On Directional vs. Undirectional Randomized Decision Tree Complexity for Read-Once Formulas

Kazuyuki Amano[1]

[1] Department of Computer Science
Gunma University
Tenjin 1-5-1, Kiryu, Gunma 376-8515, Japan
Email: amano@cs.gunma-u.ac.jp

Abstract

We investigate the relationship between the directional and the undirectional complexity of read-once Boolean formulas on the randomized decision tree model. It was known that there is a read-once Boolean formula such that an optimal randomized algorithm to evaluate it is not directional. This was first pointed out by Saks and Wigderson (1986) and an explicit construction of such a formula was given by Vereshchagin (1998). We conduct a systematic search for a certain class of functions and provide an explicit construction of a read-once Boolean formula f on n variables such that the cost of the optimal directional randomized decision tree for f is $\Omega(n^{\alpha})$ and the cost of the optimal randomized undirectional decision tree for f is $O(n^{\beta})$ with $\alpha-\beta > 0.0101$. This is the largest known gap so far.

Keywords: Computational Complexity, Randomized Decision Tree, Read-Once Formula, Directional Algorithm

1 Introduction and Results

Boolean decision tree is one of the most simple models to compute Boolean functions. In this model, a basic operation is evaluating an input variable and the algorithm branches according to the observed value. The deterministic decision tree complexity of a Boolean function f, denoted by $D(f)$, is the number of variables examined by the most efficient decision tree for f on the worst case input to that tree.

A randomized Boolean decision tree is a probability distribution on the deterministic decision trees that computes f. The randomized decision tree complexity for a Boolean function f, denoted by $R(f)$, is the maximum (over all input settings) average (according to the distribution) number of variables evaluated in the optimal distribution. Both models of decision trees have been studied extensively in many contexts (see e.g., a good survey by Buhrman and de Wolf (2002)).

In this paper, we investigate the randomized decision tree complexity for *read-once functions*. Recall that a Boolean function is called read-once if it can be represented by a Boolean formula consisting of AND/OR gates of unbounded fan-in in which each input variable appears exactly once. It is folklore that $D(f) = n$ for every read-once function on n variables. An outstanding open question is to give a lower bound on the minimum of $R(f)$ over *all* read-once Boolean functions. It is conjectured that $R(f) = \Omega(n^{\alpha})$ where $\alpha = \log_2\left(\frac{1+\sqrt{33}}{4}\right) = 0.753\cdots$ for every read-once Boolean function n variables (Saks &Wigderson (1986)). The current best lower bound is $R(f) = \Omega(n^{0.51})$ due to Heiman & Wigderson (1991). It was also shown that $R(f) = \Omega(n/2^d)$ for every function f that can be computed by a depth-d read-once threshold formula by Heiman et al. (1993).

In general, the problem for estimating the randomized decision tree complexity for a given Boolean function is a very difficult task. A natural intermediate step is to investigate a restricted class of algorithms. In this work, we focus on the restriction called *directional*, which was first introduced by Saks &Wigderson (1986). An algorithm is directional if it reads a variable in a sub-formula, then it has to evaluate the sub-formula completely before reading another variable that appears in a different sub-formula. For the read-once function of complete binary AND/OR tree, Saks &Wigderson (1986) showed that the randomized decision tree complexity $R(f)$ is $\Theta(n^{\alpha})$ where $\alpha = \log_2\left(\frac{1+\sqrt{33}}{4}\right)$, and the optimal algorithm is directional. On the other hand, it is also known that there is a read-once function such that a directional algorithm is not optimal for evaluating it. This was first pointed out by Saks &Wigderson (1986) without giving an example of such a formula.

In order to discuss the directional algorithms, there is a point that should be clarified. Consider a formula $f = f_1 \wedge f_2 \wedge f_3 \wedge f_4$ using an AND gate of fan-in four, a directional algorithm for evaluating f can evaluate f_i's in any order. On the other hand, if we represent f by $(f_1 \wedge f_2) \wedge (f_3 \wedge f_4)$, then a directional algorithm cannot evaluate in the order, say, f_1, f_3, f_2, f_4. These two cases are distinguished by whether we allow a consecutive use of AND gates or OR gates in a formula to represent a Boolean function. The first case, in which we don't allow a consecutive use of AND gates (OR gates, resp.), is referred as the *alternating setting*, and the other case is referred as the *non-restricted setting*. In the alternating setting, we restrict to formulas in which the AND and OR gates appear alternately. Note that the class of directional algorithms in the alternating setting is wider than that in the non-restricted setting.

