THEORY OF COMPUTING, Volume 11 (16), 2015, pp. 403–412
www.theoryofcomputing.org

NOTE

Quantum Algorithm for Monotonicity Testing on the Hypercube

Aleksandrs Belovs* Eric Blais†

Received March 24, 2015; Revised July 16, 2015; Published December 29, 2015

Abstract: In this note, we develop a bounded-error quantum algorithm that makes $\tilde{O}(n^{1/4}\varepsilon^{-1/2})$ queries to a function $f\colon \{0,1\}^n \to \{0,1\}$, accepts when f is monotone, and rejects when f is ε-far from being monotone. This result gives a super-quadratic improvement compared to the best known randomized algorithm for all $\varepsilon = o(1)$. The improvement is cubic when $\varepsilon = 1/\sqrt{n}$.

ACM Classification: F.1.1, F.1.2

AMS Classification: 68Q12, 81P68

Key words and phrases: Boolean functions, quantum query complexity, property testing, monotonicity

1 Introduction

The problem of testing whether a Boolean function is monotone is one of the fundamental—and most extensively studied—problems in property testing. Let $\preceq$ denote the bitwise partial order on the Boolean hypercube $\{0,1\}^n$, i. e., $x \preceq y$ iff $x_j \le y_j$ for all $j \in [n]$. The function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* iff $f(x) \le f(y)$ for all $x \preceq y$, and it is ε*-far from monotone* if it cannot be made monotone by changing

*A. B. is supported by FP7 FET Proactive project QALGO. Part of this work was completed while A. B. was at MIT, supported by Scott Aaronson's Alan T. Waterman Award from the National Science Foundation. A. B. thanks Andris Ambainis and Ronald de Wolf for their comments on the previous versions of this note.

†E. B. is funded by an NSERC Discovery Grant. Part of this work was completed while E. B. was a Simons Postdoctoral Fellow at MIT.

 DOI: 10.4086/toc.2015.v011a016

its value on at most an ε fraction of the inputs. An *ε-tester for monotonicity* is a randomized algorithm that distinguishes monotone functions from those that are ε-far from monotone with large (say, $2/3$) probability. Determining the minimum number of queries required to test monotonicity of Boolean functions was the first problem considered in the context of testing combinatorial (as opposed to algebraic) properties of Boolean functions [13] and, despite extensive study (see for example the recent papers [9, 11, 10, 16] and the references therein), it has still not been completely resolved.

Goldreich et al. [13] initiated the study of the monotonicity testing problem and showed that it is possible to ε-test the monotonicity of a function $f\colon \{0,1\}^n \to \{0,1\}$ with $O(n/\varepsilon)$ queries to f. Their algorithm is called the *edge tester*, and it is very simple: Repeatedly sample the edges of the hypercube—i. e., pairs of elements $x \preceq y \in \{0,1\}^n$ for which there is exactly one coordinate $i \in [n]$ such that $x_i \neq y_i$—uniformly at random and verify that f satisfies the monotonicity condition on the endpoints of each edge. The anti-dictator function, defined by $f(x) = 1 - x_i$ for some $i \in [n]$, shows that the analysis of this algorithm is tight.

The edge tester remained the most efficient monotonicity testing algorithm for more than a decade, until Chakrabarti and Seshadhri [9] introduced a new ε-tester for monotonicity that requires only $\tilde{O}(n^{7/8}\varepsilon^{-3/2})$ queries. (Here and afterwards $\tilde{O}$ hides terms polylogarithmic in n and $1/\varepsilon$.) The Chakrabarti–Seshadhri algorithm, like the edge tester, is a *pair tester*. A pair tester is any algorithm of the following form: It repeatedly samples from some predefined probability distribution on the pairs of the inputs $x \preceq y$ from the hypercube. If the pair violates monotonicity, the tester rejects. If no violation was found after sufficiently many samples, the tester accepts. Unlike the edge tester, however, the Chakrabarti–Seshadhri algorithm selects pairs of inputs that have Hamming distance up to $O(\sqrt{n})$. The same high-level approach has since been used by Chen, Servedio, and Tan [11] to obtain a $\tilde{O}(n^{5/6}\varepsilon^{-4})$-query ε-tester for monotonicity and, very recently, by Khot, Minzer, and Safra in their beautiful paper [16] which shows that $\tilde{O}(\sqrt{n}/\varepsilon^2)$ queries suffice to ε-test monotonicity.

