Lower bounds on quantum query complexity for read-once decision trees with parity nodes

Hideaki Fukuhara[1] **Eiji Takimoto**[2]

[1] Graduate School of Information Science, Tohoku University,
Aramaki-Aza-Aoba 6-6-05, Aoba-ku, Sendai 980-8579, Japan
Email: fk_hide@i.kyushu-u.ac.jp

[2] Department of Informatics,
Graduate School of Information Science and Electrical Engineering,
Kyushu University,
744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan
Email: eiji@i.kyushu-u.ac.jp

Abstract

We introduce a complexity measure for decision trees called the soft rank, which measures how well-balanced a given tree is. The soft rank is a somehow relaxed variant of the rank. Among all decision trees of depth d, the complete binary decision tree (the most balanced tree) has maximum soft rank d, the decision list (the most unbalanced tree) has minimum soft rank $\sqrt{d}$, and any other trees have soft rank between $\sqrt{d}$ and d. We show that, for any decision tree T in some class G of decision trees which includes all read-once decision trees, the soft rank of T is a lower bound on the quantum query complexity of the Boolean function that T represents. This implies that for any Boolean function f that is represented by a decision tree in G, the deterministic query complexity of f is only quadratically larger than the quantum query complexity of f.

Keywords: quantum query complexity, 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., Aaronson (2008)). Most of the interesting results have been shown in the *quantum query model* of computation. In the model, the algorithm is given access to a black-box containing an input assignment $x = (x_1, x_2, \ldots, x_n)$ 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 the value assigned to a single variable of some index i, $1 \leq i \leq n$, and the value x_i is returned. Queries are *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. The complexity of f is the number of queries needed to compute f on a worst-case input x. The quantum query model is the quantum analogue to the classical decision tree model, where the query complexity is measured by the depth of decision trees. It is of great interest to compare the computational powers of the two models.

One of the major algorithmic results in this regard is Grover's search algorithm (Grover 1996), which can be viewed as a quantum algorithm for computing the n-bit OR function with $O(\sqrt{n})$ queries. This contrasts with the facts that n queries are required for deterministic decision trees and $\Omega(n)$ queries for randomized decision trees. Thus the OR function provides an example where quantum computation gives a quadratic speedup over deterministic and randomized decision trees. It should be noticed that the bound of $O(\sqrt{n})$ on the quantum query complexity of the OR function holds only in the *bounded error setting*, where the computation may have a small probability of being incorrect. We write $Q_\epsilon(f)$ for the quantum query complexity of f in the bounded error setting, where ϵ is the permissible error. We also write $D(f)$ for the deterministic decision tree complexity of f. 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. With these notations, we can describe the query complexities of the n-bit OR function, denoted by OR_n, as follows: $Q_\epsilon(\mathrm{OR}_n) = O(\sqrt{n})$ for any $\epsilon \in (0, 1/2)$, and $D(\mathrm{OR}_n) = n$. (It is well known that the value of ϵ only affects $Q_\epsilon(f)$ within a constant factor, unless $\epsilon = 0$.)

On the other hand, Barnum & Saks (2004) 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 $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$ 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 decision trees 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

This work was done while the first author was visiting Kyushu University. The second author is supported by MEXT Grant-in-Aid for Scientific Research (C) No.20500001.

whose domain is all of $\{0,1\}^n$. For partial functions whose domain is restricted, there are much better speedups known (Simon 1997). The best known result toward this conjecture say that for any Boolean function f, $Q_\epsilon(f) = \Omega(D(f)^{1/6})$ for $\epsilon \in (0, 1/2)$ (Beals et al. 2001), and $Q_\epsilon(f) = \Omega(D(f)^{1/3})$ for $\epsilon = 0$ (Midrijanis 2004). Several results provide evidence for the conjecture. For example, the bound of $Q_\epsilon(f) = \Omega(D(f)^{1/2})$ holds for all symmetric functions, and for some particular functions f such as the PARITY and the MAJORITY functions, $Q_\epsilon(f) = \Omega(D(f))$ (Beals et al. 2001).

In this paper, we provide further evidence for the conjecture. We show that the bound of $Q_\epsilon(f) = \Omega(D(f)^{1/2})$ holds for any function f that can be represented as a decision tree in some class of deterministic decision trees, denoted by $\mathrm{DT}_\oplus$. The class $\mathrm{DT}_\oplus$ consists of *read-once decision trees* that may have special nodes called the *parity nodes*. A parity node is an inner node that has only one child subtree T, and represents the parity of the variable labeled at the node and the function that T represents. A read-once decision tree is a decision tree in which each variable appears at most once. The AND, OR, and PARITY functions can be represented by decision trees in $\mathrm{DT}_\oplus$.

Below we state our results in a more precise way. We introduce a complexity measure for decision trees called the *soft rank*, which measures how well-balanced a given tree is, and show that for any function f that can be represented as a decision tree T in $\mathrm{DT}_\oplus$, the bounded error quantum complexity $Q_\epsilon(f)$ is bounded below by the soft rank of T. The soft rank is a somehow relaxed variant of the rank defined by Ehrenfeucht & Haussler (1989), who investigate the learnability of low-rank decision trees. Among all decision trees of depth d, the complete binary decision tree (the most balanced tree) has maximum soft rank d, the decision list (the most unbalanced tree) has minimum soft rank $\sqrt{d}$, and any other trees have soft rank between $\sqrt{d}$ and d. Since the depth of a tree T in $\mathrm{DT}_\oplus$ is an upper bound on the decision tree complexity $D(f)$ of the function f that T represents, the bound $Q_\epsilon(f) = \Omega(D(f)^{1/2})$ immediately follows from our lower bound.

