Optimal quantum query bounds for almost all Boolean functions*

Andris Ambainis[1], Arturs Bačkurs[2], Juris Smotrovs[1], and Ronald de Wolf[3]

1 **University of Latvia**
Riga, Latvia
{ambainis,Juris.Smotrovs}@lu.lv
2 **MIT, Cambridge, MA**
(work done while at University of Latvia)
abackurs@gmail.com
3 **CWI and University of Amsterdam**
Amsterdam, The Netherlands
rdewolf@cwi.nl

Abstract

We show that almost all n-bit Boolean functions have bounded-error quantum query complexity at least $n/2$, up to lower-order terms. This improves over an earlier $n/4$ lower bound of Ambainis [1], and shows that van Dam's oracle interrogation [9] is essentially optimal for almost all functions. Our proof uses the fact that the acceptance probability of a T-query algorithm can be written as the sum of squares of degree-T polynomials.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases quantum computing, query complexity, lower bounds, polynomial method

Digital Object Identifier 10.4230/LIPIcs.STACS.2013.446

1 Introduction

Most known quantum algorithms have been developed in the setting of quantum query complexity, which is the quantum generalization of the model of decision tree complexity. Here an algorithm is charged for each "query" to the input bits, while intermediate computation is free (see [8] for more details about this model). For certain specific functions one can obtain large quantum-speedups in this model. For example, Grover's algorithm [14] computes the n-bit OR function with $O(\sqrt{n})$ queries, while any classical algorithm needs $\Omega(n)$ queries. Many more such polynomial speed-ups are known, see for example [3, 18, 11, 6]. If one considers partial functions there are even exponential speed-ups, for example [10, 20, 19, 5]. Substantial quantum speed-ups are quite rare, and exploit very specific structure in problems that makes those problems amenable to quantum speed-ups.

On the other hand, one can also obtain a smaller speed-up that holds for *almost all* Boolean functions. Classically, almost all Boolean functions $f : \{0,1\}^n \to \{0,1\}$ have

* AA, AB, RdW are partially supported by the European Commission under the project QCS (Grant No. 255961). AA and JS are partially supported by ESF project 1DP/1.1.1.2.0/09/APIA/VIAA/044. RdW is partially supported by a Vidi grant from the Netherlands Organization for Scientific Research (NWO).

30th Symposium on Theoretical Aspects of Computer Science (STACS'13).
Editors: Natacha Portier and Thomas Wilke; pp. 446–453

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


bounded-error query complexity n, minus lower-order terms. This is quite intuitive: if we have only seen 99% of the n input bits, then the restriction of a random function to the 1% remaining variables will still be roughly balanced between 0 and 1-inputs. In contrast, van Dam [9] exhibited a beautiful quantum algorithm that recovers the complete n-bit input x with high probability using roughly $n/2$ quantum queries. Briefly, his algorithm is as follows:

1. With $T = n/2 + O(\sqrt{n\log(1/\varepsilon)})$ and $B = \sum_{i=0}^{T}\binom{n}{i}$ being the number of $y \in \{0,1\}^n$ with Hamming weight $|y| \leq T$, set up the n-qubit superposition $\frac{1}{\sqrt{B}}\sum_{y\in\{0,1\}^n:|y|\leq T}|y\rangle$.
2. Apply the unitary $|y\rangle \mapsto (-1)^{x\cdot y}|y\rangle$. We can implement this using T queries to the input x, for all basis states $|y\rangle$ with $|y| \leq T$.
3. Apply a Hadamard transform to all qubits and measure.

To see correctness of this algorithm, note that the fraction of n-bit strings y of Hamming weight larger than T is $\ll \varepsilon$. Hence the state obtained in step 2 is very close to the state $\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$, whose Hadamard transform is exactly $|x\rangle$.