In summary, research on testing monotonicity of Boolean functions has led to two incomparable ε-testers: the edge tester with query complexity $O(n/\varepsilon)$, and the Khot–Minzer–Safra algorithm with query complexity $\tilde{O}(\sqrt{n}/\varepsilon^2)$. The first one has better dependence on ε, the second one on n. Furthermore, these two algorithms—along with every other algorithm that has been proposed for testing monotonicity of Boolean functions—are pair testers. Let us also note that pair testers are *non-adaptive* algorithms (they can select all their queries in advance) and have *one-sided error* (they always accept monotone functions).

In this paper, we consider the query complexity of *quantum* algorithms that test the monotonicity of Boolean functions. See [19] for a recent survey on quantum property testing. While the quantum query complexity of the monotonicity testing problem has not been explicitly studied before, we can apply quantum amplitude amplification [6] to obtain a quadratic improvement on the query complexity of any pair tester. As a result, the edge tester and the Khot–Minzer–Safra algorithm imply that we can ε-test monotonicity with $O(\sqrt{n/\varepsilon})$ and $\tilde{O}(n^{1/4}/\varepsilon)$ quantum queries, respectively. Our main result is a simple quantum algorithm that combines the best dependence from both of these algorithms.

Theorem 1.1. *It is possible to ε-test monotonicity of a function $f\colon \{0,1\}^n \to \{0,1\}$ with*

$$O\Big(\frac{1}{\sqrt{\varepsilon}} \cdot n^{1/4} \log n\Big)$$

quantum queries.

Let us compare this result with the known lower bounds on classical algorithms. Lower bounds for this problem are notoriously hard. For many years, the best known lower bound on the *non-adaptive* randomized query complexity was $\Omega(\log n)$ for constant ε by Fischer et al. [12]. Recently, it was improved by Chen et al. [11, 10] to almost $\Omega(\sqrt{n})$, essentially matching the query complexity of the Khot–Minzer–Safra algorithm. The best known *adaptive* lower bound is only $\Omega(\log n)$, which immediately follows from the non-adaptive lower bound.

For pair testers, more lower bounds are known. First, Briët et al. [7] proved that any pair tester with query complexity of the form $O(\alpha(n)/\varepsilon)$ must have $\alpha(n) = \Omega(n/\log n)$. Also, Khot, Minzer, and Safra [16] gave an example of a family of functions that are at distance $\Theta(1/\sqrt{n})$ to being monotone, but for which any pair tester needs $\Omega(n^{3/2})$ queries to find a non-monotone pair with constant probability. This matches the performance of both the edge tester and the Khot–Minzer–Safra algorithm on this instance. Our algorithm, on the other hand, needs only $\tilde{O}(\sqrt{n})$ queries, which constitutes a cubic improvement to both algorithms.

This is an interesting development, as very few super-quadratic but still polynomial quantum speed-ups are known. We can only mention a cubic speed-up for exponential congruences by van Dam and Shparlinski [24], and quartic speed-ups for finding counterfeit coins by Iwama et al. [15], and learning the "exactly-half junta" by Belovs [4]. There is also a recent algorithm by Ambainis et al. [1] for the gapped version of group testing with up to a quartic improvement.

Our algorithm is based on a technical result from [16] and the (dual) adversary bound. The technical result, Lemma 3.1 below, is the cornerstone of the analysis of the Khot–Minzer–Safra algorithm, and the main part of [16] is devoted to derivation of this lemma. With the lemma in hand, the remaining analysis of their algorithm is relatively short. Interestingly, we need the same technical result and our analysis is also quite short, but we use this result in a very different way.

The adversary bound characterizes quantum query complexity up to a constant factor, as shown by Reichardt et al. [21, 18]. Previously, the adversary bound was used for formula evaluation [23, 25], triangle and other subgraph detection [3, 17, 5], the k-distinctness problem [2], and learning symmetric juntas [4]. This paper demonstrates an application to property testing. Also, many of the above algorithms use the framework of learning graphs [3], whereas our algorithm is not directly based on this framework.

2 Adversary bound

In query algorithms, the complexity measure is the number of queries to the input, given as a black box. All other computations are considered free. For the definition of randomized and quantum query complexity, the reader may refer to [8]. We use a general result by Reichardt et al., and obtain our quantum query algorithm from the corresponding *adversary bound*. Since the input to our algorithm is a total Boolean function, we define the version of the adversary bound tailored for this special case.