To derive the lower bound, we employ the adversary method, which is one of the most successful techniques for showing lower bounds on quantum query complexity (Ambainis 2002, 2006, Barnum et al. 2003, Laplante & Magniez 2004, Špalek & Szegedy 2005). 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 $\mathrm{ADV}(f)$, gives a lower bound on $Q_\epsilon(f)$. Thus, $\mathrm{ADV}(f)$ is often referred to as the *adversary bound*. Høyer et al. (2006) consider the adversary bound for composite functions of the form $h = f \circ (g_1, g_2, \ldots, g_k)$ and give an exact expression for $\mathrm{ADV}(h)$ in terms of the adversary bounds of f and g_i for $1 \le i \le k$. More precisely, they generalize the adversary method and introduce a quantity $\mathrm{ADV}_\alpha(f)$ with a cost vector α, and show that $\mathrm{ADV}(h) = \mathrm{ADV}_\alpha(f)$ with $\alpha = (\mathrm{ADV}(g_1), \mathrm{ADV}(g_2), \ldots, \mathrm{ADV}(g_k))$, provided that the sets X_i of input variables for g_i are disjoint from each other. Despite the constraint of disjointness, this composition theorem enables us to estimate asymptotic bounds on ADV for various functions such as read-once functions (Høyer et al. 2006).

We apply the composition theorem to obtain a lower bound on ADV for decision trees in the class $\mathrm{DT}_\oplus$, based on the observation that $\mathrm{DT}_\oplus$ consists of recursively defined composite functions with two basis functions XOR and MUX, where XOR denotes the two-variable exclusive-or function and MUX denotes the three-variable multiplexer function. The lower bound obtained is in the form of a complicated recurrence equation, which seems hard to solve. So, we approximate the equation and get a simple recurrence equation, by which the soft rank is defined.

The rest of the paper is organized as follows.

In Section 2 we give basic terminologies that are used throughout the paper. In Section 3 we define the class of read-once decision trees with parity nodes and introduce the notion of soft rank. In Section 4 we state the quantum query model and review the adversary method with some useful results including the composition theorem. In Section 5 we give tight bounds on $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$. These are the main technical contribution of the paper. Applying the composition theorem to the bounds on $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$, we give in Section 6 the soft rank adversary bound for read-once decision trees with parity nodes. We also state the gap between classical and quantum query complexities. In Section 7 we demonstrate that for several functions one can easily compute the soft rank and thus the adversary bounds as well. In Section 8 we state some concluding remarks.

2 Preliminaries

For a natural number n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For a binary sequence $x \in \{0,1\}^n$, 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. For a matrix A and a column vector v, A^T and v^T denote the transposes of A and v, respectively. For a matrix A, 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, $\bar{A}[x, y] = 1 - A[x, y]$.

3 Decision trees and soft rank

In this paper, we consider the class $\mathrm{DT}_\oplus$ of read-once decision trees with parity nodes. We first begin with the definition of the *standard decision trees*. A standard decision tree is a rooted ordered binary tree, where each internal node of a decision tree 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

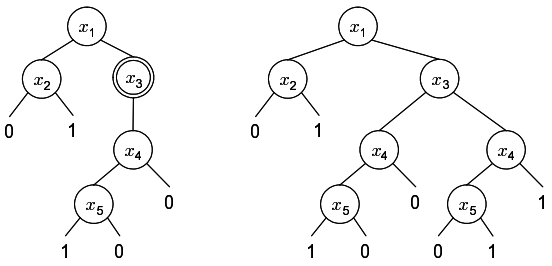

Figure 1: A decision tree in $\mathrm{DT}_\oplus$ (left) and a standard decision tree (right) equivalent to the left tree. The node designated by a double circle with x_3 in it stands for a parity node.

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.

The complexity of a tree is its depth, i.e., the number of queries made on the worst-case input. We define the decision tree complexity of f, denoted by $D(f)$, as the depth of a minimal-depth decision tree that computes f.

For a decision tree T, $\bar{T}$ denotes the decision tree that is the same as T, except that all leaf labels are inverted. Clearly, $\bar{T}$ computes the negation function of T, i.e., $\bar{T}(x) = 1 - T(x)$ for all inputs x.

3.1 Read-once decision trees with parity nodes

Now we define *decision trees with parity nodes*. A parity node is an inner node that is labeled with some input variable as a normal inner node, but it has only one child. A tree with parity nodes is evaluated as the same way as the standard trees, except the case where the current node is a parity node. When the current node is a parity node labeled with a variable x_i, first go to the (unique) child and evaluate recursively the subtree T rooted at the child, and then output $x_i \oplus T(x)$. In other words, if $x_i = 0$, then go to the child and evaluate normally the subtree T, but if $x_i = 1$, then evaluate the negation $\bar{T}$ of the subtree T.

It should be noticed that a parity node with child subtree T can be replaced by a normal inner node with left child subtree T and right child subtree $\bar{T}$, without changing the function that is computed.

We say that a tree is a read-once decision tree if every variable x_i appears at most once as node labels of the tree. Let $\mathrm{DT}_\oplus$ denote the class of read-once decision trees with parity nodes. Note that any tree in $\mathrm{DT}_\oplus$ can be transformed into a standard decision tree (with no parity nodes) without changing the depth by the node replacement as stated above, but the resultant tree may not be a read-once decision tree (See Figure 1). Therefore, the depth of a decision tree T in $\mathrm{DT}_\oplus$ is an upper bound on $D(f)$ for f that T computes.

