Lower Bounds on Quantum Query Complexity for Read-Once Formulas with XOR and MUX Operators*

Hideaki FUKUHARA[†], Nonmember and Eiji TAKIMOTO^{††a)}, Member

SUMMARY We introduce a complexity measure r for the class $\mathcal{F}$ of read-once formulas over the basis {AND, OR, NOT, XOR, MUX} and show that for any Boolean formula F in the class $\mathcal{F}$, $r(F)$ is a lower bound on the quantum query complexity of the Boolean function that F represents. We also show that for any Boolean function f represented by a formula in $\mathcal{F}$, the deterministic query complexity of f is only quadratically larger than the quantum query complexity of f . Thus, the paper gives further evidence for the conjecture that there is an only quadratic gap for all functions.

key words: quantum query complexity, read-once formulas, decision trees, adversary method

1. Introduction

The biggest challenge in quantum computation theory is to clarify the extent to which quantum computation gives an advantage over classical deterministic and randomized computation. We know some quantum algorithms that would be significantly faster for a few specific problems. But many results on the limitation of quantum computation suggest that quantum computers may outperform the classical ones only modestly (see, e.g., [1]). Most of the interesting results have been shown in the *query model* of computation. In the model, the algorithm has access to a black-box containing an input assignment $x = (x_1, x_2, \dots, x_n)$ for some n variable function f , and is required to compute $f(x)$ using as few queries to the black-box as possible. In each query, the algorithm may ask for a single bit assigned to a variable of some index i , $1 \leq i \leq n$, and the value x_i is returned. The deterministic query complexity of f , denoted $D(f)$, is defined as the minimum of the number of queries to compute $f(x)$ for a worst-case input x , where the minimization is taken over all deterministic algorithms. In the *quantum query model*, queries may be quantum coherent, which means that the algorithm may superpose different query requests i with complex amplitudes α_i , and then receive a superposition of the corresponding values x_i . We consider the *bounded error setting*, where the computation may have a small probability of being incorrect. The quantum query complexity of f , de-

noted $Q_\epsilon(f)$, is defined similarly to the deterministic one, but now the minimization is taken over all quantum algorithms that are allowed to output incorrect value $\neg f(x)$ with probability at most ϵ for every input x . It is of great interest to compare the computational powers of the two models. Note that $Q_\epsilon(f) \leq D(f)$ for any ϵ , which implies that the quantum query model is at least as powerful as the classical model.

One of the major algorithmic results in this regard is Grover's search algorithm [9], which can be viewed as a quantum algorithm for computing the n -bit OR function with $O(\sqrt{n})$ queries. This contrasts with the fact that n queries are required for deterministic query algorithms. In other words, for the n -bit OR function, denoted OR_n , we have $Q_\epsilon(\text{OR}_n) = O(\sqrt{n})$ for any $\epsilon \in (0, 1/2)$, while $D(\text{OR}_n) = n$. Thus the OR function provides an example where quantum computation gives a quadratic speedup over deterministic computation. Note that the value of ϵ only affects $Q_\epsilon(f)$ within a constant factor, unless $\epsilon = 0$.

On the other hand, Barnum and Saks [5] show that for any read-once function f of n variables, $Q_\epsilon(f) = \Omega(\sqrt{n})$, where a read-once function is a Boolean function that can be represented by a Boolean formula over the basis {AND, OR, NOT} such that each variable appears only once. Note that the OR function is a read-once function. This result implies that more than quadratic speedup of quantum computation over classical deterministic algorithms is impossible for read-once functions. Actually, no such speedup result is known for *any* Boolean function.

So, an important problem in quantum query complexity is to resolve the following conjecture.

Conjecture 1: For any Boolean function f and $\epsilon \in [0, 1/2)$, $Q_\epsilon(f) = \Omega(D(f)^{1/2})$.

Note that the conjecture is only for *total* functions whose domain is all of $\{0, 1\}^n$. For partial functions whose domain is restricted, there are much better speedups known [15]. The best known result toward this conjecture says that for any Boolean function f , $Q_\epsilon(f) = \Omega(D(f)^{1/6})$ for $\epsilon \in (0, 1/2)$ [7], and $Q_\epsilon(f) = \Omega(D(f)^{1/3})$ for $\epsilon = 0$ [14]. Several results provide evidence for the conjecture. For example, the square-root bound holds for all symmetric functions, and for some particular functions f such as the PARITY and the MAJORITY functions, $Q_\epsilon(f) = \Omega(D(f))$ [7].

In this paper, we provide further evidence for the conjecture. We show that the square-root bound holds for any function f that can be represented by a read-once formula

Manuscript received March 23, 2009.

Manuscript revised June 30, 2009.

[†]The author is with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980–8579 Japan.

^{††}The author is with the Department of Informatics, Kyushu University, Fukuoka-shi, 819–0395 Japan.

*The previous version of the paper was presented at Computing: The Australasian Theory Symposium (CATS 2009) [10].

a) E-mail: eiji@i.kyushu-u.ac.jp

DOI: 10.1587/transinf.E93.D.280

over the basis $B = \{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{MUX}\}$, where XOR is the two-variable exclusive-or function and MUX is the three variable multiplexer function. Note that since the functions XOR and MUX cannot be represented by read-once formulas over $\{\text{AND}, \text{OR}, \text{NOT}\}$, the class $\mathcal{F}$ of read-once formulas over the basis B properly contains the class of read-once formulas over $\{\text{AND}, \text{OR}, \text{NOT}\}$. Moreover, due to MUX included in the basis, $\mathcal{F}$ properly contains the class of read-once decision trees as well. Therefore, we can say that we significantly extend the result for read-once functions by [5].

To derive the square-root bound, we introduce a complexity measure r for the class $\mathcal{F}$ of read-once formulas over the basis B and show that $Q_\epsilon(f) = \Omega(r(F))$ and $r(F) \geq \sqrt{D(f)}$ for any function f that can be represented by a formula F in $\mathcal{F}$. The measure r is a generalization of the *soft rank* of decision trees, which is introduced in the previous version of the paper [10], where we show the same bounds hold for the class of read-once decision trees. Since $\mathcal{F}$ properly contains the class of read-once decision trees, this paper is an extension of the previous one.

To derive the lower bound $Q_\epsilon(f) = \Omega(r(F))$, we employ the adversary method, which is one of the most successful techniques for showing lower bounds on quantum query complexity [2], [3], [6], [13], [16]. There are several equivalent formulations of the adversary method. We use the setting of the *spectral formulation*, since it behaves well with respect to function composition. The spectral adversary method is formulated as an optimization problem specified by the function f under consideration, and its solution, denoted by $\text{ADV}(f)$, gives a lower bound on $Q_\epsilon(f)$. Thus, $\text{ADV}(f)$ is often referred to as the *adversary bound*. Høyer et al. consider the adversary bound for composite functions of the form $h = f \circ (g_1, g_2, \dots, g_k)$ and give an exact expression for $\text{ADV}(h)$ in terms of the adversary bounds of f and g_i for $1 \leq i \leq k$ [11]. More precisely, they generalize the adversary method and introduce a quantity $\text{ADV}_\alpha(f)$ with a cost vector α , and show that $\text{ADV}(h) = \text{ADV}_\alpha(f)$ with $\alpha = (\text{ADV}(g_1), \text{ADV}(g_2), \dots, \text{ADV}(g_k))$, provided that the sets X_i of input variables for g_i are disjoint from each other. We apply the composition theorem to obtain a lower bound on ADV for read-once formulas over $\{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{MUX}\}$ based on the observation that the formulas consist of recursively defined composite functions with the basis functions.