Let n be a positive integer, and assume that $\mathcal{X}$ and $\mathcal{Y}$ are two disjoint sets of total functions from $\{0,1\}^n$ to $\{0,1\}$. We will deal with the following problem. Given a query access to a function $f \in \mathcal{X} \cup \mathcal{Y}$, the task is to detect whether $f \in \mathcal{X}$ or $f \in \mathcal{Y}$. For this problem, the (dual) adversary bound is equal to the optimal value of the following optimization problem:

$$\begin{aligned}
&\text{minimize} && \max_{f\in\mathcal{X}\cup\mathcal{Y}} \sum\nolimits_{x\in\{0,1\}^n} X_x[\![f,f]\!] && && \text{(1a)}\\
&\text{subject to} && \sum\nolimits_{x: f(x)\neq g(x)} X_x[\![f,g]\!] = 1 && \text{for all } f\in\mathcal{X} \text{ and } g\in\mathcal{Y}; && \text{(1b)}\\
& && X_x \succeq 0 && \text{for all } x\in\{0,1\}^n, && \text{(1c)}
\end{aligned}$$

where X_x are $(\mathcal{X}\cup\mathcal{Y})\times(\mathcal{X}\cup\mathcal{Y})$ positive semi-definite matrices, and $X_x[\![f,g]\!]$ stands for the (f,g)-entry of the matrix X_x. The adversary bound is very useful because of the following result:

Theorem 2.1 ([14, 21, 22, 18])**.** *The (bounded-error) quantum query complexity of distinguishing $\mathcal{X}$ and $\mathcal{Y}$ is equal to the value of the adversary bound* (1)*, up to a constant factor.*

In particular, Theorem 2.1 implies that the value of a feasible solution to the optimization problem (1) gives an upper bound on the quantum query complexity of the corresponding problem.

3 Proof of Theorem 1.1

We complete the proof of Theorem 1.1 by constructing a feasible solution to the adversary bound (1) in the case where $\mathcal{X}$ is the set of all monotone functions and $\mathcal{Y}$ is the set of all functions that are ε-far away from any monotone function.

Let us introduce some notation. The *hypercube* is the graph with the vertex set $\{0,1\}^n$, where two vertices are adjacent iff they differ in exactly one coordinate. We consider the hypercube as an oriented graph, i. e., we define an edge of the hypercube as a pair xy, where $x\prec y$ and x and y differ in exactly one coordinate. For a function $f\colon\{0,1\}^n\to\{0,1\}$, an *$(a,b)$-edge of f* is an edge xy of the hypercube such that $f(x)=a$ and $f(y)=b$. We also use the notation $\{x,y\}$ to denote an edge of the hypercube without imposing the order, i. e., $\{x,y\}=xy$ if $x\prec y$, and $\{x,y\}=yx$ if $x\succ y$. The *total influence* or *average sensitivity* $\mathbb{I}(f)$ is the number of edges xy of the hypercube such that $f(x)\neq f(y)$, divided by 2^{n-1}. For a monotone function, it is known to be $O(\sqrt{n})$ [20, Theorem 2.33]. We use $x^{\oplus j}$ for the string x with the j-th bit flipped. Finally, for any predicate P, we use 1_P to denote the indicator variable that equals 1 when P is true and 0 otherwise.

Unlike the Khot–Minzer–Safra algorithm, which tests pairs of inputs at significant distance, our algorithm is essentially an improvement of the edge tester. Let us first describe a solution to the adversary bound that corresponds to the edge tester. This solution has query complexity $O(\sqrt{n/\varepsilon})$. We then show how to improve the query complexity to $\tilde{O}(n^{1/4}\varepsilon^{-1/2})$.

Edge tester. For a function $g\in\mathcal{Y}$, let E_g denote the set of all $(1,0)$-edges of g. Goldreich et al. [13] showed that for every function $g\in\mathcal{Y}$,

$$|E_g|\geq \varepsilon 2^n\,.$$

This inequality is the central component of the analysis of the classical edge tester; it is also a key component of the analysis of the quantum algorithm described below.