3.2 Representation with composite functions

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,

$$\begin{aligned} \mathrm{XOR}(x_1, x_2) &= x_1 \oplus x_2, \\ \mathrm{MUX}(x_1, x_2, x_3) &= \bar{x}_1 x_2 \vee x_1 x_3. \end{aligned}$$

We observe that a decision tree T in $\mathrm{DT}_\oplus$ can be viewed as a recursively defined composite function with two basis functions XOR and MUX. If the root of T is a normal inner node labeled with x_i with left subtree T_1 and right subtree T_2, then T can be represented as $\mathrm{MUX}(x_i, T_1, T_2)$. Similarly, if the root of T is a parity node labeled with x_i with unique subtree T_1, then T can be represented as $\mathrm{XOR}(x_i, T_1)$. In this way, we can recursively represent the subtrees T_1 and T_2 as composite functions with XOR and MUX. For example, the left tree in Figure 1 is represented as

$$\mathrm{MUX}(x_1, x_2, \mathrm{XOR}(x_3, \mathrm{MUX}(x_4, \bar{x}_5, 0))).$$

3.3 Rank and soft rank

Below we define the complexity measure *soft rank* for decision trees that may have parity nodes. The soft rank measures how well-balanced a given tree is. For comparison, we give the definition of a similar measure called the *rank* (Ehrenfeucht & Haussler 1989).

Definition 1 (rank) *For a decision tree T that may have parity nodes, the rank of T, denoted by $r(T)$, is defined recursively as follows. If T is a leaf, then $r(T) = 0$. If the root of T is a parity node with subtree T_1, then $r(T) = r(T_1)+1$. If the root of T is a normal inner node with left subtree T_1 and right subtree T_2, then*

$$r(T) = \begin{cases} \max\{r(T_1), r(T_2)\} & \text{if } r(T_1) \neq r(T_2), \\ r(T_1) + 1 & \text{if } r(T_1) = r(T_2). \end{cases}$$

So, when we combine two trees T_1 and T_2 to make a larger tree $T = \mathrm{MUX}(x_i, T_1, T_2)$, the rank does not increase unless $r(T_1) = r(T_2)$. While on the other hand, the soft rank always increases by an amount determined by the difference $|\tilde{r}(T_1) - \tilde{r}(T_2)|$.

Definition 2 (soft rank) *For a decision tree T that may have parity nodes, the soft rank of T, denoted by $\tilde{r}(T)$, is defined recursively as follows. If T is a leaf, then $\tilde{r}(T) = 0$. If the root of T is a parity node with subtree T_1, then $\tilde{r}(T) = \tilde{r}(T_1) + 1$. If the root of T is a normal inner node with left subtree T_1 and right subtree T_2, then*

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

where $l = |\tilde{r}(T_1) - \tilde{r}(T_2)|$.

It is clear from definition that the soft rank is at least as large as the rank for any decision trees.

Proposition 1 *For any decision tree T with parity nodes,*

$$\tilde{r}(T) \geq r(T).$$

In fact, for the left tree T of Figure 1, we can easily verify that $r(T) = 2$ and $\tilde{r}(T) = 1 + \sqrt{3}$. Note that the same results hold for the right tree of Figure 1. Generally, the rank (and the soft rank, resp.) of a decision tree T with parity nodes are the same as the rank (and the soft rank, resp.) of the standard decision tree with no parity nodes that is transformed from T.

We show in the next proposition that, among all decision trees of depth d, the complete binary decision

tree (the most balanced tree) has maximum soft rank d, the decision list (the most unbalanced tree) has minimum soft rank $\sqrt{d}$, and any other trees have soft rank between $\sqrt{d}$ and d. Here, by a decision list we mean a standard decision tree such that one of the two children of each internal node is a leaf.

Proposition 2 *For any decision tree T of depth d,*

$$\sqrt{d} \leq \tilde{r}(T) \leq d.$$

Furthermore, the first equality is attained by a decision list, and the second equality is attained by the complete binary decision tree.

Proof. Without loss of generality, we consider only standard decision trees. Let $\tilde{r}_{\max}(d)$ (and $\tilde{r}_{\min}(d)$, resp.) denote the maximum (and the minimum, resp.) of $\tilde{r}(T)$ over all decision trees T of depth d. It is clear that $\tilde{r}_{\max}(d)$ and $\tilde{r}_{\min}(d)$ are monotonically increasing functions of d and take value 0 when $d = 0$. Let T be an arbitrary decision tree of depth d whose root has left subtree T_1 and right subtree T_2. We assume that the depth of T_1 is $d-1$ and the depth of T_2 is $d' \leq d-1$.

We first show that $\tilde{r}_{\max}(d) = d$. By the definition of the soft rank and the monotonicity of $\tilde{r}_{\max}$, we have

$$\begin{aligned}\tilde{r}(T) &= \max\{\tilde{r}(T_1), \tilde{r}(T_2)\} - l + \sqrt{l^2+1} \\ &\leq \max\{\tilde{r}_{\max}(d-1), \tilde{r}_{\max}(d')\} - l + \sqrt{l^2+1} \\ &= \tilde{r}_{\max}(d-1) - l + \sqrt{l^2+1},\end{aligned}$$

where $l = |\tilde{r}(T_1) - \tilde{r}(T_2)|$. Since $-l + \sqrt{l^2+1} \leq 1$ for any $l \geq 0$, we have

$$\tilde{r}(T) \leq \tilde{r}_{\max}(d-1) + 1.$$

We thus establish a recurrence formula

$$\tilde{r}_{\max}(d) \leq \tilde{r}_{\max}(d-1) + 1,$$

which gives

$$\tilde{r}_{\max}(d) \leq d.$$

On the other hand, the complete binary decision tree of depth d has soft rank d. So, we have

$$\tilde{r}_{\max}(d) = d.$$

Next we show that $\tilde{r}_{\min}(d) = \sqrt{d}$. By the definition of the soft rank, we have

$$\tilde{r}(T) = \begin{cases} \tilde{r}(T_1) + \sqrt{l^2+1} & \text{if } \tilde{r}(T_2) > \tilde{r}(T_1), \\ \tilde{r}(T_1) - l + \sqrt{l^2+1} & \text{if } \tilde{r}(T_2) \leq \tilde{r}(T_1), \end{cases}$$

from which it follows that

$$\tilde{r}(T) \geq \tilde{r}(T_1) - l + \sqrt{l^2+1}$$

for any l, $0 \leq l \leq \tilde{r}(T_1)$. Since $-l + \sqrt{l^2+1}$ is minimized at $l = \tilde{r}(T_1)$, we have

$$\tilde{r}(T) \geq \sqrt{\tilde{r}(T_1)^2 + 1} \geq \sqrt{\tilde{r}_{\min}(d-1)^2 + 1}.$$