Since obtaining x suffices to compute $f(x)$ for any f of our choice, van Dam's algorithm implies that the ε-error quantum query complexity of f is

$$Q_\varepsilon(f) \leq n/2 + O(\sqrt{n\log(1/\varepsilon)}) \quad \text{for all Boolean functions.}$$

It is known that this upper bound is essentially tight for *some* Boolean functions. For example, $Q_\varepsilon(f) = \lceil n/2\rceil$ for the n-bit Parity function [4, 12]. Our goal in this paper is to show that it is tight for *almost all* Boolean functions, i.e., that $Q_\varepsilon(f)$ is essentially lower bounded by $n/2$ for almost all f (and fixed ε). How can we prove such a lower bound? Two general methods are known for proving quantum query lower bounds: the polynomial method [4] and the adversary method [2, 15]. As we explain below, in their standard form neither method is strong enough to prove our desired $n/2$ lower bound.

First, the adversary method in its strongest incarnation [15, Theorem 2] has the form

$$Q_\varepsilon(f) \geq \frac{1}{2}(1-\sqrt{\varepsilon(1-\varepsilon)})\mathrm{ADV}^{\pm}(f),$$

where the "negative-weights adversary bound" $\mathrm{ADV}^{\pm}(f)$ is a quantity that is at most n. Accordingly, for constant error probability ε the adversary method can only prove lower bounds of the form cn for some $c < 1/2$.

Second, the polynomial method uses the fact (first proved in [13, 4]) that the acceptance probability of a T-query algorithm can be written as a degree-$2T$ n-variate multilinear real polynomial $p(x)$ of the input. If the algorithm computes f with error probability $\leq \varepsilon$, then $p(x)$ will approximate $f(x)$: $p(x) \in [0,\varepsilon]$ for every $x \in f^{-1}(0)$ and $p(x) \in [1-\varepsilon, 1]$ for every $x \in f^{-1}(1)$. Accordingly, a lower bound of d on the ε-approximate polynomial degree $\deg_\varepsilon(f)$ implies a lower bound of $d/2$ on the ε-error quantum query complexity of f. This is how Ambainis [1] proved the current best lower bound of roughly $n/4$ that holds for almost all n-bit Boolean functions: he showed that almost all f satisfy $\deg_\varepsilon(f) \geq (1/2 - o(1))n$. However, O'Donnell and Servedio [17] proved a nearly matching upper bound: $\deg_\varepsilon(f) \leq (1/2 + o(1))n$ for almost all f. Hence Ambainis's lower bound approach via approximate degree cannot be improved to obtain our desired lower bound of $n/2$ on $Q_\varepsilon(f)$.[1] This suggests that also the polynomial method is unable to obtain the conjectured factor $1/2$ in the lower bound.

[1] In fact, the *unbounded-error* quantum query complexity of almost all Boolean functions is only $n/4$ up to lower-order terms. This follows from the degree upper bound of [17] combined with [7, Theorem 1] and the fact that d-bit Parity can be computed with $\lceil d/2\rceil$ quantum queries.

However, looking under the hood of the polynomial method, it actually gives a bit more information about the acceptance probability: $p(x)$ is not an arbitrary degree-$2T$ polynomial, but the sum of squares of degree-T polynomials. Using this extra information, we prove in this paper that indeed $Q_\varepsilon(f) \geq n/2$ up to lower-order terms for almost all f.[2]

Our main technical result will be a claim about certain random matrices (Claim 1 below), which may have further applications. It says the following. Let $\mathcal{B} = \{x \in \{0,1\}^n : |x| \leq T\}$ be the set of strings of weight at most T, and $B = |\mathcal{B}|$ its size. Suppose F is a $2^n \times 2^n$ diagonal matrix with randomly chosen signs on its diagonal, and $\widehat{F} = HFH$ is F conjugated with the unitary Hadamard transform. Then the principal minor of $\widehat{F}$ restricted to entries in $\mathcal{B} \times \mathcal{B}$ has (with probability $1 - o(1)$) operator norm $O(\sqrt{nB^{1+o(1)}/2^n})$. In particular, if $T \leq (1/2 - \varepsilon)n$ for any fixed positive ε then with high probability this operator norm is $o(1)$.