To construct a feasible solution to the adversary bound (1), we first construct a positive semidefinite matrix $Z_{x,j}$ for each edge $\{x, x^{\oplus j}\}$ of the hypercube. Formally, let $Z_{x,j} = \phi_{x,j}\phi_{x,j}^*$, where the entries of the vector $\phi_{x,j}$ are labeled by functions $f \in \mathcal{X} \cup \mathcal{Y}$ and are defined by

$$\phi_{x,j}[\![f]\!] = \begin{cases} 1/\sqrt{L} & \text{if } f \in \mathcal{X},\ x_j = 0, \text{ and } f(x) = 0, \\ 1/\sqrt{L} & \text{if } f \in \mathcal{X},\ x_j = 1, \text{ and } f(x^{\oplus j}) = f(x) = 1, \\ \sqrt{L}/|E_f| & \text{if } f \in \mathcal{Y}, \text{ and } \{x, x^{\oplus j}\} \in E_f, \text{ or} \\ 0 & \text{otherwise,} \end{cases}$$

where L is a parameter to be specified later. The coefficients of $\phi_{x,j}$ are chosen so that for any functions $f \in \mathcal{X}$, $g \in \mathcal{Y}$ and any edge xy of the hypercube, we have

$$Z_{x,j}[\![f,g]\!] \cdot 1_{f(x)\neq g(x)} + Z_{y,j}[\![f,g]\!] \cdot 1_{f(y)\neq g(y)} = \frac{1_{xy\in E_g}}{|E_g|} \tag{2}$$

if x and y differ in the j-th coordinate.

We can obtain a feasible solution to the adversary bound (1) by setting

$$X_x = \sum_{j\in[n]} Z_{x,j}$$

for every $x \in \{0,1\}^n$. The matrices X_x clearly satisfy (1c). Summing (2) over all edges of the hypercube, we see that the matrices also satisfy (1b). For any $f \in \mathcal{X}$ and $g \in \mathcal{Y}$, we have

$$\sum_{x\in\{0,1\}^n} X_x[\![f,f]\!] = \frac{n2^{n-1}}{L} \qquad \text{and} \qquad \sum_{x\in\{0,1\}^n} X_x[\![g,g]\!] = 2|E_g| \cdot \frac{L}{|E_g|^2} \leq \frac{L}{\varepsilon 2^{n-1}}.$$

Choosing $L = 2^{n-1}\sqrt{n\varepsilon}$ yields $\sqrt{n/\varepsilon}$ as the value of the objective function (1a).

Our solution. Intuitively, the main problem with the above solution is that each $f \in \mathcal{X}$ is used in $n2^{n-1}$ matrices $Z_{x,j}$, whereas each $g \in \mathcal{Y}$ is only used in $\varepsilon 2^n$ of them. Thus, many uses of f are "wasted," which results in the relatively high query complexity of the algorithm. To reduce the query complexity, we wish to reduce the number of matrices featuring f.

A natural idea is to "pack" together matrices that share the same vertex of the hypercube. To describe this idea, we need some more notation. For $g \in \mathcal{Y}$, define the $(1,0)$-*graph of* g to be the subgraph of the hypercube induced by the set of all $(1,0)$-edges of g. Let G_g be a subgraph of the $(1,0)$-graph of g, and let E_g denote the set of edges in G_g. Note that this time we do not require E_g to contain *all* $(1,0)$-edges of g. The precise choice of G_g will be specified later. Let $\deg_g(x)$ denote the degree of a vertex $x \in \{0,1\}^n$ in G_g.

For each $x \in \{0,1\}^n$, consider a positive semidefinite matrix $Y_x = \psi_x\psi_x^*$, where

$$\psi_x[\![f]\!] = \begin{cases} 1/\sqrt{K} & \text{if } f \in \mathcal{X}, \text{ or} \\ \sqrt{K}\deg_f(x)/|E_f| & \text{if } f \in \mathcal{Y}, \end{cases}$$

and K is a parameter to be defined later. For $f \in \mathcal{X}$ and $g \in \mathcal{Y}$, we have

$$\sum_{x: f(x)\neq g(x)} Y_x[\![f,g]\!] = \sum_{x: f(x)\neq g(x)} \frac{\deg_g(x)}{|E_g|} = \sum_{xy\in E_g} \frac{1_{f(x)\neq g(x)} + 1_{f(y)\neq g(y)}}{|E_g|}\,. \tag{3}$$