The rest of the paper is organized as follows.

In Sect. 2 we define the class $\mathcal{F}$ of read-once formulas over $\{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{MUX}\}$ and give the definition of the complexity measure r . In Sect. 3 we state the quantum query model and review the adversary method with some useful results including the composition theorem. In Sect. 4 we give bounds on $\text{ADV}_\alpha(\text{XOR})$ and $\text{ADV}_\alpha(\text{MUX})$ in terms of α . These are the main technical contribution of the paper. We give in Sect. 5 adversary bounds for read-once formulas in $\mathcal{F}$. We also state the gap between deterministic and quantum query complexities. In Sect. 6 we apply the results to read-once decision trees and give adversary bounds in terms

of the soft rank and depth of decision trees. In Sect. 7 we state some concluding remarks.

2. Preliminaries

Let XOR denote the two-variable exclusive-or function and MUX denote the three-variable multiplexer function. The multiplexer function MUX outputs the second or third input bit that is singled out by the first bit. More formally,

$$\text{XOR}(x_1, x_2) = \bar{x}_1 x_2 \vee x_1 \bar{x}_2,$$

$$\text{MUX}(x_1, x_2, x_3) = \bar{x}_1 x_2 \vee x_1 x_3.$$

Let $\mathcal{F}$ denote the set of read-once Boolean formulas over the basis $B = \{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{MUX}\}$, where a formula is read-once if every variable appears at most once in the formula. For each formula F in $\mathcal{F}$, we may assume without loss of generality that NOT gates appear as negated input literals.

We introduce two complexity measures r and d for Boolean formulas over the basis B , which are recursively defined as follows.

Definition 1: Let F be a formula in $\mathcal{F}$.

1. If F is a single literal, then $r(F) = d(F) = 1$.
2. If $F = G_1 \wedge G_2$ or $F = G_1 \vee G_2$ for some formulas G_1 and G_2 in $\mathcal{F}$, then

$$r(F) = \sqrt{r(G_1)^2 + r(G_2)^2}$$

and

$$d(F) = d(G_1) + d(G_2).$$

3. If $F = \text{XOR}(G_1, G_2)$ for some G_1 and G_2 , then

$$r(F) = r(G_1) + r(G_2)$$

and

$$d(F) = d(G_1) + d(G_2).$$

4. If $F = \text{MUX}(G_1, G_2, G_3)$ for some G_1, G_2 and G_3 , then

$$r(F) = \min\{r(G_2), r(G_3)\} + \sqrt{r(G_1)^2 + (r(G_2) - r(G_3))^2}$$

and

$$d(F) = d(G_1) + \max\{d(G_2), d(G_3)\}.$$

Later we will show that for any read-once formula F in $\mathcal{F}$, $r(F)$ is a lower bound on the quantum query complexity of F and $d(F)$ is an upper bound on the deterministic query complexity of F .

3. Quantum Query Model

In the query model of computation, we wish to compute some Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with access

to the input through queries. More precisely, the algorithm has access to a black-box (or an oracle) containing an input assignment $x = (x_1, x_2, \dots, x_n)$ and is required to compute $f(x)$ using as few queries to the black-box as possible. In each query, the algorithm may ask for a single bit assigned to a variable of some index i , $1 \leq i \leq n$, and the value x_i is returned. The *deterministic query complexity* of f , denoted $D(f)$, is defined as the minimum of the number of queries to compute $f(x)$ for a worst-case input x , where the minimization is taken over all deterministic algorithms.

In the quantum query model, we can make queries in superposition. Formally, a query O corresponds to the unitary transformation

$$O : |i, b, z\rangle \mapsto |i, b \oplus x_i, z\rangle$$

where $i \in \{1, 2, \dots, n\}$, $b \in \{0, 1\}$, and z represents the workspace. A t -query quantum algorithm A has the form $A = U_t O U_{t-1} O \dots O U_1 O U_0$, where the U_k are fixed unitary transformations independent of the input x . The computation begins in the state $|0\rangle$, and the result of the computation of A is the observation of the rightmost bit of $A|0\rangle$. We say that A ϵ -approximates f if the observation of the rightmost bit of $A|0\rangle$ is equal to $f(x)$ with probability at least $1 - \epsilon$, for every x . The ϵ -error quantum query complexity of f , denoted $Q_\epsilon(f)$, is defined as the minimum of the number of queries to ϵ -approximate f , where the minimization is taken over all quantum query algorithms. More background on quantum query complexity can be found in [4].

3.1 Adversary Bound

Along with the polynomial method [7], the adversary method is one of the main techniques for deriving lower bounds on quantum query complexity. There are several formulations of the adversary method. We use the setting of the spectral formulation of the adversary method.

In what follows, when we refer to a matrix, it means, unless otherwise stated, a square matrix of size $2^n \times 2^n$ for some integer n , of which rows and columns are indexed by n -bit strings. For a binary string $x \in \{0, 1\}^n$, x_i denotes the i -th bit of x . The set of real numbers is denoted by $\mathbb{R}$ and the set of real positive numbers is denoted by $\mathbb{R}_+$. For a Boolean function f , $\bar{f}$ denotes the negation function of f , that is, for any input x , $\bar{f}(x) = 1$ if $f(x) = 0$ and $\bar{f}(x) = 0$ if $f(x) = 1$.

For a matrix A , $A[x, y]$ denotes its (x, y) element and A^* denotes its conjugate transpose. For a column vector v , let $|v|$ denote the l_2 -norm of v , that is, $|v| = \sqrt{v^* v}$. For a square matrix A , $\|A\|$ denotes the spectral norm of A , that is,

$$\begin{aligned} \|A\| &= \sqrt{(\text{maximum eigenvalue of } A^* A)} \\ &= \max_{v: |v| \neq 0} \frac{|Av|}{|v|}. \end{aligned}$$

Note that for any real symmetric matrix A , its spectral norm equals the maximum absolute eigenvalue of A . Let $A \circ B$ denote the Hadamard product of A and B , that is, $(A \circ B)[x, y] = A[x, y]B[x, y]$. A Boolean matrix is a matrix

whose elements are 0 or 1. For a Boolean matrix A , $\bar{A}$ gives the element-wise negation of A , that is, $\bar{A}[x, y] = 1 - A[x, y]$.