2 Proof

Suppose we have a quantum algorithm that uses T queries to its n-bit input x. Then by [4, Lemma 4.1], its final state can be written as a function of the input as

$$\sum_z \alpha_z(x)|z\rangle,$$

where z ranges over the computational basis states of the algorithm's space, and the amplitudes $\alpha_z(x)$ are complex-valued multilinear n-variate polynomials of degree $\leq T$. We assume w.l.o.g. that the algorithm determines its Boolean output by measuring the first qubit of the final state. Then the acceptance probability (as a function of input x) is the following polynomial of degree $\leq 2T$:

$$p(x) = \sum_{z:z_1=1} |\alpha_z(x)|^2.$$

Let $\alpha_z \in \mathbb{C}^{2^n}$ denote the vector with entries $\alpha_z(x)$. Define the following $2^n \times 2^n$ matrix P:

$$P = \sum_{z:z_1=1} \alpha_z \alpha_z^*.$$

The diagonal entry P_{xx} of this matrix is $p(x)$. Since P is positive semidefinite, we have[3]

$$\|P\|_1 = \mathrm{Tr}(P) = \sum_{x \in \{0,1\}^n} p(x).$$

With H denoting the n-qubit Hadamard transform, $H\alpha_z$ is proportional to the Fourier transform $\widehat{\alpha_z}$, which has support only on the $B = \sum_{i=0}^{T} \binom{n}{i}$ monomials of degree $\leq T$. Hence the matrix HPH has support only on a $B \times B$ submatrix.

It will be convenient to use $+1$ and -1 as the range of a Boolean function, rather than 0 and 1. Consider Boolean function $f : \{0,1\}^n \to \{\pm 1\}$. For $s \in \{0,1\}^n$, the corresponding

[2] Magnin and Roland [16] independently found similar ways to strengthen the standard polynomial method; however they do not apply their tools to the analysis of random Boolean functions.

[3] We use the following matrix-analytic notation. For $m \times m$ matrices A and A', define inner product $\langle A, A' \rangle = \mathrm{Tr}(A^* A') = \sum_{i,j} A^*_{ij} A'_{ij}$. Note that this inner product is basis-independent: for every unitary U we have $\langle UAU^*, UA'U^* \rangle = \langle A, A' \rangle$. Let $\|A\|_p$ denote the (unitarily invariant) Schatten p-norm of A, which is the p-norm of the m-dimensional vector of singular values of A. In particular, $\|A\|_1$ is the sum of the singular values of A, and $\|A\|_\infty$ is its largest singular value, which is the operator norm of A. It is easy to see that $\|A\|_2^2 = \mathrm{Tr}(A^* A) = \sum_{i,j} |A_{ij}|^2$, and $\langle A, B \rangle \leq \|A\|_1 \|B\|_\infty$.

Fourier coefficient of f is defined as $\widehat{f}(s) = \frac{1}{2^n}\sum_x (-1)^{s\cdot x} f(x)$. Let F be the $2^n \times 2^n$ diagonal matrix with diagonal entries $f(x)$. Define $\widehat{F} = HFH$. Then for $s,t \in \{0,1\}^n$, we have

$$\widehat{F}_{s,t} = \langle s|HFH|t\rangle = \frac{1}{2^n}\sum_{x,y}(-1)^{s\cdot x}(-1)^{t\cdot y}F_{xy} = \frac{1}{2^n}\sum_x (-1)^{(s\oplus t)\cdot x} f(x) = \widehat{f}(s\oplus t).$$

Let $\widehat{F}_T$ denote $\widehat{F}$ after zeroing out all s,t-entries where $|s| > T$ and/or $|t| > T$. Note that HPH doesn't have support on the entries that are zeroed out, hence $\langle HPH, \widehat{F}\rangle = \langle HPH, \widehat{F}_T\rangle$.