Since $g(x) = 1$ and $g(y) = 0$ for any $xy \in E_g$, we essentially recover (2) if $f(x) = f(y)$. The only bad case is when xy is a $(0,1)$-edge for f, in which case we get 2 in the numerator on the right-hand side of (3). We can fix this using matrices $Z_{x,j}$ that are very similar to those used in the edge tester. Specifically, for every $x \in \{0,1\}^n$ and $j \in [n]$, let $Z_{x,j} = \phi_{x,j}\phi^*_{x,j}$ with

$$\phi_{x,j}[\![f]\!] = \begin{cases} -1/\sqrt{L} & \text{if } f \in \mathcal{X},\ x_j = 0, \text{ and } (x, x^{\oplus j}) \text{ is a } (0,1)\text{-edge,} \\ \sqrt{L}/|E_f| & \text{if } f \in \mathcal{Y},\ x_j = 0, \text{ and } (x, x^{\oplus j}) \in E_f, \text{ or} \\ 0 & \text{otherwise,} \end{cases}$$

where L is another parameter to be fixed later. Note that in this definition, any function $f \in \mathcal{X}$ is used in only $O(\sqrt{n}2^n)$ of the $Z_{x,j}$ matrices, since this is the number of $(0,1)$-edges of f. This is the reason we get the improvement from $n^{1/2}$ to $n^{1/4}$ queries. Also, this step has no classical analogue, and that is where we get a super-quadratic improvement to the classical edge tester and the Khot–Minzer–Safra algorithm.

Define $Z_x = \sum_{j\in[n]} Z_{x,j}$. For every $f \in \mathcal{X}$ and $g \in \mathcal{Y}$,

$$\sum_{x: f(x)\neq g(x)} Z_x[\![f,g]\!] = -\sum_{xy\in E_g} \frac{1_{f(x)\neq g(x)} \cdot 1_{f(y)\neq g(y)}}{|E_g|}\,. \tag{4}$$

Finally, define $X_x = Y_x + Z_x$. Clearly, this choice satisfies (1c). Also, for any $xy \in E_g$, at least one of the conditions $f(x) \neq g(x)$ or $f(y) \neq g(y)$ is satisfied, so equations (3) and (4) imply (1b).

It remains to bound the objective value (1a). The contribution of the matrices Z_x to (1a) is easily computed. Namely, for $f \in \mathcal{X}$ and $g \in \mathcal{Y}$,

$$\sum_{x\in\{0,1\}^n} Z_x[\![f,f]\!] = \frac{\mathbb{I}(f)2^{n-1}}{L} \qquad \text{and} \qquad \sum_{x\in\{0,1\}^n} Z_x[\![g,g]\!] = |E_g| \cdot \frac{L}{|E_g|^2} \leq \frac{L}{\varepsilon 2^n}\,.$$

The matrices Y_x are more problematic. For $g \in \mathcal{Y}$, we have

$$\sum_{x\in\{0,1\}^n} Y_x[\![g,g]\!] = \frac{K}{|E_g|^2} \sum_{x\in\{0,1\}^n} \deg_g(x)^2\,.$$

For this expression to be small enough to obtain the query complexity claimed in Theorem 2.1, we not only need E_g to be as large as in the edge tester, but we also need the edges of G_g to be evenly distributed among the vertices. As it turns out, the main technical result behind the analysis of the Khot–Minzer–Safra algorithm claims exactly this:

Lemma 3.1 ([16, Lemma 7.1])**.** *For every $g\colon \{0,1\}^n \to \{0,1\}$ that is ε-far from being monotone, there exists a subgraph G_g of the $(1,0)$-graph of g such that*

$$|E_g| = \Omega\left(\frac{\varepsilon 2^n \sqrt{\Delta(G_g)}}{\log^2 n}\right), \tag{5}$$

where E_g is the edge set of G_g, and $\Delta(G_g)$ is the maximal degree of G_g.

Fix G_g to be one of the graphs whose existence is guaranteed by Lemma 3.1. We can now estimate the objective value (1a). For a function $f \in \mathcal{X}$, we have

$$\sum_{x\in\{0,1\}^n} X_x[\![f,f]\!] = 2^n\left(\frac{1}{K}+\frac{\mathbb{I}(f)}{2L}\right) = 2^n\cdot O\left(\frac{1}{K}+\frac{\sqrt{n}}{L}\right). \tag{6}$$

Meanwhile, for $g \in \mathcal{Y}$, we have

$$\sum_{x\in\{0,1\}^n} X_x[\![g,g]\!] = \sum_{x\in\{0,1\}^n} \frac{K\deg_g(x)^2}{|E_g|^2} + |E_g|\cdot\frac{L}{|E_g|^2}.$$