We say that a matrix Γ is an adversary matrix for a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ if Γ satisfies the following conditions:

1. Γ is a symmetric matrix;
2. $\Gamma[x, y] \geq 0$ for every $x, y \in \{0, 1\}^n$;
3. $\Gamma[x, y] > 0$ for some $x, y \in \{0, 1\}^n$; and
4. $\Gamma[x, y] = 0$ if $f(x) \neq f(y)$.

Let D_i be a Boolean matrix defined by $D_i[x, y] = 1$ if and only if bitstrings x and y differ in the i -th coordinate.

The spectral adversary method is formulated as an optimization problem specified by the function f under consideration, and its solution, denoted by $\text{ADV}(f)$, gives a lower bound on $Q_\epsilon(f)$ [6]. Thus, $\text{ADV}(f)$ is referred to as the *adversary bound* of f .

Definition 2: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and Φ_f be the set of adversary matrices for f . $\text{ADV}(f)$ is defined as

$$\text{ADV}(f) = \max_{\Gamma \in \Phi_f} \min_{i \in \{1, 2, \dots, n\}} \frac{\|\Gamma\|}{\|\Gamma \circ D_i\|}.$$

Theorem 1 ([6]): For any Boolean function f and any $\epsilon \in [0, 1/2]$,

$$Q_\epsilon(f) \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2} \text{ADV}(f).$$

3.2 Adversary Bounds for Composite Functions

Høyer et al. generalize the adversary method and introduce a quantity ADV_α with a cost vector α [11].

Definition 3: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and Φ_f be the set of adversary matrices for f . For every vector $\alpha \in \mathbb{R}_+^n$, $\text{ADV}_\alpha(f)$ is defined as

$$\text{ADV}_\alpha(f) = \max_{\Gamma \in \Phi_f} \min_{i \in \{1, 2, \dots, n\}} \frac{\alpha_i \|\Gamma\|}{\|\Gamma \circ D_i\|}.$$

Note that when $\alpha = (1, 1, \dots, 1)$, $\text{ADV}_\alpha(f) = \text{ADV}(f)$. It is clear from Definition 3 that ADV_α is monotonically increasing with respect to α . For two vectors $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \dots, \beta_n)$, we write $\alpha \leq \beta$ if $\alpha_i \leq \beta_i$ for any index $i \in \{1, 2, \dots, n\}$.

Proposition 1: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$. For any vectors α and β in $\mathbb{R}_+^n$ such that $\alpha \leq \beta$,

$$\text{ADV}_\alpha(f) \leq \text{ADV}_\beta(f).$$

Moreover, ADV_α has the following obvious but useful properties.

Proposition 2: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $\alpha \in \mathbb{R}_+^n$. Then, for any $c \in \mathbb{R}_+$,

$$\text{ADV}_{c\alpha}(f) = c \text{ADV}_\alpha(f).$$

Proposition 3: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{R}_+^n$. For a permutation π over the set $\{1, 2, \dots, n\}$, $\pi(f)$ denotes the function defined as $f(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$, and $\pi(\alpha)$ denotes the vector defined as $(\alpha_{\pi(1)}, \alpha_{\pi(2)}, \dots, \alpha_{\pi(n)})$. For any permutation π over $\{1, 2, \dots, n\}$,

$$\text{ADV}_{\pi(\alpha)}(\pi(f)) = \text{ADV}_\alpha(f).$$

Proposition 4: Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{R}_+^n$. For a bit $v \in \{0, 1\}$, let $v^0 = v$ and $v^1 = \bar{v}$. For any function f' defined as $f'(x_1, x_2, \dots, x_n) = f(x_1^{b_1}, x_2^{b_2}, \dots, x_n^{b_n})$ for some binary sequence $b = (b_1, b_2, \dots, b_n) \in \{0, 1\}^n$,

$$\text{ADV}_\alpha(f') = \text{ADV}_\alpha(f).$$

Høyer et al. consider the adversary bound for composite functions of the form $h = f \circ (g_1, g_2, \dots, g_k)$ and give an exact expression for $\text{ADV}(h)$ in terms of $\text{ADV}_\alpha(f)$ and $\text{ADV}(g_i)$ for $1 \leq i \leq k$ [11].

Theorem 2 ([11]): Let h be a Boolean function of n variables given by

$$h(x) = f(g_1(x^1), g_2(x^2), \dots, g_k(x^k))$$

for some Boolean function $f : \{0, 1\}^k \rightarrow \{0, 1\}$ and k Boolean functions $g_1 : \{0, 1\}^{n_1} \rightarrow \{0, 1\}, \dots, g_k : \{0, 1\}^{n_k} \rightarrow \{0, 1\}$, where x is the concatenation of the k binary sequences $x^1 \in \{0, 1\}^{n_1}, x^2 \in \{0, 1\}^{n_2}, \dots, x^k \in \{0, 1\}^{n_k}$. Then,

$$\text{ADV}(h) = \text{ADV}_\alpha(f)$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_k)$ with $\alpha_i = \text{ADV}(g_i)$.

One limitation of the theorem above is that we require the sub-functions g_i to act on disjoint subsets of the input bits.

The usefulness of Theorem 2 is that it reduces the problem of computing the adversary bound for a complicated function h into a few subproblems of computing the adversary bounds for functions of smaller size. In fact, Høyer et al. give a simple proof of the $\sqrt{n}$ adversary bound for read-once functions [11]. More precisely, they first show that for two-variable AND and OR functions,

$$\text{ADV}_\alpha(\text{AND}) = \text{ADV}_\alpha(\text{OR}) = \sqrt{\alpha_1^2 + \alpha_2^2} \quad (1)$$

for any $\alpha = (\alpha_1, \alpha_2) \in \mathbb{R}_+^2$, and then apply Theorem 2 to derive the bound.

Unlike the cases of the AND and OR functions, it seems very hard to represent $\text{ADV}_\alpha(f)$ as a closed form expression of α for most functions f . The main contribution of this paper is that we give closed form expressions for $\text{ADV}_\alpha(\text{XOR})$ and $\text{ADV}_\alpha(\text{MUX})$.

Below we give a *dual* version of the spectral adversary formulation. The dual version is an equivalent expression for ADV_α in terms of a minimization problem. Note that since the primal formulation is expressed as a maximization problem, any feasible (not necessarily optimal) solution

gives a lower bound on ADV_α . Similarly, any feasible solution for the dual formulation gives an upper bound. We will use the both formulations to derive almost tight bounds on ADV_α .

Definition 4: Let $p : \{0, 1\}^n \times \{1, 2, \dots, n\} \rightarrow \mathbb{R}$ be a set of probability distributions in the sense that $p_x(i) \geq 0$ and $\sum_{i \in \{1, 2, \dots, n\}} p_x(i) = 1$ for every $x \in \{0, 1\}^n$. Let P_n denote the set of all such sets p . For a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we define

$$\text{MM}_\alpha(f) = \min_{p \in P_n} \max_{\substack{x \in f^{-1}(0) \\ y \in f^{-1}(1)}} \frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i},$$

where $f^{-1}(c) = \{x \in \{0, 1\}^n \mid f(x) = c\}$ for $c \in \{0, 1\}$.