Suppose our T-query quantum algorithm computes f with worst-case error probability at most some fixed constant $\leq \varepsilon$. Output 1 means the algorithm thinks $f(x) = 1$, and output 0 means it thinks $f(x) = -1$. Then for every $x \in \{0,1\}^n$, $2p(x) - 1$ differs from $f(x)$ by at most 2ε. Hence:

$$\begin{aligned}
(1-2\varepsilon)2^n &\leq \langle 2P - I, F\rangle \\
&= 2\langle P, F\rangle - \sum_x f(x) \\
&= 2\langle HPH, \widehat{F}\rangle - \sum_x f(x) \\
&= 2\langle HPH, \widehat{F}_T\rangle - \sum_x f(x) \\
&\leq 2\|P\|_1 \left\|\widehat{F}_T\right\|_\infty - \sum_x f(x) \\
&= 2\left\|\widehat{F}_T\right\|_\infty \sum_x p(x) - \sum_x f(x).
\end{aligned}$$

We can assume w.l.o.g. that $\sum_x f(x) \geq 0$ (if this doesn't hold for f then just take its negation, which has the same query complexity as f). Since $\sum_x p(x) \leq 2^n$, we get

$$\left\|\widehat{F}_T\right\|_\infty \geq 1/2 - \varepsilon. \tag{1}$$

The technically hard part is to upper bound $\left\|\widehat{F}_T\right\|_\infty$ for most f. So consider the case where $f : \{0,1\}^n \to \{\pm 1\}$ is a *uniformly random* function, meaning that the 2^n values $f(x)$ are independent uniformly random signs. In the next subsection we show

▶ Claim 1. With probability $1-o(1)$ (over the choice of f) we have $\left\|\widehat{F}_T\right\|_\infty = O\left(\sqrt{\frac{nB^{1+o(1)}}{2^n}}\right)$.

Combining this with the lower bound (1), we get that $B \geq 2^{n-o(n)}$. On the other hand, a well-known upper bound on the sum of binomial coefficients is $B = \sum_{i=0}^T \binom{n}{i} \leq 2^{nH(T/n)}$, where $H(q) = -q\log q - (1-q)\log(1-q)$ denotes the binary entropy function. Hence, $2^{n-o(n)} \leq 2^{nH(T/n)}$ which implies $T \geq n/2 - o(n)$. This shows that $Q_\epsilon(f) \geq n/2 - o(n)$ for almost all f (and fixed constant ε).

2.1 Proof of Claim 1

Below, unless mentioned otherwise, probabilities and expectations will be taken over the random choice of f. We choose $T = n/2 - o(n)$ sufficiently small that $B = \sum_{i=0}^T \binom{n}{i} = o(2^n)$, i.e., the $o(n)$ term in T is taken to be $\omega(\sqrt{n})$.

Let λ_i be the i-th eigenvalue of $\widehat{F}_T$. Since $\widehat{F}_T$ is symmetric we have

$$\left\|\widehat{F}_T\right\|_\infty = \max_i |\lambda_i| = \sqrt[2k]{\max_i \lambda_i^{2k}} \leq \sqrt[2k]{\sum_i \lambda_i^{2k}} = \sqrt[2k]{\mathrm{Tr}(\widehat{F}_T^{2k})}.$$

We are going to show that

$$\mathbb{E}\left[\mathrm{Tr}(\widehat{F}_T^{2k})\right] = O\left(B\,(B/2^n)^k\right) \tag{2}$$

for every constant k (with a big-O constant depending on k). This means that, using Markov's inequality,

$$\begin{aligned}\Pr\left[\left\|\widehat{F}_T\right\|_\infty > C\sqrt{nB^{1+1/k}/2^n}\right] &\leq \Pr\left[\sqrt[2k]{\mathrm{Tr}(\widehat{F}_T^{2k})} > C\sqrt{nB^{1+1/k}/2^n}\right] \\ &= \Pr\left[\mathrm{Tr}(\widehat{F}_T^{2k}) > C^{2k}n^kB^{k+1}/2^{nk}\right] \\ &\leq \frac{\mathbb{E}\left[\mathrm{Tr}(\widehat{F}_T^{2k})\right]}{C^{2k}n^kB^{k+1}/2^{nk}} = o(1).\end{aligned}$$

Since this is true for any constant k, Claim 1 follows.