Using the inequality $\sum_x \deg_g(x)^2 \le \Delta(G_g)\sum_x \deg_g(x)$ and the identity $\sum_x \deg_g(x) = 2|E_g|$ (the handshaking lemma), we observe that

$$\sum_{x\in\{0,1\}^n} X_x[\![g,g]\!] \le K\frac{\Delta(G_g)}{|E_g|}\cdot\frac{\sum_x \deg_g(x)}{|E_g|} + \frac{L}{|E_g|} = 2K\frac{\Delta(G_g)}{|E_g|} + \frac{L}{|E_g|}.$$

Applying (5), we obtain

$$\sum_{x\in\{0,1\}^n} X_x[\![g,g]\!] \le \frac{\log^2 n}{\varepsilon 2^n}\cdot O\left(K\sqrt{\Delta(G_g)} + \frac{L}{\sqrt{\Delta(G_g)}}\right) = \frac{\log^2 n}{\varepsilon 2^n}\cdot O\big(K\sqrt{n}+L\big). \tag{7}$$

Comparing (6) and (7), we see that the objective value is minimized when

$$K = 2^n\sqrt{\varepsilon}n^{-1/4}/\log n \qquad \text{and} \qquad L = 2^n\sqrt{\varepsilon}n^{1/4}/\log n.$$

Taking these values, the objective value (1a) is $O(n^{1/4}\varepsilon^{-1/2}\log n)$, as desired.

References

[1] ANDRIS AMBAINIS, ALEKSANDRS BELOVS, ODED REGEV, AND RONALD DE WOLF: Efficient quantum algorithms for (gapped) group testing and junta testing. Preprint, 2015. To appear in SODA'16. [arXiv:1507.03126] 405

[2] ALEKSANDRS BELOVS: Learning-graph-based quantum algorithm for k-distinctness. In *Proc. 53rd FOCS*, pp. 207–216. IEEE Comp. Soc. Press, 2012. [doi:10.1109/FOCS.2012.18, arXiv:1205.1534] 405

[3] ALEKSANDRS BELOVS: Span programs for functions with constant-sized 1-certificates. In *Proc. 44th STOC*, pp. 77–84. ACM Press, 2012. [doi:10.1145/2213977.2213985, arXiv:1105.4024] 405

[4] ALEKSANDRS BELOVS: Quantum algorithms for learning symmetric juntas via the adversary bound. *Comput. Complexity*, 24(2):255–293, 2015. Preliminary version in CCC'14. [doi:10.1007/s00037-015-0099-2, arXiv:1311.6777] 405

[5] ALEKSANDRS BELOVS AND BEN W. REICHARDT: Span programs and quantum algorithms for *st*-connectivity and claw detection. In *Proc. 20th Ann. European Symp. on Algorithms (ESA'12)*, volume 7501 of *LNCS*, pp. 193–204, 2012. [doi:10.1007/978-3-642-33090-2_18, arXiv:1203.2603] 405

[6] GILLES BRASSARD, PETER HØYER, MICHELE MOSCA, AND ALAIN TAPP: Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pp. 53–74, 2002. [doi:10.1090/conm/305, arXiv:quant-ph/0005055] 404

[7] 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 versions in RANDOM'10 and ECCC. [doi:10.1007/s00493-012-2765-1] 405

[8] HARRY BUHRMAN AND RONALD DE WOLF: Complexity measures and decision tree complexity: a survey. *Theoret. Comput. Sci.*, 288(1):21–43, 2002. [doi:10.1016/S0304-3975(01)00144-X] 405

[9] DEEPARNAB CHAKRABARTY AND COMANDUR SESHADHRI: A $o(n)$ monotonicity tester for boolean functions over the hypercube. In *Proc. 45th STOC*, pp. 411–418. ACM Press, 2013. [doi:10.1145/2488608.2488660, arXiv:1302.4536] 404

[10] 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 Press, 2015. [doi:10.1145/2746539.2746570, arXiv:1412.5657] 404, 405

[11] 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 Comp. Soc. Press, 2014. [doi:10.1109/FOCS.2014.38, arXiv:1412.5655] 404, 405

[12] 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 Press, 2002. [doi:10.1145/509907.509977] 405