Theorem 3 ([11]): For every $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $\alpha \in \mathbb{R}_+^n$,

$$\text{ADV}_\alpha(f) = \text{MM}_\alpha(f).$$

4. $\text{ADV}_\alpha(\text{XOR})$ and $\text{ADV}_\alpha(\text{MUX})$

In this section, we show that $\text{ADV}_\alpha(\text{XOR})$ is exactly expressed as the l_1 -norm of α . Moreover, we give an almost tight bound on $\text{ADV}_\alpha(\text{MUX})$ in a closed form.

Theorem 4: For any $\alpha = (a, b) \in \mathbb{R}_+^2$,

$$\text{ADV}_\alpha(\text{XOR}) = a + b.$$

Proof. We first show that $\text{ADV}_\alpha(\text{XOR}) \leq a + b$. We set up the set of probability distribution p as follows. For any $x \in \{0, 1\}^2$,

$$p_x(i) = \begin{cases} \frac{a}{a+b} & \text{if } i = 1, \\ \frac{b}{a+b} & \text{if } i = 2. \end{cases}$$

It is clear that p is a feasible solution of the optimization problem for $\text{MM}_\alpha(\text{XOR})$. It is easy to check that for any pair $(x, y) \in \{(00, 01), (00, 10), (11, 01), (11, 10)\}$ of negative and positive inputs for f ,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i} = a + b.$$

Therefore, from Theorem 3 $\text{ADV}_\alpha(\text{XOR}) = \text{MM}_\alpha(\text{XOR}) \leq a + b$.

Next we show that $\text{ADV}_\alpha(\text{XOR}) \geq a + b$. We set up the following adversary matrix Γ for XOR.

	00	01	10	11
00	0	b	a	0
01	b	0	0	a
10	a	0	0	b
11	0	a	b	0

It is easy to check that Γ given above is a feasible solution of the optimization problem for $\text{ADV}_\alpha(\text{XOR})$. The characteristic polynomial for Γ is

$$|\Gamma - xI| = (x + a - b)(x + a + b)(x - a + b)(x - a - b).$$

So $a + b$ is the maximum absolute eigenvalue of Γ , and we thus have $\|\Gamma\| = a + b$. Similarly, since the characteristic polynomials for $\Gamma \circ D_1$ and $\Gamma \circ D_2$ are

$$|\Gamma \circ D_1 - xI| = (x - a)^2(x + a)^2,$$

$$|\Gamma \circ D_2 - xI| = (x - b)^2(x + b)^2,$$

we have $\|\Gamma \circ D_1\| = a$ and $\|\Gamma \circ D_2\| = b$. Therefore, for any $i \in \{1, 2\}$,

$$\frac{\alpha_i \|\Gamma\|}{\|\Gamma \circ D_i\|} = a + b,$$

which implies that $\text{ADV}_\alpha(\text{MUX}) \geq a + b$. $\square$

For the multiplexer function MUX, we have not succeeded to derive the exact expression of $\text{ADV}_\alpha(\text{MUX})$. Instead, we give lower and upper bounds that match up to an additive factor of $O(l\alpha_1^2/(l^2 + \alpha_1^2))$, where $l = |\alpha_2 - \alpha_3|$. Note that since $\text{MUX}(x_1, x_2, x_3) = \text{MUX}(\bar{x}_1, x_3, x_2)$, Propositions 3 and 4 imply that

$$\text{ADV}_{(a,c,b)}(\text{MUX}) = \text{ADV}_{(a,b,c)}(\text{MUX})$$

for any $\alpha = (a, b, c) \in \mathbb{R}_+^3$. So, we may assume without loss of generality that the cost vector $\alpha = (a, b, c)$ satisfies that $b \leq c$.

Theorem 5: For any $\alpha = (a, b, c) \in \mathbb{R}_+^3$ such that $b \leq c$,

$$\text{ADV}_\alpha(\text{MUX}) \geq b + \sqrt{(c - b)^2 + a^2}.$$

Proof. First we show that it suffices to prove the theorem only for the case where $a = 1$, i.e.,

$$\text{ADV}_\alpha(\text{MUX}) \geq b + \sqrt{(c - b)^2 + 1} \quad (2)$$

for $\alpha = (1, b, c)$ with $b \leq c$. This is because, for any $\alpha = (a, b, c) \in \mathbb{R}_+^3$ such that $b \leq c$, Proposition 2 says that $\text{ADV}_\alpha(\text{MUX}) = a\text{ADV}_\beta(\text{MUX})$ for $\beta = (1, b/a, c/a)$, from which together with (2) the theorem follows.

Now we show (2) by constructing an adversary matrix Γ for MUX so that it satisfies the following four conditions.

$$\|\Gamma\| \geq b + \sqrt{(c - b)^2 + 1} \quad (3)$$

$$\|\Gamma \circ D_1\| = 1, \quad (4)$$

$$\|\Gamma \circ D_2\| = b, \quad (5)$$

$$\|\Gamma \circ D_3\| = c. \quad (6)$$

By the definition of ADV_α , (2) follows from (3), (4), (5) and (6). We give in Fig. 1 such an adversary matrix Γ . (It is easy to verify that Γ is actually an adversary matrix for MUX.) In the following we verify that each of the conditions (3), (4), (5) and (6) holds.

First we verify (3). Define a matrix A as

$$A = \begin{bmatrix} b & 0 & 1 \\ 0 & 0 & \sqrt{c(c-b)} \\ 1 & \sqrt{c(c-b)} & b \end{bmatrix} \quad (7)$$

	000	001	010	011	100	101	110	111
000	0	0	0	0	0	0	0	0
001	0	0	b	0	0	1	0	0
010	0	b	0	0	0	0	1	0
011	0	0	0	0	0	0	0	0
100	0	0	0	0	0	W	0	0
101	0	1	0	0	W	0	b	0
110	0	0	1	0	0	b	0	W
111	0	0	0	0	0	0	W	0

Fig. 1 The adversary matrix Γ for MUX, where $W = \sqrt{c(c-b)}$.