So now our goal is to prove (2). Below we let each of $s_1,\ldots,s_{2k}$ range over the B n-bit strings of weight $\leq T$, and each of $x_1,\ldots,x_{2k}$ range over $\{0,1\}^n$. For simplicity we abbreviate $\vec{s} = s_1,s_2,\ldots,s_{2k}$ and $\vec{x} = x_1,x_2,\ldots,x_{2k}$. Writing out the $2k$-fold matrix product, we have

$$\mathbb{E}\left[\mathrm{Tr}(\widehat{F}_T^{2k})\right] = \mathbb{E}\left[\sum_{\vec{s}} \widehat{f}(s_1\oplus s_2)\widehat{f}(s_2\oplus s_3)\cdots\widehat{f}(s_{2k}\oplus s_1)\right] \tag{3}$$

$$= \frac{1}{2^{2nk}}\sum_{\vec{s}}\sum_{\vec{x}} \mathbb{E}\left[(-1)^{(s_1\oplus s_2)\cdot x_1} f(x_1)\cdots(-1)^{(s_{2k}\oplus s_1)\cdot x_{2k}} f(x_{2k})\right] \tag{4}$$

$$= \frac{1}{2^{2nk}}\sum_{\vec{s}}\sum_{\vec{x}} (-1)^{(s_1\oplus s_2)\cdot x_1+\cdots+(s_{2k}\oplus s_1)\cdot x_{2k}}\,\mathbb{E}\left[f(x_1)\cdots f(x_{2k})\right]. \tag{5}$$

For a particular $y\in\{0,1\}^n$, there are as many Boolean functions having $f(y)=1$ as having $f(y)=-1$, independently of what is known about values of f on other inputs. Thus, if any y occurs an odd number of times in $\vec{x}=(x_1,\ldots,x_{2k})$, then $\mathbb{E}[f(x_1)\cdots f(x_{2k})]=0$. So only those summands are left where all multiplicities of distinct values among $x_1,\ldots,x_{2k}$ are even. We call such $\vec{x}$ *even*. We have

$$\begin{aligned}\mathbb{E}\left[\mathrm{Tr}(\widehat{F}_T^{2k})\right] &= \frac{1}{2^{2nk}}\sum_{\vec{s}}\sum_{\vec{x}\text{ even}} (-1)^{\sum_{i=1}^{2k}(s_i\oplus s_{i+1})\cdot x_i} \\ &= \frac{1}{2^{2nk}}\sum_{r}\sum_{\substack{\text{partition of}\\ \{1,\ldots,2k\}\text{ into even}\\ \text{non-empty } I_1,\ldots,I_r}}\sum_{\vec{s}}\sum_{\substack{x^{(1)},\ldots,x^{(r)}\\ \text{different}}} (-1)^{\sum_{j=1}^{r}\left(\bigoplus_{i\in I_j}(s_i\oplus s_{i+1})\right)\cdot x^{(j)}}\end{aligned} \tag{6}$$

where $s_{2k+1}=s_1$ and the second summation is over all partitions of $\{1,\ldots,2k\}$ into even-sized non-empty parts $I_1,\ldots,I_r$ with the implied condition that $x_i=x_j$ iff i and j belong to the same part. Since the number of such partitions $(I_1,I_2,\ldots,I_r)$ depends only on k (which is a constant), it suffices to prove that each term in the sum is of the order $O(B(B/2^n)^k)$. We will do this by proving

▶ Claim 2. For any fixed m and any partition $I_1,\ldots,I_r$ of $\{1,\ldots,m\}$:

$$\sum_{\vec{s}}\sum_{\substack{x^{(1)},\ldots,x^{(r)}\\ \text{different}}} (-1)^{\sum_{j=1}^{r} t_j(\vec{s})\cdot x^{(j)}} = O(B^{m-r+1}\cdot 2^{nr}) \tag{7}$$

where $t_j(\vec{s}) = \bigoplus_{i\in I_j}(s_i\oplus s_{i+1})$, $s_{m+1}=s_1$, and the big-O constant depends on m and the partition.