We thus establish a recurrence formula

$$\tilde{r}_{\min}(d) \geq \sqrt{\tilde{r}_{\min}(d-1)^2 + 1},$$

which gives

$$\tilde{r}_{\min}(d) \geq \sqrt{d}.$$

On the other hand, a decision list of depth d has soft rank $\sqrt{d}$. So, we have

$$\tilde{r}_{\min}(d) = \sqrt{d}.$$

□

4 Quantum query model

As with the classical counter part, in the quantum query model we wish to compute some Boolean function $f : \{0,1\}^n \to \{0,1\}$, and we access the input through queries. The complexity of f is the number of queries needed to compute f. Unlike the classical case, however, we can now 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 $i \in [n]$, $b \in \{0,1\}$, and z represents the workspace. A t-query quantum algorithm A has the form $A = U_t O U_{t-1} O \cdots O U_1 O U_0$, 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. We denote by $Q_\epsilon(f)$ the minimum query complexity of a quantum query algorithm that ϵ-approximates f.

Along with the polynomial method (Beals et al. 2001), one of the main techniques for showing lower bounds in quantum query complexity is the quantum adversary method.

4.1 Adversary bound

There are several formulation 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.

We say that a matrix Γ is an adversary matrix for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ 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 $f(x) = f(y)$.

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 $\mathrm{ADV}(f)$, gives a lower bound on $Q_\epsilon(f)$ (Barnum et al. 2003). Thus, $\mathrm{ADV}(f)$ is referred to as the *adversary bound* of f.

Definition 3 *Let $f : \{0,1\}^n \to \{0,1\}$ and Φ_f be the set of adversary matrices for f.* $\mathrm{ADV}(f)$ *is defined as*

$$\mathrm{ADV}(f) = \max_{\Gamma \in \Phi_f} \min_{i \in [n]} \frac{\|\Gamma\|}{\|\Gamma \circ D_i\|}.$$

Theorem 1 (Barnum et al. (2003)) *For any Boolean function f and any $\epsilon \in [0, 1/2)$,*

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

4.2 Adversary bounds for composite functions

Høyer et al. (2006) generalize the adversary method and introduce a quantity ADV_α with a cost vector α.

Definition 4 *Let $f : \{0,1\}^n \to \{0,1\}$ and Φ_f be the set of adversary matrices for f. For every vector $\alpha \in \mathbb{R}_+^n$, $\mathrm{ADV}_\alpha(f)$ is defined as*

$$\mathrm{ADV}_\alpha(f) = \max_{\Gamma \in \Phi_f} \min_{i \in [n]} \frac{\alpha_i \|\Gamma\|}{\|\Gamma \circ D_i\|}.$$

Note that when $\alpha = (1, 1, \ldots, 1)$, $\mathrm{ADV}_\alpha(f) = \mathrm{ADV}(f)$. It is clear from Definition 4 that ADV_α is monotonically increasing with respect to α. For two vectors $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, we write $\alpha \le \beta$ if $\alpha_i \le \beta_i$ for any index $i \in [n]$.

Proposition 3 *Let $f : \{0,1\}^n \to \{0,1\}$. For any vectors α and β in $\mathbb{R}_+^n$ such that $\alpha \le \beta$,*

$$\mathrm{ADV}_\alpha(f) \le \mathrm{ADV}_\beta(f).$$

Moreover, ADV_α has the following obvious but useful properties.

Proposition 4 *Let $f : \{0,1\}^n \to \{0,1\}$ and $\alpha \in \mathbb{R}_+^n$. Then, for any $c \in \mathbb{R}_+$,*

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

Proposition 5 *Let $f : \{0,1\}^n \to \{0,1\}$ and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{R}_+^n$. For a permutation π over the set $\{1, 2, \ldots, n\}$, $\pi(f)$ denotes the function defined as $f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$, and $\pi(\alpha)$ denotes the vector defined as $(\alpha_{\pi(1)}, \alpha_{\pi(2)}, \ldots, \alpha_{\pi(n)})$. For any permutation π over $\{1, 2, \ldots, n\}$,*

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

Proposition 6 *Let $f : \{0,1\}^n \to \{0,1\}$ and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{R}_+^n$. For a bit $v \in \{0,1\}$, let $v^0 = v$ and $v^1 = \bar{v}$. For any function f' defined as $f'(x_1, x_2, \ldots, x_n) = f(x_1^{a_1}, x_2^{a_2}, \ldots, x_n^{a_n})$ for some binary sequence $a = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$,*

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

Høyer et al. (2006) consider the adversary bound for composite functions of the form $h = f \circ (g_1, g_2, \ldots, g_k)$ and give an exact expression for $\mathrm{ADV}(h)$ in terms of $\mathrm{ADV}_\alpha(f)$ and $\mathrm{ADV}(g_i)$ for $1 \le i \le k$.

