr)}\right)$.

Proof. Consider the functions defined in the proof of Theorem 7. Assume that $\ell = 2^a$ for some integer $a$, and choose $k$ so that $a + k = db$ for some integer $b$. We consider the domain of the input functions $[\ell^2k] = [2^{a+k}]$ as $[2^b]$, where we break the binary representation of $x \in [2^{a+k}]$ into $d$ groups of $b$ bits.

If a function is monotone on $[2^{a+k}]$, it is still monotone when considered on $[2^b]$. Also, the analysis of the algorithm does not involve the order relation defined on the domain of the input functions. The only thing that might go wrong is that the functions in the support of $\tilde{\nu}$ are no
longer $\varepsilon$-far from monotone when considered on $[2^b]^d$. But this does not happen. Indeed, in the proof of Claim 2 the monotonicity-violating pairs $(x, y)$ differ in exactly one bit. So if $x < y$ in $[2^{a+k}]$, this is still true in $[2^b]^d$, hence the proof of Claim 2 carries over. Thus, all the functions in the support of $\tilde{\nu}$ are $\varepsilon$-far from monotone. The statement of the theorem now follows from the proof of Theorem 7.

Acknowledgements

I am grateful to Eric Blais for introducing me to this problem, and for his encouragement to work on it, as well as for pointing out Ref. [18]. I would also like to thank Dmitry Gavinsky, Ansis Rosmanis, and Ronald de Wolf for helpful discussions. The construction of Theorem 7 is due to Ronald de Wolf.

This research is supported by the ERDF project number 1.1.1.2/1/16/113. Part of this work was done while visiting Institute of Mathematics of the Czech Academy of Sciences in Prague, and Centre for Quantum Technologies in Singapore. I would like to thank Dmitry Gavinsky, Pavel Pudlák, and Miklos Santha for hospitality.

References

[1] A. Belovs and E. Blais. Quantum algorithm for monotonicity testing on the hypercube. *Theory of Computing*, 11(16): 403–412, 2015. arXiv:1503.02868.
[2] A. Belovs and E. Blais. A polynomial lower bound for testing monotonicity. In *Proc. of 48th ACM STOC*, pages 1021–1032, 2016. arXiv:1511.05053.
[3] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012. Earlier: CCC’11, ECCC:2011/045.
[4] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proc. of 29th IEEE CCC*, pages 309–320, 2014.
[5] D. Chakrabarty and C. Seshadhri. A $o(n)$ monotonicity tester for Boolean functions over the hypercube. In *Proc. of 45th ACM STOC*, pages 411–418, 2013. arXiv:1302.4536.
[6] D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proc. of 45th ACM STOC*, pages 419–428, 2013. arXiv:1204.0849.
[7] D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014. Earlier: RANDOM’13, ECCC:2013/062.
[8] X. Chen, A. De, R. A. Servedio, and L.-Y. Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proc. of 47th ACM STOC*, pages 519–528, 2015. arXiv:1412.5657.
[9] X. Chen, R. A. Servedio, and L.-Y. Tan. New algorithms and lower bounds for monotonicity testing. In *Proc. of 55th IEEE FOCS*, pages 286–295, 2014. arXiv:1412.5655.
[10] X. Chen, E. Waingarten, and J. Xie. Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proc. of 49th ACM STOC*, pages 523–536, 2017. arXiv:1702.06997.
[11] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proc. of 3rd RANDOM*, pages 97–108 Springer, 1999. ECCC:1999/017.
[12] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000. Earlier: STOC’95.
[13] E. Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.
[14] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In Proc. of 34th ACM STOC, pages 474–483, 2002.

[15] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. Combinatorica, 20(3):301–337, 2000. Earlier: [FOCS’98]

[16] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. Journal of the ACM, 45(4):653–750, 1998. Earlier: [FOCS’96]

[17] S. Khot, D. Minzer, and M. Safra. On monotonicity testing and Boolean isoperimetric type theorems. In Proc. of 56th IEEE FOCS, pages 52–58, 2015. [ECCC:2015/011]

[18] R. K. S. Pallavoor, S. Raskhodnikova, and N. Varma. Parameterized property testing of functions. In Proc. of 8th ACM ITCS, volume 67 of LIPIcs, page 12. Dagstuhl, 2017.

[19] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. SIAM Journal on Computing, 25(2):232–271, 1996.