and let λ_1, λ_2 and λ_3 be the three eigenvalues of A . We show that the following six values $\lambda_1, -\lambda_1, \lambda_2, -\lambda_2, \lambda_3, -\lambda_3$ are eigenvalues of Γ . Note that the rest two eigenvalues of Γ are both 0 with corresponding eigenvectors $(1, 0, 0, 0, 0, 0, 0)^T$ and $(0, 0, 0, 1, 0, 0, 0)^T$. Let λ be an eigenvalue of A and $v = (v_1, v_2, v_3)^T$ be the corresponding eigenvector, i.e., $Av = \lambda v$. It is easy to check that $v' = (0, v_1, v_1, 0, v_2, v_3, v_3, v_2)^T$ and $v'' = (0, v_1, -v_1, 0, v_2, -v_3, v_3, -v_2)^T$ are eigenvectors of Γ corresponding to the eigenvalues λ and $-\lambda$, respectively. So the largest absolute eigenvalue of Γ equals the largest absolute eigenvalue of A . Namely, $\|\Gamma\| = \|A\|$. The characteristic polynomial of A is

$$\begin{aligned} p_A(x) &= |A - xI| \\ &= -x^3 + 2bx^2 + (c(c-b) - b^2 + 1)x - bc(c-b), \end{aligned}$$

whose roots are eigenvalues of A . We show that $p_A(x_0) > 0$ for

$$x_0 = b + \sqrt{(c-b)^2 + 1}.$$

Since $p_A(+\infty) = -\infty$, x_0 is a lower bound on the largest root of p_A , and thus a lower bound on $\|\Gamma\|$ as required.

$$p_A(x_0) = b(c-b)(\sqrt{(c-b)^2 + 1} - (c-b)) \geq 0,$$

since $c \geq b$.

Next we verify (4). By a similar argument to the one stated above, the largest absolute eigenvalue of $\Gamma \circ D_1$ equals the largest absolute eigenvalue of the matrix B defined as

$$B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Obviously, $\|\Gamma \circ D_1\| = \|B\| = 1$.

Next we verify (5). The largest absolute eigenvalue of $\Gamma \circ D_2$ equals the largest absolute eigenvalue of the matrix C defined as

$$C = \begin{bmatrix} b & 0 \\ 0 & b \end{bmatrix}.$$

Obviously, $\|\Gamma \circ D_2\| = \|C\| = b$.

Finally we verify (6). The largest absolute eigenvalue of $\Gamma \circ D_3$ equals the largest absolute eigenvalue of the matrix D defined as

$$D = \begin{bmatrix} b & 0 & 0 \\ 0 & 0 & \sqrt{c(c-b)} \\ 0 & \sqrt{c(c-b)} & b \end{bmatrix}.$$

Obviously, $\|\Gamma \circ D_3\| = \|D\| = c$. $\square$

An upper bound on $\text{ADV}_\alpha(\text{MUX})$ we give in the following theorem says that the lower bound of Theorem 5 is almost tight.

Theorem 6: For any $\alpha = (a, b, c) \in \mathbb{R}_+^3$ such that $b \leq c$,

$$\text{ADV}_\alpha(\text{MUX}) \leq \frac{b + c + \sqrt{(c - b)^2 + 4a^2}}{2}$$

Proof. As in the proof of Theorem 5, it suffices to prove the theorem only for the case where $a = 1$. That is, we will prove the following.

$$\text{ADV}_\alpha(\text{MUX}) \leq \frac{b + c + \sqrt{(c - b)^2 + 4}}{2} \quad (8)$$

for any $\alpha = (1, b, c) \in \mathbb{R}_+^3$ such that $b \leq c$.

To show (8), we use the dual formulation of the spectral adversary method. In other words, we construct a feasible solution of the optimization problem $\text{MM}_\alpha(\text{MUX})$, from which we derive an upper bound on $\text{ADV}_\alpha(\text{MUX})$.

Put $A = \frac{b+c+\sqrt{(c-b)^2+4}}{2}$. We set up a set of probability distribution p as follows. For every $x \in \{0, 1\}^3$,

$$p_x(i) = \begin{array}{c|ccc} x \setminus i & 1 & 2 & 3 \\ \hline 000, 001, 010, 011 & d & 1-d & 0 \\ 100, 111 & f & g & 1-f-g \\ 110, 101 & j & 0 & 1-j \end{array}$$

where

$$\begin{aligned} d &= 1 - \frac{b}{A}, \\ f &= \frac{b^2(A-b)}{A(bA-b^2+1)^2}, \\ g &= \frac{b}{A(bA-b^2+1)^2}, \\ j &= \frac{1}{A^2-bA}. \end{aligned}$$

It is easy to check that p is a feasible solution of $\text{MM}_\alpha(\text{MUX})$, that is, p_x is a probability distribution for each input x . What we need to show is that for any pair $(x, y) \in \text{MUX}^{-1}(0) \times \text{MUX}^{-1}(1)$ of negative and positive inputs of MUX,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} \leq A. \quad (9)$$

Note that $\alpha_1 = 1$, $\alpha_2 = b$, and $\alpha_3 = c$ by our choice of $\alpha = (1, b, c)$. We partition the set $\text{MUX}^{-1}(0) \times \text{MUX}^{-1}(1)$ consisting of 16 pairs into six classes and verify (9) for each class.

For the case where $(x, y) \in \{(000, 010), (000, 011), (001, 010), (001, 011)\}$,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \frac{b}{1-d} = A.$$

For the case where $(x, y) \in \{(000, 101), (001, 101), (110, 010), (110, 011)\}$,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \frac{1}{\sqrt{dj}} = A.$$

For the case where $(x, y) \in \{(000, 111), (001, 111), (100, 010), (100, 011)\}$,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \left(\sqrt{df} + \frac{\sqrt{(1-d)g}}{b} \right)^{-1} = A.$$

For the case where $(x, y) \in \{(100, 101), (110, 111)\}$,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \frac{c}{\sqrt{(1-f-g)(1-j)}} \leq A.$$

To see why the last inequality holds, we show that

$$\begin{aligned} & \frac{A \sqrt{(1-f-g)(1-j)}}{c} \\ &= \frac{\sqrt{A-b-1} \sqrt{A-b+1} \sqrt{bA+1}}{\sqrt{c} \sqrt{c-b} \sqrt{bA-b^2+1}} \\ &\geq 1. \end{aligned}$$

The inequality holds because

$$\begin{aligned} & (A-b-1)(A-b+1)(bA+1) \\ & - c(c-b)(bA-b^2+1) \\ &= \frac{1}{2}((c-b)\sqrt{(c-b)^2+4} - (c-b)^2) \geq 0. \end{aligned}$$

For the case where $(x, y) = (100, 111)$, we let

$$\begin{aligned} w &= \sqrt{c^2 - 2b^3c + 4}, \\ z &= b^2c^2 + (2b - b^3)c. \end{aligned}$$

Then

$$\begin{aligned} & A \left(\frac{g}{b} + \frac{1-f-g}{c} \right) \\ &= \frac{b^2c^3 + w(z+1) + (b^4+3)c + 3b}{b^2c^3 + wz + (b^4+3)c} \\ &\geq 1 \end{aligned}$$

Consequently,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \left(\frac{g}{b} + \frac{1-f-g}{c} \right)^{-1} \leq A.$$

For the case where $(x, y) = (110, 101)$,

$$\frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i)p_y(i)/\alpha_i}} = \frac{c}{1-j} = A.$$

$\square$

By an easy calculation, we can see that our lower bound and upper bounds on $\text{ADV}_\alpha(\text{MUX})$ match up to a small constant factor (at most 1.2) for any α . Furthermore, the two bounds coincide when $b = c$. In other words, we have an exact expression of $\text{ADV}_{(a,b,c)}(\text{MUX})$ when $b = c$. Specifically, $\text{ADV}_{(a,b,b)} = a + b$.