Theorem 2 (Høyer et al. (2006)) *Let h be a Boolean function of n variables given by*

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

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

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

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

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. (2006) give a simple proof of the $\sqrt{n}$ adversary bound for read-once functions. Specifically, they first show that

$$\mathrm{ADV}_\alpha(\mathrm{AND}) = \mathrm{ADV}_\alpha(\mathrm{OR}) = |\alpha|$$

and then apply Theorem 2, by which the bound is immediately derived.

Unlike the cases of the AND and OR functions, it seems very hard to represent $\mathrm{ADV}_\alpha(f)$ 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 $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$.

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 5 *Let $p : \{0,1\}^n \times [n] \to \mathbb{R}$ be a set of probability distributions in the sense that $p_x(i) \ge 0$ and $\sum_{i \in [n]} p_x(i) = 1$ for every $x \in \{0,1\}^n$. Let P_n denote the set of all such sets p. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we define*

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

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

Theorem 3 (Høyer et al. (2006)) *For every $f : \{0,1\}^n \to \{0,1\}$ and $\alpha \in \mathbb{R}_+^n$,*

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

5 $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$

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

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

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

Proof. We first show that $\mathrm{ADV}_\alpha(\mathrm{XOR}) \le a + b$. 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 $\mathrm{MM}_\alpha(\mathrm{XOR})$. 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,

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

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

Next we show that $\mathrm{ADV}_\alpha(\mathrm{XOR}) \ge a+b$. 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 $\mathrm{ADV}_\alpha(\mathrm{XOR})$. 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

$$\begin{aligned} |\Gamma \circ D_1 - xI| &= (x-a)^2(x+a)^2, \\ |\Gamma \circ D_2 - xI| &= (x-b)^2(x+b)^2, \end{aligned}$$

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 $\mathrm{ADV}_\alpha(\mathrm{XOR}) \ge a+b$. □

For the multiplexer function MUX, we have not succeeded to derive the exact expression of $\mathrm{ADV}_\alpha(\mathrm{MUX})$. 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 $\mathrm{MUX}(x_1,x_2,x_3) = \mathrm{MUX}(\bar{x}_1,x_3,x_2)$, Propositions 5 and 6 imply that

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

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 $b \le c$.

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

$$\mathrm{ADV}_\alpha(\mathrm{MUX}) \ge \frac{b}{2} + \sqrt{(c-\frac{b}{2})^2 + \frac{a^2(c+a)}{c-b+a}}.$$

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

$$\mathrm{ADV}_\alpha(\mathrm{MUX}) \ge \frac{b}{2} + \sqrt{(c-\frac{b}{2})^2 + \frac{c+1}{c-b+1}} \quad (1)$$

for $\alpha = (1,b,c)$ with $b \le c$. This is because, for any $\alpha = (a,b,c) \in \mathbb{R}_+^3$ such that $b \le c$, Proposition 4 says that $\mathrm{ADV}_\alpha(\mathrm{MUX}) = a\mathrm{ADV}_\beta(\mathrm{MUX})$ for $\beta = (1,b/a,c/a)$, from which together with (1) the theorem follows.

Now we show (1) by constructing an adversary matrix Γ for MUX so that it satisfies the following four conditions.

$$\begin{aligned} \|\Gamma\| &\ge \frac{b}{2} + \sqrt{(c-\frac{b}{2})^2 + \frac{c+1}{c-b+1}}, && (2) \\ \|\Gamma \circ D_1\| &= 1, && (3) \\ \|\Gamma \circ D_2\| &= b, && (4) \\ \|\Gamma \circ D_3\| &= c. && (5) \end{aligned}$$

By the definition of ADV_α, (1) follows from (2), (3), (4) and (5). We give in Figure 2 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 (2), (3), (4) and (5) holds.

First we verify (2). Define a matrix A as

$$A = \begin{bmatrix} b & 0 & 1 \\ 0 & 0 & \sqrt{c(c-b)} \\ 1 & \sqrt{c(c-b)} & b \end{bmatrix} \quad (6)$$

and let λ_1, λ_2 and λ_3 be the three eigenvalues of A. We show that the following six values λ_1, $-\lambda_1$, λ_2, $-\lambda_2$, λ_3, $-\lambda_3$ are eigenvalues of Γ. Note that the rest two eigenvalues of Γ are both 0 with corresponding eigenvectors $(1,0,0,0,0,0,0,0)^T$ and $(0,0,0,1,0,0,0,0)^T$. 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 $v' = (0,v_1,v_1,v_2,0,v_3,v_3,v_2)^T$ and $v'' = (0,v_1,-v_1,v_2,0,-v_3,v_3,-v_2)^T$ 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

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

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. To see $p_A(x_0) > 0$,

$$\begin{aligned} & p_A(x_0) \\ &= \frac{b\sqrt{2c-b+2}}{2(c-b+1)^{3/2}} \left(\sqrt{(c-b+1)(2c-b+2)} \right. \\ &\quad \left. - \sqrt{2c^2-3bc+b^2+b+2} \right), \quad (7) \end{aligned}$$

and

$$(c-b+1)(2c-b+2) - (2c^2-3bc+b^2+b+2) = 4(c-b) \ge 0$$

since $c \ge b$. So both the numerator and denominator of (7) are positive.

Next we verify (3). 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 (4). 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, $\|\Gamma \circ D_2\| = \|C\| = b$.