We first show that Claim 2 implies Claim 1. In our case, $m = 2k$. Since $B = o(2^n)$, the upper bound $B^{2k-r+1} \cdot 2^{nr}$ increases when r increases. Since each partition of $\{1, \ldots, 2k\}$ into even-sized non-empty parts $I_1, \ldots, I_r$ must contain at least 2 elements in each I_j, we must have $r \leq (2k)/2 = k$ and every term of the sum (6) is upper bounded by

$$\frac{1}{2^{2nk}} O\left(B^{2k-k+1} \cdot 2^{nk}\right) = O\left(B\left(B/2^n\right)^k\right).$$

It remains to prove Claim 2, which we do by induction on r. If $r = 1$ then $t_1(\vec{s}) = \oplus_{i=1}^{m}(s_i \oplus s_{i+1})$ includes each s_i exactly twice and hence sums to the all-0 string, hence

$$\sum_{\vec{s}} \sum_{x \in \{0,1\}^n} (-1)^{t_1(\vec{s}) \cdot x} = \sum_{\vec{s}} \sum_{x \in \{0,1\}^n} (-1)^{0 \cdot x} = B^m \cdot 2^n.$$

For the inductive step, suppose Claim 2 is true for $r-1$. Rewrite the left-hand side of (7) as

$$\begin{aligned}
&\sum_{\vec{s}} \sum_{\substack{x^{(1)},\ldots,x^{(r)} \\ \text{different}}} (-1)^{\sum_{j=1}^{r} t_j(\vec{s}) \cdot x^{(j)}} \\
&\quad = \sum_{\vec{s}} \sum_{x^{(1)}} \sum_{\substack{x^{(2)},\ldots,x^{(r)} \\ \text{different}}} (-1)^{\sum_{j=1}^{r} t_j(\vec{s}) \cdot x^{(j)}} - \sum_{\vec{s}} \sum_{a=2}^{r} \sum_{\substack{x^{(2)},\ldots,x^{(r)} \\ \text{different, } x^{(1)} = x^{(a)}}} (-1)^{\sum_{j=1}^{r} t_j(\vec{s}) \cdot x^{(j)}}.
\end{aligned} \tag{8}$$

Let us estimate both sums of (8). Since $\sum_{x^{(1)}} (-1)^{t_1(\vec{s}) x^{(1)}}$ equals 2^n if $t_1(\vec{s}) = 0^n$, and that sum equals 0 otherwise, the first sum of (8) equals

$$2^n \sum_{\vec{s}: t_1(\vec{s}) = 0} \sum_{\substack{x^{(2)},\ldots,x^{(r)} \\ \text{different}}} (-1)^{\sum_{j=2}^{r} t_j(\vec{s}) \cdot x^{(j)}}. \tag{9}$$

We now transform this sum into the form of the left-hand side of (7), with both m and r smaller by 1 compared to their current values. After that, we will apply the induction hypothesis.

Let ℓ be such that $\ell \in I_1$, $\ell - 1 \notin I_1$. Then $t_1(\vec{s})$ contains s_ℓ with coefficient 1 (because $t_1(\vec{s})$ includes $s_\ell \oplus s_{\ell+1}$ but not $s_{\ell-1} \oplus s_\ell$). We can use the condition $t_1(\vec{s}) = 0$ to express s_ℓ in terms of $s_1, \ldots, s_{\ell-1}$ and $s_{\ell+1}, \ldots, s_m$ as follows:

$$s_\ell = s_{\ell+1} \oplus \bigoplus_{i \in I_1 : i \neq \ell} (s_i \oplus s_{i+1}). \tag{10}$$

Let b be such that $\ell - 1 \in I_b$. Then $t_b(\vec{s})$ contains $s_{\ell-1} \oplus s_\ell$ and we can substitute (10) into $t_b(\vec{s})$, obtaining

$$t_b(\vec{s}) = s_{\ell-1} \oplus s_{\ell+1} \oplus \bigoplus_{i \in I_1 : i \neq \ell} (s_i \oplus s_{i+1}) \oplus \bigoplus_{i \in I_b : i \neq \ell - 1} (s_i \oplus s_{i+1}).$$