Vereshchagin (1998) first provided an explicit construction of a read-once formula on n variables that has the gap $\Omega(n^{0.0058})$ between the randomized decision tree complexity and the cost of the optimal directional algorithm in the alternating setting, and also remarked that the exponent can be improved to 0.0077. As pointed out by Vereshchagin (1998), if the largest gap is not significantly greater than this, then we can restrict ourselves with directional algorithms

This work is partially supported by Grant-in-Aid for Scientific Research (C) 21500005.

for computing read-once functions being sure that our loss is very small. This would be nice since the optimal directional algorithm is easy to find (see Saks &Wigderson (1986) or Vereshchagin (1998)).

Motivated by this, we investigate the relationship between the directional and unrestricted complexity for read-once functions more extensively. In particular, we conduct a systematic search for a certain class of formulas to see the merit of using *undirectional* algorithms. As a result, we give an explicit construction of a formula on n variables that has the gap $\Omega(n^{0.0101})$ in the alternating setting and the one that has the gap $\Omega(n^{0.0110})$ in the non-restricted setting. Our construction is essentially the recursion of a certain depth-two formula of the form "AND of ORs". We also discuss the limit of such a CNF based construction. These are the main contributions of this work and are described in Section 3.

Before closing this section, we describe some related work. For non read-once functions, one of the most well studied functions is the recursive majority-of-three function. The directional algorithm is not optimal for this function (Saks &Wigderson (1986), see also Jayram, Kumar & Sivakumar (2003)). The deterministic decision tree complexity is $\Theta(3^h)$, the directional complexity is $\Theta((8/3)^h)$, and the randomized decision tree complexity is between $\Omega((7/3)^h)$ and $o((2.655\cdots)^h) = o((8/3)^h)$, where h is the number of recursions. The last lower bound was recently shown by Jayram, Kumar & Sivakumar (2003). In recent years, the problem on the query complexity in the quantum setting and its relation to the classical setting have also extensively studied (see e.g., Barnum & Saks (2004), Reichardt & Špalek (2008), Reichardt (2009a,b) and the references therein).

2 Decision Trees

Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and f be a Boolean function on X. A *deterministic decision tree* is a rooted ordered binary tree T. Each internal node of T is labeled with a variable $x_i \in X$, and each leaf is labeled with a value 0 or 1. For every internal node, one of the two outgoing edge is labeled by 0 and the other is labeled by 1. When an input $x \in \{0,1\}^n$ is given, a path from the root to a leaf in T is specified in a natural way. The output $T(x)$ for this input is the value on the reached leaf. The depth of a decision tree is the number of edges in a longest path from the root to a leaf. The decision tree complexity of f, denoted by $D(f)$, is the smallest depth among all decision trees that compute f, i,e, $T(x) = f(x)$ for every input x.

A *randomized decision tree* is a distribution on the deterministic decision trees that compute f. The complexity is measured by averaging. The randomized decision tree complexity, denoted by $R(f)$, is the maximum (over all input settings) average (according to the distribution) number of variables evaluated in the optimal distribution.

Example 1 *Let MAJ_3 denote the majority function on 3 variables x_1, x_2 and x_3. It is obvious that $D(f) = 3$ since for every two variables there is an assignment to these variables such that the output is undetermined. We will explain that $R(f) \le 8/3$. Let μ be a uniform distribution over the six ways of ordering of $\{x_1, x_2, x_3\}$ to evaluate. When an input is $(x_1, x_2, x_3) = (0,0,0)$ or $(1,1,1)$, then we only need two queries in any ordering. For any other input, we need to evaluate the third variable if and only if the results of the first two queries are different, which is occurred with probability $2/3$. So the cost for such an input is $1/3 \cdot 2 + 2/3 \cdot 3 = 8/3$. This implies $R(f) \le \max\{2, 8/3\} = 8/3$. Note that in fact $R(f) = 8/3$.* □

3 Gap between Directional and Undirectional

In this section, we investigate the relationship between the directional and the undirectional complexity for a certain class of read-once functions.