5. Adversary Bounds for $\mathcal{F}$

We apply Theorem 2 together with Theorems 4 and 5 to obtain a lower bound on ADV for formulas in the class $\mathcal{F}$ of read-once formulas over the basis $\{\text{AND}, \text{OR}, \text{NOT}, \text{XOR}, \text{MUX}\}$.

Theorem 7: For any formula F in $\mathcal{F}$,

$$\text{ADV}(F) \geq r(F).$$

Proof. We prove $\text{ADV}(F) \geq r(F)$ by induction on the depth of F .

(Basis) If the depth of F is 0, i.e., F is a literal, then clearly $\text{ADV}(F) = r(F) = 1$.

(Induction step) If the depth of F is greater than 1, then we should consider three cases, where $F = G_1 \vee G_2$ or $F = G_1 \wedge G_2$ (Case 1), $F = \text{XOR}(G_1, G_2)$ (Case 2) and $F = \text{MUX}(G_1, G_2, G_3)$ (Case 3).

First we examine Case 1, i.e., $F = G_1 \vee G_2$ or $F = G_1 \wedge G_2$. By Theorem 2 and (1) we have

$$\text{ADV}(F) = \sqrt{\text{ADV}(G_1)^2 + \text{ADV}(G_2)^2}. \quad (10)$$

From induction hypothesis, we have

$$\text{ADV}(G_1) \geq r(G_1), \quad \text{ADV}(G_2) \geq r(G_2) \quad (11)$$

and by the definition of r , we have

$$r(F) = \sqrt{r(G_1)^2 + r(G_2)^2}. \quad (12)$$

Plugging (11) and (12) into (10), we have

$$\text{ADV}(F) \geq r(F),$$

as required.

Next we examine Case 2, i.e., $F = \text{XOR}(G_1, G_2)$. Applying Theorem 2, we have

$$\text{ADV}(F) = \text{ADV}_\alpha(\text{XOR})$$

with $\alpha = (\text{ADV}(G_1), \text{ADV}(G_2))$, and so Theorem 4 says that

$$\text{ADV}(F) = \text{ADV}(G_1) + \text{ADV}(G_2). \quad (13)$$

From induction hypothesis, we have

$$\text{ADV}(G_1) \geq r(G_1), \quad \text{ADV}(G_2) \geq r(G_2) \quad (14)$$

and by the definition of r , we have

$$r(F) = r(G_1) + r(G_2). \quad (15)$$

Plugging (14) and (15) into (13), we have

$$\text{ADV}(F) \geq r(F),$$

as required.

Finally we examine Case 3, i.e., $F = \text{MUX}(G_1, G_2, G_3)$. We may assume without loss of generality that $\text{ADV}(G_2) \leq \text{ADV}(G_3)$, since otherwise we can examine the equivalent function $\text{MUX}(\tilde{G}_1, G_3, G_2)$.

Applying Theorem 2, we have

$$\text{ADV}(F) = \text{ADV}_\alpha(\text{MUX}) \quad (16)$$

with $\alpha = (\text{ADV}(G_1), \text{ADV}(G_2), \text{ADV}(G_3))$. From induction hypothesis, we have

$$\text{ADV}(G_i) \geq r(G_i) \text{ for every } i, 1 \leq i \leq 3. \quad (17)$$

Now we consider two subcases where $r(G_2) \leq r(G_3)$ and $r(G_3) \leq r(G_2)$.

For the subcase where $r(G_2) \leq r(G_3)$, we let $\beta = (r(G_1), r(G_2), r(G_3))$. By (17) we have $\alpha \geq \beta$. So Proposition 1 implies that

$$\text{ADV}_\alpha(\text{MUX}) \geq \text{ADV}_\beta(\text{MUX}),$$

and by Theorem 5 we have

$$\begin{aligned} \text{ADV}_\beta(\text{MUX}) &\geq r(G_2) + \sqrt{(r(G_3) - r(G_2))^2 + r(G_1)^2} \\ &= r(F), \end{aligned} \quad (18)$$

where the last equality is from the definition of r , since we have assumed $r(G_2) \leq r(G_3)$. By (16) and (18), we have

$$\text{ADV}(F) \geq r(F),$$

as required.

Finally we examine the subcase where $r(G_3) \leq r(G_2)$. In this case, we let $\beta = (r(G_1), r(G_3), r(G_2))$. By the assumption of $\text{ADV}(G_2) \leq \text{ADV}(G_3)$ and the induction hypothesis (17), we have

$$r(G_3) \leq r(G_2) \leq \text{ADV}(G_2) \leq \text{ADV}(G_3).$$

Hence we have $\beta \leq \alpha$. So applying Proposition 1 and Theorem 5, we have

$$\begin{aligned} \text{ADV}_\alpha(\text{MUX}) &\geq \text{ADV}_\beta(\text{MUX}) \\ &\geq r(G_3) + \sqrt{(r(G_2) - r(G_3))^2 + r(G_1)^2} \\ &= r(F). \end{aligned}$$

The inequality above and (16) gives

$$\text{ADV}(F) \geq r(F),$$

as required. $\square$

Using the theorem above together with Theorem 1, we immediately obtain the following results.

Corollary 1: For any function f represented by a formula F in $\mathcal{F}$, and for any $\epsilon \in [0, 1/2)$,

$$Q_\epsilon(f) = \Omega(r(F)).$$

Moreover, we show as our main result that for functions represented by formulas in $\mathcal{F}$, there is at most a quadratic gap in the query complexity between deterministic and quantum algorithms. To show this, we need two technical lemmas.

Lemma 1: For any function f represented by a formula F in $\mathcal{F}$,

$$d(F) \geq D(f).$$

Proof. We devise a deterministic query algorithm in a straightforward manner based on F , which behaves as follows: If F is a single literal x_i or $\bar{x}_i$, then query with i to obtain the value of x_i , and output it if $F = x_i$ and its negation if $F = \bar{x}_i$. If $F = F_1 \wedge F_2$, or $F = F_1 \vee F_2$, or $F = \text{XOR}(F_1, F_2)$ for some formulas F_1 and F_2 in $\mathcal{F}$, then recursively evaluate F_1 and F_2 to obtain the values of F_1 and F_2 , respectively, and then compute the value of F and output it. If $F = \text{MUX}(F_1, F_2, F_3)$ for some formulas F_1, F_2 and F_3 in $\mathcal{F}$, then first evaluate F_1 and then evaluate F_2 or F_3 according to the value of F_1 . It is obvious that the number of queries made by the above algorithm is bounded by $d(F)$. $\square$

Lemma 2: For any formula F in $\mathcal{F}$,

$$\sqrt{d(F)} \leq r(F). \quad (19)$$

Proof. We prove (19) by induction on the depth of F .