	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	$\sqrt{c(c-b)}$	0	0
101	0	1	0	0	$\sqrt{c(c-b)}$	0	b	0
110	0	0	1	0	0	b	0	$\sqrt{c(c-b)}$
111	0	0	0	0	0	0	$\sqrt{c(c-b)}$	0

Figure 2: The adversary matrix Γ for MUX

Finally we verify (5). 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, $\|\Gamma \circ D_3\| = \|D\| = c$. □

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

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

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

Actually, by an easy calculation, we can see that our lower bound and upper bound on $\mathrm{ADV}_\alpha(\mathrm{MUX})$ with $\alpha = (a,b,c)$ match up to an additive factor of $O(la^2/(l^2+a^2))$, where $l = |b-c|$. Note that if $l = 0$, then the two bounds coincide. In other words, we have an exact expression of $\mathrm{ADV}_{(a,b,c)}(\mathrm{MUX})$ when $b = c$. Specifically, $\mathrm{ADV}_{(a,b,b)} = a + b$. Moreover, if l or a is a constant, then the two bounds coincide up to an additive constant. So, the bounds we will use in the next section for deriving a lower bound on ADV of decision trees are tight, because we apply our lower bound only in the case where $a = 1$.

6 Soft rank lower bound for read-once decision trees with parity nodes

We could apply Theorem 2 together with Theorems 4 and 5 to obtain a lower bound on ADV for decision trees in the class $\mathrm{DT}_\oplus$, based on the observation that $\mathrm{DT}_\oplus$ consists of recursively defined composite functions with two basis functions XOR and MUX. However, the lower bound obtained is in the form of a complicated recurrence equation, due to Theorem 5, where the parameters b and c in the theorem are recursively substituted by the lower bounds on the ADVs of the left and right child subtrees. So, we approximate the lower bound of Theorem 5 and get a simple recurrence equation.

Lemma 1 *For any $\alpha = (a,b,c) \in \mathbb{R}^3_+$ such that $b \le c$,*

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

Proof. Let $l = c - b$. By Theorem 5,

$$\begin{aligned} &\mathrm{ADV}_\alpha(\mathrm{MUX}) \\ &\ge \frac{b}{2} + \sqrt{\left(c - \frac{b}{2}\right)^2 + \frac{a^2(c+a)}{c-b+a}} \\ &= \frac{b}{2} + \sqrt{\frac{b^2}{4} + \left(l + \frac{a^2}{l+a}\right) b + l^2 + a^2} \\ &\ge \frac{b}{2} + \sqrt{\frac{b^2}{4} + b\sqrt{l^2+a^2} + l^2 + a^2} \qquad (8) \\ &= \frac{b}{2} + \sqrt{\left(\frac{b}{2} + \sqrt{l^2+a^2}\right)^2} \\ &= b + \sqrt{l^2+a^2}. \end{aligned}$$

Inequality (8) holds since

$$(l + \frac{a^2}{l+a})^2 = l^2 + a^2 + (\frac{al}{l+a})^2 \ge l^2 + a^2.$$

□

Now we give our main theorem of the paper. A decision tree T in $\mathrm{DT}_\oplus$ is said to be *non-redundant* if the following conditions hold.

1. There is no parity node whose child is a leaf.
2. There is no normal inner node whose two children are leaves labeled with the same value.

Theorem 7 *For any non-redundant decision tree T in $DT_\oplus$,*

$$\mathrm{ADV}(T) \ge \tilde{r}(T) \ge r(T).$$

Proof. By Proposition 1, we have $\tilde{r}(T) \ge r(T)$. We prove $\mathrm{ADV}(T) \ge \tilde{r}(T)$ by induction on the depth of T.

(Basis 1) If the depth of T is 0, i.e., T is a leaf, then clearly $\mathrm{ADV}(T) = \tilde{r}(T) = 0$.

(Basis 2) If the depth of T is 1, then since T is non-redundant, T should consist of the root and two leaves, where the root is a normal node labeled with some variable x_i and the two leaves have different values. Clearly, $\tilde{r}(T) = 1$. Furthermore, T computes either of the projection functions $T(x) = x_i$ or $T(x) = \bar{x}_i$. It is well known that the adversary bound for the projection functions is 1, and we thus have $\mathrm{ADV}(T) = 1$.

(Induction step) If the depth d of T is greater than or equal to 2, then we should consider two cases, where the root of T is a parity node (Case 1) and the root of T is a normal node (Case 2).

First we examine Case 1. In this case, the root of T is a parity node labeled with some variable x_i and it has one child subtree T_1 of depth $d-1$. Obviously, T

computes the function $\mathrm{XOR}(x_i, T_1)$. Applying Theorems 2, we have

$$\mathrm{ADV}(T) = \mathrm{ADV}_\alpha(\mathrm{XOR})$$

with $\alpha = (1, \mathrm{ADV}(T_1))$, and so Theorem 4 says that

$$\mathrm{ADV}(T) = 1 + \mathrm{ADV}(T_1). \tag{9}$$

From induction hypothesis, we have

$$\mathrm{ADV}(T_1) \geq \tilde{r}(T_1) \tag{10}$$

and by the definition of soft rank, we have

$$\tilde{r}(T) = 1 + \tilde{r}(T_1). \tag{11}$$

Plugging (10) and (11) into (9), we have

$$\mathrm{ADV}(T) \geq \tilde{r}(T),$$

as required.

Next we examine Case 2, where the root of T is a normal node labeled with some variable x_i, and it has two child subtrees of depth at most $d-1$. We denote by T_1 and T_2 the left child subtree and the right child subtree of T, respectively. Obviously, T computes the function $\mathrm{MUX}(x_i, T_1, T_2)$. We may assume without loss of generality that $\mathrm{ADV}(T_1) \leq \mathrm{ADV}(T_2)$, since otherwise we can examine the equivalent function $\mathrm{MUX}(\bar{x}_i, T_2, T_1)$.