A *randomized directional algorithm A* for evaluating $f = f_1 \wedge \cdots \wedge f_k$ (or $f = f_1 \vee \cdots \vee f_k$) is specified by a set of randomized directional algorithms $A_1, \ldots, A_k$ for evaluating $f_1, \ldots, f_k$, respectively, and a probability distribution on the set of permutations of $1, \ldots, k$. A is performed by selecting a permutation σ of $1, \ldots, k$ according to this distribution and evaluating $f_1, \ldots, f_k$ in σ order using $A_1, \ldots, A_k$ until the value of f is known. For example, for $f = (f_1 \wedge f_2) \vee (f_3 \wedge f_4)$, a directional algorithm cannot evaluate sub-functions in the order, say, f_1, f_3, f_2, f_4. Let $d(f)$ denote the minimum expected number of variables tested on a worst case input over all randomized directional algorithms. Similarly, let $d_0(f)$ ($d_1(f)$, respectively) denote the analogous quantity when the input x is restricted so that $f(x) = 0$ ($f(x) = 1$, respectively). Note that $d(f) = \max\{d_0(f), d_1(f)\}$ (Saks &Wigderson (1986, Lemma 3.1)). If we only use AND and OR gates of fan-in two, then the complexity of the best directional algorithm is completely determined.

Theorem 2 (Saks &Wigderson (1986)) *For any read-once formula f with AND/OR gates of fan-in two, the cost of the optimal directional algorithm is given by the following:*

$$d_0(f) = \begin{cases} 1, & \text{if } f \text{ is a single variable,} \\ d_0(g) + d_0(h), & \text{if } f = g \vee h, \\ \Psi(d_0(g), d_0(h), d_1(g), d_1(h)), & \\ & \text{if } f = g \wedge h, \end{cases}$$

$$d_1(f) = \begin{cases} 1, & \text{if } f \text{ is a single variable,} \\ d_1(g) + d_1(h), & \text{if } f = g \wedge h, \\ \Psi(d_1(g), d_1(h), d_0(g), d_0(h)), & \\ & \text{if } f = g \vee h, \end{cases}$$

where

$$\Psi(a_0, b_0, a_1, b_1) = \max\left\{a_0, b_0, \frac{a_0 a_1 + b_0 b_1 + a_1 b_1}{a_1 + b_1}\right\}.$$

□

As noted by Saks &Wigderson (1986), this can naturally be extended to the unbounded fan-in case. If $f = f_1 \vee \cdots \vee f_k$, then $d_0(f) = d_0(f_1) + \cdots + d_0(f_k)$, and if $f = f_1 \wedge \cdots \wedge f_k$, then $d_1(f) = d_1(f_1) + \cdots + d_1(f_k)$. However, the expression of $d_1(f)$ for $f = f_1 \vee \cdots \vee f_k$ and that of $d_0(f)$ for $f = f_1 \wedge \cdots \wedge f_k$ is not so simple. So we only describe the following lemma, which is sufficient for our purpose. Here and hereafter $[k]$ denotes the set $\{1, \ldots, k\}$ for a positive integer k.

Lemma 3 *Suppose that $f = \bigwedge_{i \in [k]} f_i$ for a positive integer k. If there is an index $i \in [k]$ such that for every $j \in [k] \backslash \{i\}$,*

$$d_0(f_i) \ge d_0(f_j) + \sum_{\ell \in [k] \backslash \{j\}} d_1(f_\ell)$$

holds, then $d_0(f) = d_0(f_i)$.

Proof. We show here the proof of the case $k = 3$. The proofs of another cases are similar. Let $f = f_1 \wedge f_2 \wedge f_3$. Without loss of generality, we assume that $d_0(f_1) \geq d_0(f_2) + d_1(f_1) + d_1(f_3)$ and $d_0(f_1) \geq d_0(f_3) + d_1(f_1) + d_1(f_2)$.

A directional algorithm A for evaluating f is specified by the probability distribution on the set of permutations on $\{1, 2, 3\}$ that represents the order of subfuntions evaluated by A. For three distinct integers $k_1, k_2, k_3 \in \{1, 2, 3\}$, let $p_{k_1k_2k_3}$ denote the probability that the order of functions evaluated by A is f_{k_1}, f_{k_2} and f_{k_3}. An easy calculation shows that $d_0(f)$ is given by

$$\min_{p's} \max \left\{ \begin{array}{l} d_0(f_1) + (p_{213} + p_{231} + p_{321})d_1(f_2) \\ \qquad +(p_{231} + p_{312} + p_{321})d_1(f_3), \\ d_0(f_2) + (p_{123} + p_{132} + p_{312})d_1(f_1) \\ \qquad +(p_{132} + p_{312} + p_{321})d_1(f_3), \\ d_0(f_3) + (p_{123} + p_{132} + p_{213})d_1(f_1) \\ \qquad +(p_{123} + p_{213} + p_{231})d_1(f_2) \end{array} \right\} \quad (1)$$

where the minimization is over p_*'s such that the sum of them is equal to 1. If we put $p_{213} = p_{231} = p_{312} = p_{321} = 0$, which means that A first evaluates f_1, then we have $d_0(f) \leq d_0(f_1)$. Since it is trivial that $d_0(f) \geq d_0(f_1)$ by Eq. (1), we can conclude that $d_0(f) = d_0(f_1)$. This completes the proof. □

In this work, we choose the class of depth-two formulas of the form "AND of ORs" as a building block of our construction.