(Basis) If the depth of F is 0, i.e., F is a literal, then clearly $r(F) = d(F) = 1$, and we thus have (19).

(Induction step) We consider three cases, where $F = G_1 \vee G_2$ or $F = G_1 \wedge G_2$, $F = \text{XOR}(G_1, G_2)$ and $F = \text{MUX}(G_1, G_2, G_3)$.

First we examine the case where $F = G_1 \wedge G_2$ or $F = G_1 \vee G_2$. By the definition of r and d , we have

$$r(F) = \sqrt{r(G_1)^2 + r(G_2)^2}$$

and

$$d(F) = d(G_1) + d(G_2).$$

From these formulas with the induction hypotheses $r(G_1) \geq \sqrt{d(G_1)}$ and $r(G_2) \geq \sqrt{d(G_2)}$, Eq. (19) follows.

For the case where $F = \text{XOR}(G_1, G_2)$, we have, by definition, that

$$r(F) = r(G_1) + r(G_2)$$

and

$$d(F) = d(G_1) + d(G_2).$$

By the induction hypothesis,

$$\begin{aligned} r(F) &= r(G_1) + r(G_2) \\ &\geq \sqrt{d(G_1)} + \sqrt{d(G_2)} \\ &\geq \sqrt{d(G_1) + d(G_2)} \\ &= \sqrt{d(F)}, \end{aligned}$$

which verified (19).

Finally, for the case where $F = \text{MUX}(G_1, G_2, G_3)$, we have

$$d(F) = d(G_1) + \max\{d(G_2), d(G_3)\}$$

and

$$r(F) = \begin{cases} r(G_2) + \sqrt{l^2 + r(G_1)^2} & \text{if } r(G_3) > r(G_2), \\ r(G_3) + \sqrt{l^2 + r(G_1)^2} & \text{if } r(G_3) \leq r(G_2), \end{cases}$$

where $l = |r(G_2) - r(G_3)|$. Note that if $r(G_3) > r(G_2)$ ($r(G_3) \leq r(G_2)$, resp.), then $r(F)$ is monotone increasing function with respect to $r(G_2)$ ($r(G_3)$, resp.), and thus $r(F)$ is minimized at $r(G_2) = 0$ ($r(G_3) = 0$, resp.). So, we have

$$r(F) \geq \sqrt{\max\{r(G_2), r(G_3)\}^2 + r(G_1)^2}.$$

This inequality together with induction hypothesis immediately verifies (19). $\square$

Combining all the inequalities in Corollary 1, Lemma 1 and Lemma 2, we get our main theorem.

Theorem 8: For any function f represented by a formula F in $\mathcal{F}$,

$$Q_\epsilon(f) = \Omega(\sqrt{D(f)}).$$

6. Adversary Bounds for Read-Once Decision Trees

In this section, we show that the class $\mathcal{F}$ properly contains the class of read-once decision trees and interesting implications of the two measures r and d when applied to read-once decision trees. More precisely, r and d can be interpreted as the complexity measures for decision trees called *soft rank* and *depth*, respectively.

A decision tree is a rooted ordered binary tree, where each internal node is labeled with a variable x_i and each leaf is labeled with a value 0 or 1. Given an input $x \in \{0, 1\}^n$, the tree is evaluated as follows. Start at the root, and repeat the following procedure: If the current node is a leaf, then stop and output the value (0 or 1) of the leaf; Otherwise, query the variable x_i of the current node and go to the left child if $x_i = 0$, and go to the right child if $x_i = 1$.

We say a decision tree computes f if its output equals $f(x)$, for all $x \in \{0, 1\}^n$. In what follows, we sometimes identify a decision tree T with the function that it computes, and denote by $T(x)$ the output of T for input x . A decision tree T is read-once if each variable appears at most once in T . We give in Fig. 2 an example of a read-once decision tree.

The depth of a decision tree T , denoted by $\text{depth}(T)$, is

Fig. 2 A read-once decision tree.

defined as the length of a longest path from the root to a leaf in the tree T . In other words, the depth of T is the number of deterministic queries made on the worst-case input to obtain the value of T . The soft rank is introduced in the previous version of the paper [10] and defined recursively as follows. If a decision tree T is a leaf, then the soft rank of T , denoted by $\tilde{r}(T)$ is 0. Otherwise, T consists of a root with two child subtrees T_1 and T_2 , and

$$\begin{aligned}\tilde{r}(T) &= \min\{\tilde{r}(T_1), \tilde{r}(T_2)\} + \sqrt{l^2 + 1}, \\ &= \max\{\tilde{r}(T_1), \tilde{r}(T_2)\} - l + \sqrt{l^2 + 1},\end{aligned}$$

where $l = |\tilde{r}(T_1) - \tilde{r}(T_2)|$. For example, the depth and soft rank of the decision tree T in Fig. 2 is $\text{depth}(T) = 3$ and $\tilde{r}(T) = 1 + \sqrt{4 - 2\sqrt{2}}$, respectively.

Intuitively, the soft rank measures how well-balanced a given tree is. To see this, observe that the function $\sigma(l) = -l + \sqrt{l^2 + 1}$ is monotone decreasing and the smaller l (meaning the better-balanced the decision tree T), the larger the soft rank $\tilde{r}(T)$. Indeed, among all decision trees of depth k , the complete binary decision tree (the most balanced tree) has maximum soft rank k , the decision list (the most unbalanced tree) has soft rank $\sqrt{k}$, and any other trees have soft rank between $\sqrt{k}$ and k . The soft rank is a somehow relaxed variant of the *rank* defined by Ehrenfeucht and Haussler [8], who investigate the learnability of low-rank decision trees. The rank is defined similarly to the soft rank, except that the function $\sigma(l)$ is replaced by $\delta(l)$, which is 1 if $l = 0$ and 0 otherwise. For more detail, see [10].

Now we observe that a decision tree T can be viewed as a Boolean formula over the basis {MUX} with additional constant inputs 0 and 1. This is because if the root of T is labeled with x_i with left subtree T_1 and right subtree T_2 , then T can be represented as $\text{MUX}(x_i, T_1, T_2)$. For example, the decision tree in Fig. 2 is represented as

$$\text{MUX}(x_1, \text{MUX}(x_2, 0, 1), \text{MUX}(x_3, \text{MUX}(x_4, 1, 0), 0)).$$

Furthermore, we can eliminate from the formula all occurrences of “0” and “1” by allowing AND, OR and NOT additional operators and using the substitution laws such as $\text{MUX}(x_i, 0, 1) = x_i$, $\text{MUX}(x_i, g, 0) = \bar{x}_i \wedge g$, and $\text{MUX}(x_i, 1, g) = \bar{x}_i \vee g$. For example, the decision tree in Fig. 2 can be represented as

$$\text{MUX}(x_1, x_2, \bar{x}_3 \wedge \bar{x}_4).$$

In this way, any read-once decision tree T has an equivalent Boolean formula F_T in the class $\mathcal{F}$.