[13] 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] 404, 406

[14] PETER HØYER, TROY LEE, AND ROBERT ŠPALEK: Negative weights make adversaries stronger. In *Proc. 39th STOC*, pp. 526–535. ACM Press, 2007. [doi:10.1145/1250790.1250867] 406

[15] KAZUO IWAMA, HARUMICHI NISHIMURA, RUDY RAYMOND, AND JUNICHI TERUYAMA: Quantum counterfeit coin problems. *Theoret. Comput. Sci.*, 456:51–64, 2012. Preliminary version in ISAAC'10. [doi:10.1016/j.tcs.2012.05.039, arXiv:1009.0416] 405

[16] SUBHASH KHOT, DOR MINZER, AND MULI SAFRA: On monotonicity testing and boolean isoperimetric type theorems. In *Proc. 56th FOCS*, pp. 52–58. IEEE Comp. Soc. Press, 2015. ECCC 2015/011. [doi:10.1109/FOCS.2015.13] 404, 405, 408

[17] TROY LEE, FRÉDÉRIC MAGNIEZ, AND MIKLOS SANTHA: Improved quantum query algorithms for triangle finding and associativity testing. In *Proc. 24th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 1486–1502. ACM Press, 2013. [doi:10.1137/1.9781611973105.107, arXiv:1210.1014] 405

[18] TROY LEE, RAJAT MITTAL, BEN W. REICHARDT, ROBERT ŠPALEK, AND MARIO SZEGEDY: Quantum query complexity of state conversion. In *Proc. 52nd FOCS*, pp. 344–353. IEEE Comp. Soc. Press, 2011. [doi:10.1109/FOCS.2011.75, arXiv:1011.3020] 405, 406

[19] ASHLEY MONTANARO AND RONALD DE WOLF: A Survey of Quantum Property Testing. Preprint, 2013. To appear as a *Theory of Computing Graduate Survey*. [arXiv:1310.2035] 404

[20] RYAN O'DONNELL: *Analysis of Boolean Functions*. Cambridge Univ. Press, 2014. 406

[21] BEN W. REICHARDT: Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Proc. 50th FOCS*, pp. 544–551. IEEE Comp. Soc. Press, 2009. [doi:10.1109/FOCS.2009.55, arXiv:0904.2759] 405, 406

[22] BEN W. REICHARDT: Reflections for quantum query algorithms. In *Proc. 22nd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'11)*, pp. 560–569. ACM Press, 2011. [doi:10.1137/1.9781611973082.44, arXiv:1005.1601] 406

[23] BEN W. REICHARDT AND ROBERT ŠPALEK: Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012. Preliminary version in STOC'08. [doi:10.4086/toc.2012.v008a013] 405

[24] WIM VAN DAM AND IGOR E. SHPARLINSKI: Classical and quantum algorithms for exponential congruences. In *Proc. 3rd Conf. Theory of Quantum Comput., Communic. and Cryptography (TQC'08)*, volume 5106 of *LNCS*, pp. 1–10. Springer, 2008. [doi:10.1007/978-3-540-89304-2_1, arXiv:0804.1109] 405

[25] BOHUA ZHAN, SHELBY KIMMEL, AND AVINATAN HASSIDIM: Super-polynomial quantum speed-ups for Boolean evaluation trees with hidden structure. In *Proc. 3rd Symp. Innovations in Theoretical Computer Science (ITCS'12)*, pp. 249–265. ACM Press, 2012. [doi:10.1145/2090236.2090258, arXiv:1101.0796] 405

AUTHORS

Aleksandrs Belovs
Researcher
University of Latvia, Riga, Latvia
stiboh@gmail.com
`http://home.lu.lv/~belovs`

Eric Blais
Assistant professor
University of Waterloo, Waterloo, ON
eric.blais@uwaterloo.ca
http://www.cs.uwaterloo.ca/~eblais

ABOUT THE AUTHORS

ALEKSANDRS BELOVS is a researcher at the Faculty of Computing, University of Latvia. He obtained his Master's degree from the University of Waterloo, and his Ph. D. in 2014 from the University of Latvia under the supervision of Andris Ambainis. Afterwards, he spent a year as a postdoc at CSAIL, MIT. Currently, he is at CWI, Amsterdam. His main research area is quantum computing.

ERIC BLAIS is an assistant professor at the University of Waterloo. He completed his Ph. D. at Carnegie Mellon University. His advisor was Ryan O'Donnell. He was subsequently a Simons Foundation Postdoctoral Fellow in the Theory of Computation group at MIT.