Applying Theorems 2, we have

$$\mathrm{ADV}(T) = \mathrm{ADV}_\alpha(\mathrm{MUX}) \tag{12}$$

with $\alpha = (1, \mathrm{ADV}(T_1), \mathrm{ADV}(T_2))$. From induction hypothesis, we have

$$\mathrm{ADV}(T_1) \geq \tilde{r}(T_1) \text{ and } \mathrm{ADV}(T_2) \geq \tilde{r}(T_2). \tag{13}$$

Now we consider the two subcases where $\tilde{r}(T_1) \leq \tilde{r}(T_2)$ and $\tilde{r}(T_2) \leq \tilde{r}(T_1)$.

For the subcase where $\tilde{r}(T_1) \leq \tilde{r}(T_2)$, we let $\beta = (1, \tilde{r}(T_1), \tilde{r}(T_2))$. By (13) we have $\alpha \geq \beta$. So Proposition 3 implies that

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

and by Lemma 1 we have

$$\begin{aligned} \mathrm{ADV}_\beta(\mathrm{MUX}) &\geq \tilde{r}(T_1) + \sqrt{(\tilde{r}(T_2) - \tilde{r}(T_1))^2 + 1} \\ &= \tilde{r}(T), \end{aligned} \tag{14}$$

where the last equality is from the definition of soft rank, since we have assumed $\tilde{r}(T_1) \leq \tilde{r}(T_2)$. By (12) and (14), we have

$$\mathrm{ADV}(T) \geq \tilde{r}(T),$$

as required.

Finally we examine the subcase where $\tilde{r}(T_2) \leq \tilde{r}(T_1)$. In this case, we let $\beta = (1, \tilde{r}(T_2), \tilde{r}(T_1))$. By the assumption of $\mathrm{ADV}(T_1) \leq \mathrm{ADV}(T_2)$ and the induction hypothesis (13), we have

$$\tilde{r}(T_2) \leq \tilde{r}(T_1) \leq \mathrm{ADV}(T_1)$$

and

$$\tilde{r}(T_1) \leq \mathrm{ADV}(T_1) \leq \mathrm{ADV}(T_2).$$

Hence we have $\beta \leq \alpha$. So applying Proposition 3 and Lemma 1, we have

$$\begin{aligned} \mathrm{ADV}_\alpha(\mathrm{MUX}) &\geq \mathrm{ADV}_\beta(\mathrm{MUX}) \\ &\geq \tilde{r}(T_2) + \sqrt{(\tilde{r}(T_1) - \tilde{r}(T_2))^2 + 1} \\ &= \tilde{r}(T). \end{aligned}$$

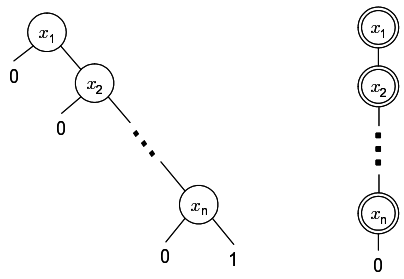

Figure 3: Decision trees that represent the AND (left) and PARITY (right) functions. Recall that the double circles indicate parity nodes.

The inequality above and (12) gives

$$\mathrm{ADV}(T) \geq \tilde{r}(T),$$

as required. □

Using the theorem above together with Theorem 1, we immediately obtain our main result: The soft rank of a decision tree T in the class $\mathrm{DT}_\oplus$ gives a lower bound on the quantum query complexity of the function that T represents. Moreover, from the main result we can show that for functions represented by decision trees in $\mathrm{DT}_\oplus$, there is at most a quadratic gap in the query complexity between deterministic and quantum algorithms.

Corollary 1 *For any function f that can be represented by a decision tree T in $DT_\oplus$, and for any $\epsilon \in [0, 1/2)$,*

$$Q_\epsilon(f) = \Omega(\tilde{r}(T))$$

and

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

Proof. Let T be a non-redundant decision tree in $\mathrm{DT}_\oplus$ that represents a function f. Let d be the depth of T. By Theorems 1 and 7, and Proposition 2, we have

$$Q_\epsilon(f) = \Omega(\tilde{r}(T)) = \Omega(\sqrt{d}).$$

Since d is an upper bound on $D(f)$, the corollary follows. □

7 Examples

We demonstrate the usefulness of Theorem 7 by showing lower bounds on ADV for some functions.

7.1 The AND function

The n-bit AND function, denoted by AND_n, can be recursively represented as

$$\mathrm{AND}_n = \mathrm{MUX}(x_n, 0, \mathrm{AND}_{n-1}).$$

Equivalently, AND_n has a decision tree representation (the left tree of Figure 3) of depth n and soft rank $\sqrt{n}$. So Theorem 7 says that $\mathrm{ADV}(\mathrm{AND}_n) \geq \sqrt{n}$. Note that the rank of the tree is 1, and so the rank does not give a good lower bound in this case.

7.2 The PARITY function

The n-bit PARITY function, denoted by PARITY_n, can be recursively represented as

$$\mathrm{PARITY}_n = \mathrm{XOR}(x_n, \mathrm{PARITY}_{n-1}).$$

Equivalently, PARITY_n has a decision tree representation (the right tree of Figure 3) of depth n and soft rank n. So Theorem 7 says that $\mathrm{ADV}(\mathrm{PARITY}_n) \geq n$.

7.3 Read-once AVL decision trees

We consider functions represented as read-once AVL decision trees. An AVL tree is a standard decision tree such that for any internal node, the depth of its left child subtree and that of its right child subtree are different by at most one.

Theorem 8 *For any read-once AVL tree T of depth d,*

$$\mathrm{ADV}(T) \geq \lceil d/2 \rceil.$$

Proof. Let

$$r_d = \min\{r(T) \mid T \text{ is an AVL decision tree of depth } d\}.$$

By an easy induction on d, we can show that

$$r_d = \lceil d/2 \rceil,$$

which implies the theorem. □

Note that any read-once AVL decision tree T of depth d has decision tree complexity $D(T) \leq d$. So the theorem above implies $Q_\epsilon(T) = \Omega(D(T))$. In other words, it provides a class of functions for which the bounded error quantum query complexity is not smaller than the classical decision tree complexity up to a constant factor.