Definition 4 *Let k be a positive integer and $V = (v_1, \ldots, v_k)$ be a sequence of k integers such that $v_1 \leq \cdots \leq v_k$. Let f_V denote a read-once function on the variable set $\{x_{i,j} \mid i \in [k], j \in [v_i]\}$ defined as $f_V := \bigwedge_{i\in[k]} f_i$, where $f_i := \bigvee_{j\in[v_i]} x_{i,j}$. For a sequence V, V^* denotes the largest integer among all integers in V, and $||V||$ denotes the sum of all integers in V, i.e., $||V|| := \sum_{i=1}^{k} v_i$ and it is equal to the number of variables in f_V.*

As to the construction by Vereshchagin (1998) we use a trick that the directional complexity of $(f_1 \wedge f_2) \wedge f_3$ may be different from that of $f_1 \wedge (f_2 \wedge f_3)$. A function that has a big discrepancy seems to be good for our purpose. Some more precisely, we want a depth-two formula f_V such that $\log_{||V||}\left(\frac{d(f_V^{worst})}{R(f_V)}\right)$ is large, where $d(f_V^{worst})$ denotes the largest value of the directional complexity $d(f_V^T)$ among all AND/OR formula f_V^T consisting only of gates of *fan-in two* that computes f_V. Through an experimental computation using a computer, we found that every good candidate has a common property.

Definition 5 *A sequence $V = (v_1, \ldots, v_k)$ with $v_1 \leq \cdots \leq v_k$ is said to be good if v_k is odd, $v_{k-1} = \frac{v_k - 1}{2}$ and $d(f_V) = d_0(f_V) = d_1(V) = V^*$.* □

Example 6 *A sequence $V = (1, 3, 7)$ is good. Below we verify this as an illustrative example. Let OR_u denote the OR of u variables. It is easy to see that the optimal directional algorithm for OR_u is to evaluate variables in a random order until the value 1 is found, and so we have $d_0(OR_u) = u$ and $d_1(OR_u) = (u + 1)/2$. Since $d_1(f_V) = d_1(OR_1 \wedge OR_3 \wedge OR_7) = d_1(OR_1) + d_1(OR_3) + d_1(OR_7)$, we have $d_1(f_V) = 1 + 2 + 4 = 7$. By Eq. (1) and $d_0(OR_7) = 7$, we have $d_0(f_V) \geq 7$. By using Eq. (1) again with $p_{123} = 1$ and $p_{132} = p_{213} = p_{231} = p_{312} = p_{321} = 0$, we have $d_0(f_V) \leq 7$, and hence $d_0(f_V) = 7$. This concludes that $d(f_V) = d_0(f_V) = d_1(f_V) = 7 = V^*$. On the other hand, it is easy to calculate (by using Theorem 2) that $d_0((OR_1 \wedge OR_7) \wedge OR_3) = 51/7 = 7.28\cdots$, and $d_1((OR_1 \wedge OR_7) \wedge OR_3)) = 7$.* □

In the following, we fix the sequence $V = (11, 12, 12, 39, 79)$. Below we verify that this sequence is also good.

Consider a directional algorithm to evaluate f_V as $f'_V := (\mathrm{OR}_{11} \wedge \mathrm{OR}_{12} \wedge \mathrm{OR}_{12}) \wedge (\mathrm{OR}_{39} \wedge \mathrm{OR}_{79})$. We use Eq. (1) with a uniform distribution (i.e., $p_* = 1/6$) to show that

$$\begin{aligned} & d_0(\mathrm{OR}_{11} \wedge \mathrm{OR}_{12} \wedge \mathrm{OR}_{12}) \\ & \quad \leq \ \max\{11 + 6.5, 12 + 6.25\} = 18.25 < 19. \end{aligned}$$