Moreover, it is clear from definition that $d(F_T) = \text{depth}(T)$ and $r(F_T) = \tilde{r}(T)$. We thus have the following bounds on the quantum query complexity for read-once decision trees.

Corollary 2: For any read-once decision tree T ,

$$Q_\epsilon(T) = \Omega(\tilde{r}(T)), \quad Q_\epsilon(T) = \Omega(\sqrt{\text{depth}(T)}).$$

7. Concluding Remarks

We introduced a complexity measure r for the class $\mathcal{F}$ of read-once formulas over the basis {AND, OR, NOT, XOR, MUX} and showed that for any Boolean function represented by a formula F in the class $\mathcal{F}$, $r(F)$ is a lower bound on the bounded error quantum query complexity of the function. The measure r has an interesting implication if we restrict the class to decision trees. We also showed that for functions represented by formulas in $\mathcal{F}$, there is at most a quadratic gap in the query complexity between deterministic and quantum algorithms. So we made some progress toward the conjecture that an at most quadratic gap exists for any Boolean function.

The technical contribution of this paper is computing tight bounds on ADV_α for the two-bit XOR function and the three-bit multiplexer function (MUX). To derive the bounds on $\text{ADV}_\alpha(\text{XOR})$ and $\text{ADV}_\alpha(\text{MUX})$, we explicitly constructed adversary matrices, which were obtained in the following way. Using a semi-definite programming package, we numerically calculated the values of $\text{ADV}_\alpha(\text{XOR})$ and $\text{ADV}_\alpha(\text{MUX})$ together with optimal adversary matrices with about twelve-digit accuracy, for various settings of cost vectors α . For each of XOR and MUX, we observed the matrices obtained and tried to guess a general form of the optimal matrix in terms of α , i.e., we tried to express each element of the optimal adversary matrix as a function of α . Note that any (not necessarily optimal) adversary matrix gives a lower bound on ADV_α . Fortunately, we easily obtained the exact expression of the optimal matrix for XOR, from which we derived $\text{ADV}_\alpha(\text{XOR}) = a + b$ for $\alpha = (a, b)$. However, the adversary matrix Γ for MUX we gave in Fig. 1 is not optimal. Based on the observation of the numerical calculations, we are pretty sure that the optimal matrix for $\text{ADV}_\alpha(\text{MUX})$ with $\alpha = (1, b, c)$ should be of the following form:

$$\begin{bmatrix} b - x^2/b & x & \sqrt{1 - x^2} \\ x & 0 & \sqrt{c(c - b)} \\ \sqrt{1 - x^2} & \sqrt{c(c - b)} & b \end{bmatrix}$$

for some real number $x \in [0, 1]$, where the rows are indexed by 001, 100, 110, and the columns by 010, 111, 101. In other words, we should replace the matrix A of (7) with the one above. The value $\text{ADV}_\alpha(\text{MUX})$ is then given by maximizing the spectral norm of Γ over $x \in [0, 1]$. However,

we have not succeeded to obtain the optimal value of x explicitly in terms of b and c . In this paper, we set x to be a constant, i.e., $x = 0$, because it makes the subsequent arguments easy and yet gives a good approximation.

It should be noted that we could use a new adversary method by [12], who introduce an adversary bound $\text{ADV}^\pm$, which always gives larger lower bounds than ADV . But it seems much more complicated calculations involved.

Acknowledgments

This work was supported by MEXT Grant-in-Aid for Scientific Research (C) No.20500001.

References

- [1] S. Aaronson, "The limits of quantum computers," *Scientific American*, vol.298, no.3, pp.62–69, 2008.
- [2] A. Ambainis, "Quantum lower bounds by quantum arguments," *J. Comput. Syst. Sci.*, vol.64, no.4, pp.750–767, 2002.
- [3] A. Ambainis, "Polynomial degree vs. quantum query complexity," *J. Comput. Syst. Sci.*, vol.72, no.2, pp.220–238, 2006.
- [4] H. Barnum and R. de Wolf, "Complexity measures and decision tree complexity: A survey," *Theor. Comput. Sci.*, no.288, pp.21–43, 2002.
- [5] H. Barnum and M. Saks, "A lower bound on the quantum query complexity of read-once functions," *J. Comput. Syst. Sci.*, vol.69, no.2, pp.244–258, 2004.
- [6] H. Barnum, M. Saks, and M. Szegedy, "Quantum decision trees and semidefinite programming," *Proc. 18th IEEE Conference on Computational Complexity*, pp.179–193, 2003.
- [7] R. Beals, E. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," *J. ACM*, vol.48, no.4, pp.778–797, 2001.
- [8] A. Ehrenfeucht and D. Haussler, "Learning decision trees from random examples," *Inf. Comput.*, vol.82, no.3, pp.231–246, 1989.
- [9] L.K. Grover, "A fast quantum mechanical algorithm for database search," *Proc. 28th ACM Symposium on Theory of Computing (STOC '96)*, pp.212–219, 1996.
- [10] H. Fukuhara and E. Takimoto, "Lower bounds on quantum query complexity for read-once decision trees with parity nodes," *Proc. 15th Computing: The Australasian Theory Symposium (CATS 2009)*, vol.94, pp.91–100, 2009.
- [11] P. Høyer, T. Lee, and R. Špalek, "Tight adversary bounds for composite functions," *arXiv:quant-ph/0509067v3*, 2006.
- [12] P. Høyer, T. Lee, and R. Špalek, "Negative weights make adversaries stronger," *Proc. 39th ACM Symposium on Theory of Computing (STOC '07)*, pp.526–535, 2007.
- [13] S. Laplante and F. Magniez, "Lower bounds for randomized and quantum query complexity using Kolmogorov arguments," *Proc. 19th IEEE Conference on Computational Complexity*, pp.294–304, 2004.
- [14] G. Midrijanis, "Exact quantum query complexity for total boolean functions," *arXiv:quant-ph/0403168v2*, 2004.
- [15] D. Simon, "On the power of quantum computation," *SIAM J. Comput.*, vol.26, pp.1474–1483, 1997.
- [16] R. Špalek and M. Szegedy, "All quantum adversary methods are equivalent," *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, pp.1299–1322, 2005.

Hideaki Fukuhara received his M.E. degree in information sciences from Tohoku University in 2006. He is a doctoral course student at Graduate School of Information Sciences, Tohoku University, Sendai, Japan. His research interests include circuit complexity.

Eiji Takimoto received his M.E. and D.E. degrees in information engineering from Tohoku University in 1987 and 1991, respectively. Presently, he is a professor of Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan. His research interests include complexity theory, computational learning theory, and on-line decision making.