8 Concluding remarks

We considered Boolean functions that can be represented by read-once decision trees with parity nodes, and showed that the soft rank of the decision tree is a lower bound on the bounded error quantum query complexity of the function. The soft rank measures how well-balanced a given tree is, and the lower bound suggests that the quantum query complexity of a function is highly correlated to the *balancedness* of its decision tree representation. The soft rank is easily evaluated for several functions and explains quickly why the OR function can be computed in $\sqrt{n}$ queries whereas the parity function requires n queries.

We also showed that for the class of functions that are represented by read-once decision trees with parity nodes, 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). The composition theorem of Høyer et al. (2006) together with the bounds for XOR and MUX gives our lower bound. To derive the bounds on $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$, we explicitly constructed adversary matrices, which were obtained in the following way. Using a semi-definite programming package, we numerically calculated the values of $\mathrm{ADV}_\alpha(\mathrm{XOR})$ and $\mathrm{ADV}_\alpha(\mathrm{MUX})$ 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 $\mathrm{ADV}_\alpha(\mathrm{XOR}) = a + b$ for $\alpha = (a, b)$. However, the adversary matrix Γ for MUX we gave in Figure 2 is not optimal. Based on the observation of the numerical calculations, we are pretty sure that the optimal matrix for $\mathrm{ADV}_\alpha(\mathrm{MUX})$ 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 (6) with the one above. The value $\mathrm{ADV}_\alpha(\mathrm{MUX})$ 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 Høyer et al. (2007), who introduce an adversary bound $\mathrm{ADV}^{\pm}$, which always gives larger lower bounds than ADV. But it seems much more complicated calculations involved.

Acknowledgments

We would like to thank the anonymous referees for their helpful comments.

References

Aaronson, S. (2008), 'The limits of quantum computers', *Scientific American* **298**(3), 62–69.

Ambainis, A. (2002), 'Quantum lower bounds by quantum arguments', *Journal of Computer and System Sciences* **64**(4), 750–767.

Ambainis, A. (2006), 'Polynomial degree vs. quantum query complexity', *Journal of Computer and System Sciences* **72**(2), 220–238.

Barnum, H. & Saks, M. (2004), 'A lower bound on the quantum query complexity of read-once functions', *Journal of Computer and System Sciences* **69**(2), 244–258.

Barnum, H., Saks, M. & Szegedy, M. (2003), Quantum decision trees and semidefinite programming, *in* 'Proc. of 18th IEEE Conference on Computational Complexity', pp. 179–193.

Beals, R., Buhrman, E., Cleve, R., Mosca, M. & de Wolf, R. (2001), 'Quantum lower bounds by polynomials', *Journal of the ACM* **48**(4), 778–797.

Ehrenfeucht, A. & Haussler, D. (1989), 'Learning decision trees from random examples', *Information and Computation* **82**(3), 231–246.

Grover, L. (1996), A fast quantum mechanical algorithm for database search, *in* 'Proc. 28th ACM Symposium on Theory of Computing (STOC '96)', pp. 212–219.

Høyer, P., Lee, T. & Špalek, R. (2006), 'Tight adversary bounds for composite functions', arXiv:quant-ph/0509067v3.

Høyer, P., Lee, T. & Špalek, R. (2007), Negative weights make adversaries stronger, *in* 'Proc. 39th ACM Symposium on Theory of Computing (STOC '07)', pp. 526–535.

Laplante, S. & Magniez, F. (2004), Lower bounds for randomized and quantum query complexity using kolmogorov arguments, *in* 'Proc. 19th IEEE Conference on Computational Complexity', pp. 294–304.

Midrijanis, G. (2004), 'Exact quantum query complexity for total boolean functions', arXiv:quant-ph/0403168v2.

Simon, D. (1997), 'On the power of quantum computation', *SIAM Journal on Computing* **26**, 1474–1483.

Špalek, R. & Szegedy, M. (2005), All quantum adversary methods are equivalent, *in* 'Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)', pp. 1299–1322.

A Proof of Theorem 6

Here we give a proof of Theorem 6.

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.

$$\mathrm{ADV}_\alpha(\mathrm{MUX}) \le \frac{b + c + \sqrt{(c-b)^2+4}}{2} \qquad (15)$$

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

To show (15), we use the dual formulation of the spectral adversary method. In other words, we construct a feasible solution of the optimization problem $\mathrm{MM}_\alpha(\mathrm{MUX})$, from which we derive an upper bound on $\mathrm{ADV}_\alpha(\mathrm{MUX})$.

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

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

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

$$\frac{1}{\sum_{i:x_i \ne y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i} \le A. \qquad (16)$$

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

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

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

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

$$\begin{aligned} \frac{1}{\sum_{i:x_i \ne y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i} &= \frac{1}{\sqrt{dj}} \\ &= A. \end{aligned}$$

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

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

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

$$\begin{aligned} \frac{1}{\sum_{i:x_i \ne y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i} &= \frac{c}{\sqrt{(1-f-g)(1-j)}} \\ &\le A. \end{aligned}$$

To see why the last inequality holds, we show that

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

The inequality holds because

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

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

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

Consequently,

$$\begin{aligned} &\frac{1}{\sum_{i:x_i \ne y_i} \sqrt{p_x(i)p_y(i)}/\alpha_i} \\ &= \left(\frac{g}{b} + \frac{1-f-g}{c}\right)^{-1} \\ &\le A. \end{aligned}$$

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

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

□