By Theorem 2, Lemma 3 and the facts that $d_0(\mathrm{OR}_u) = u$ and $d_1(\mathrm{OR}_u) = (u + 1)/2$, we have

$$\begin{aligned} d_1(\mathrm{OR}_{11} \wedge \mathrm{OR}_{12} \wedge \mathrm{OR}_{12}) &= 6 + \frac{13}{2} + \frac{13}{2} = 19, \\ d_0(\mathrm{OR}_{39} \wedge \mathrm{OR}_{79}) &= 79, \\ d_1(\mathrm{OR}_{39} \wedge \mathrm{OR}_{79}) &= 20 + 40 = 60, \end{aligned}$$

and hence by Theorem 2, we have

$$\begin{aligned} V^* = 79 \leq \ d_0(f_V) &\leq d_0(f'_V) = 79, \\ d_1(f_V) &= \sum_{i=1}^{5} \frac{v_i + 1}{2} = 79. \end{aligned}$$

This completes the proof for $d(f_V) = d_0(f_V) = d_1(f_V) = 79 = V^*$.

Definition 7 *For a Boolean function on n variables and an integer $k \geq 0$, let $f^{(k)}$ denote a Boolean function on n^k variables defined as follows: $f^{(0)}$ is a single variable and*

$$\begin{aligned} f^{(k)} \ = \ & f(f^{(k-1)}(x_1, \ldots, x_{n^{k-1}}), \\ & \ldots, f^{(k-1)}(x_{n^k - n^{k-1}+1}, \ldots, x_{n^k})), \end{aligned}$$

for $k \geq 1$.

Below we give an explicit construction of a Boolean function that has a large gap between the directional complexity and the undirectional complexity.

Our function is the recursive function that uses the following depth-four (AND-OR-AND-OR) formula as a building block. Let

$$f = ((f_1 \wedge f_2 \wedge f_3 \wedge f_5) \vee z) \wedge f_4,$$

where $f_i = \bigvee_{j\in[v_i]} x_{i,j}$. Recall that we put $V = (11, 12, 12, 39, 79)$, and so f has $||V|| + 1 = 154$ input variables. Note that the construction by Vereshchagin (1998) is the recursive function that uses a certain formula of depth 6 as its building block.

Theorem 8 *Let $f^{(k)}$ be a Boolean function defined as above, and let N be the total number of input variables in $f^{(k)}$. Then $d(f^{(k)})/R(f^{(k)}) = \Omega(N^{0.0101})$.*

Proof. We first show that $R(f^{(k)}) = O(80^k)$. Since $R(f^{(k)}) \leq R(f)^k$, it is sufficient to show that $R(f) \leq 80$. This can easily be verified by considering the following algorithm evaluating f: Evaluate first z. The worst case is $z = 0$. In this case, the remaining function is equivalent to f_V. An undirectional algorithm described above (in the proof of the goodness of V) can evaluate this function with cost at most 79.

We now show the lower bound on $d(f^{(k)})$. Let $f_i^{(k)}$ denote the function $\bigvee_{j\in[v_i]} f^{(k)}(\overrightarrow{x_{i,j}})$, where $\overrightarrow{x_{i,j}}$

represents the list of n^k input variables for $f^{(k)}$. Similarly, $g^{(k)}$ denotes $\bigwedge_{i\in\{1,2,3,5\}} f_i^{(k)}$, and $h^{(k)}$ denotes $g^{(k)} \vee f^{(k)}$. Note that $f^{(k+1)} = h^{(k)} \wedge f_4^{(k)}$. Recall that every function is read-once.

Below we need the inequality $d_0(f^{(k)}) \geq d_1(f^{(k)})$, which can be verified by an induction on k. The base case $k = 0$ is trivial since $d_0(f^{(0)}) = d_1(f^{(0)}) = 1$. The induction step is verified from Eqs. (3) and (4) by observing that $\frac{497256}{6341}(\sim 78.4) > \frac{743}{10}(= 74.3)$ and $\frac{38927}{6341}(\sim 6.14) > \frac{397}{80}(\sim 4.96)$.