We can now remove the variable s_ℓ (because it was only contained in $s_{\ell-1} \oplus s_\ell$ and $s_\ell \oplus s_{\ell+1}$) and redefine I_b to be $I_1 \cup I_b \setminus \{\ell\}$. Then we get that (9) is equal to

$$2^n \sum_{\substack{s_1,\ldots,s_{\ell-1} \\ s_{\ell+1},\ldots,s_m}} \sum_{\substack{x^{(2)},\ldots,x^{(r)} \\ \text{different}}} (-1)^{\sum_{j=2}^{r} t_j(\vec{s}) \cdot x^{(j)}} = 2^n \cdot O\left(B^{m-r+1} \cdot 2^{n(r-1)}\right) = O\left(B^{m-r+1} \cdot 2^{nr}\right)$$

with the estimate following from the induction hypothesis (with both m and r being smaller by 1).

As for the second sum of (8), it is equal to

$$\sum_{a=2}^{r} \sum_{\vec{s}} \sum_{\substack{x^{(2)},\ldots,x^{(r)} \\ \text{different}}} (-1)^{\sum_{j=2}^{r} t_j^{(a)}(\vec{s})\cdot x^{(j)}} = O\left(B^{m-r+2} \cdot 2^{n(r-1)}\right)$$

where $t_j^{(a)}(\vec{s}) = t_j(\vec{s})$ except for $t_a^{(a)}(\vec{s}) = t_a(\vec{s}) \oplus t_1(\vec{s})$ (thus merging the partition parts I_1 and I_a). We have eliminated $x^{(1)}$ and apply the induction hypothesis (with r being smaller by 1 and m remaining the same). The outer sum over a introduces only a factor depending on $r \leq m$.

Since $B = o(2^n)$ we have $B^{m-r+2} \cdot 2^{n(r-1)} = o(B^{m-r+1} \cdot 2^{nr})$. Hence the bound on the first sum in (8) is of a larger order and we have completed the proof of Claim 2.

Acknowledgement

We thank Loïck Magnin and Jérémie Roland for sending us a copy of [16].

References

1 A. Ambainis. A note on quantum black-box complexity of almost all Boolean functions. *Information Processing Letters*, 71(1):5–7, 1999. quant-ph/9811080.

2 A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. quant-ph/0002066.

3 A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. Earlier version in FOCS'04. quant-ph/0311001.

4 R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. quant-ph/9802049.

5 N. de Beaudrap, R. Cleve, and J. Watrous. Sharp quantum vs. classical query complexity separations. *Algorithmica*, 34(4):449–461, 2002. quant-ph/0011065.

6 A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of 43rd ACM STOC*, pages 77–84, 2012. arXiv:1105.4024.

7 H. Buhrman, N. Vereshchagin, and R. de Wolf. On computation and communication with small bias. In *Proceedings of 22nd IEEE Conference on Computational Complexity*, pages 24–32, 2007.

8 H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

9 W. van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of 39th IEEE FOCS*, pages 362–367, 1998. quant-ph/9805006.

10 D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. In *Proceedings of the Royal Society of London*, volume A439, pages 553–558, 1992.

11 C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. Earlier version in ICALP'04.

12 E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81:5442–5444, 1998. quant-ph/9802045.

13 L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999. Earlier version in Complexity'98. Also cs.CC/9811023.

14 L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

15 P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proceedings of 39th ACM STOC*, pages 526–535, 2007. quant-ph/0611054.

16 L. Magnin and J. Roland. Explicit relation between all lower bound techniques for quantum query complexity. In *Proceedings of 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, 2013. arXiv:1209.2713.

17 R. O'Donnell and R. Servedio. Extremal properties of polynomial threshold functions. *Journal of Computer and System Sciences*, 74(3):298–312, 2008. Earlier version in Complexity'03.

18 M. Santha. Quantum walk based search algorithms. In *Proceedings of 5th TAMC*, pages 31–46, 2008. arXiv/0808.0059.

19 P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94. quant-ph/9508027.

20 D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. Earlier version in FOCS'94.