It is easy to see that $d_0(f_i^{(k)}) = v_i d_0(f^{(k)})$, $d_1(f_i^{(k)}) = \frac{v_i-1}{2}d_0(f^{(k)}) + d_1(f^{(k)})$. Since $d_1(\bigwedge_i f_i^{(k)}) = \sum_i d_1(f_i^{(k)})$, we have

$$d_1(g^{(k)}) = 55d_0(f^{(k)}) + 4d_1(f^{(k)}).$$

For every $j \in \{1, 2, 3\}$, we have

$$\begin{aligned} d_0(f_j^{(k)}) + & \sum_{\ell\in\{1,2,3,5\}\setminus\{j\}} d_1(f_\ell^{(k)}) \\ & \leq \; 61.5d_0(f^{(k)}) + 3d_1(f^{(k)}). \end{aligned}$$

Since $d^0(f^{(k)}) \geq d^1(f^{(k)})$, the RHS of the above inequality is at most $64.5d_0(f^{(k)}) < 79d_0(f^{(k)}) = d_0(f_5^{(k)})$. Hence we can apply Lemma 3 with $i = 5$ to get

$$d_0(g^{(k)}) = 79d_0(f^{(k)}).$$

By Theorem 2, we have

$$\begin{aligned} d_0(h^{(k)}) &= d_0(g^{(k)} \vee f^{(k)}) \\ &= d_0(g^{(k)}) + d_0(f^{(k)}) = 80d_0(f^{(k)}), \\ d_1(h^{(k)}) &= \Big\{d_0(g^{(k)})d_1(g^{(k)}) + d_0(f^{(k)})d_1(f^{(k)}) \\ &\quad +d_0(g^{(k)})d_0(f^{(k)})\Big\}/\Big\{d_0(g^{(k)})+d_0(f^{(k)})\Big\} \\ &= \frac{553}{10}d_0(f^{(k)}) + \frac{317}{80}d_1(f^{(k)}). \qquad (2) \end{aligned}$$

Eq. (2) follows from the fact that the RHS of (2) is greater than $\max\{d_1(g^{(k)}), d_1(f^{(k)})\}$. This holds since $d_0(f^{(k)}) \geq d_1(f^{(k)})$. Finally, we have

$$\begin{aligned} d_1(f^{(k+1)}) &= d_1(h^{(k)} \wedge f_4^{(k)}) = d_1(h^{(k)}) + d_1(f_4^{(k)}) \\ &= \frac{553}{10}d_0(f^{(k)}) + \frac{317}{80}d_0(f^{(k)}) \\ &\qquad + 19d_0(f^{(k)}) + d_1(f^{(k)}) \\ &= \frac{743}{10}d_0(f^{(k)}) + \frac{397}{80}d_1(f^{(k)}), \qquad (3) \end{aligned}$$

$$\begin{aligned} & d_0(f^{(k+1)}) \\ &= \Big\{d_0(h^{(k)})d_1(h^{(k)}) + d_0(f_4^{(k)})d_1(f_4^{(k)}) \\ &\quad +d_1(h^{(k)})d_1(f_4^{(k)})\Big\} \,/\, \Big\{d_1(g^{(k)}) + d_1(f_4^{(k)})\Big\} \\ &= \Big\{\frac{62157}{10}d_0(f^{(k)})^2 + \frac{38927}{80}d_0(f^{(k)})d_1(f^{(k)}) \\ &\quad +\frac{317}{80}d_1(f^{(k)})^2\Big\}/\Big\{\frac{743}{10}d_0(f^{(k)}) + \frac{397}{80}d_1(f^{(k)})\Big\} \\ &\geq \Big\{\frac{62157}{10}d_0(f^{(k)})^2 + \frac{38927}{80}d_0(f^{(k)})d_1(f^{(k)}) \\ &\quad +\frac{317}{80}d_1(f^{(k)})^2\Big\}/\Big\{\frac{6341}{80}d_0(f^{(k)})\Big\} \\ &> \frac{497256}{6341}d_0(f^{(k)}) + \frac{38927}{6341}d_1(f^{(k)}). \qquad (4) \end{aligned}$$

Here we use the facts $d_0(g^{(k)}) \leq d_1(g^{(k)}) + d_0(f_4^{(k)})$ and $d_0(f_4^{(k)}) \leq d_1(f_4^{(k)}) + d_0(g^{(k)})$ to derive the first equality, and use $d_0(f^{(k)}) \geq d_1(f^{(k)})$ to derive the second last inequality. We can conclude that

$$\begin{aligned} & \begin{pmatrix} d_0(f^{(k+1)}) \\ d_1(f^{(k+1)}) \end{pmatrix} \\ \geq & \begin{pmatrix} \frac{497256}{6341} & \frac{38927}{6341} \\ \frac{743}{10} & \frac{397}{80} \end{pmatrix} \begin{pmatrix} d_0(f^{(k)}) \\ d_1(f^{(k)}) \end{pmatrix}. \qquad (5) \end{aligned}$$

Now we define $d_0'(k)$ and $d_1'(k)$ recursively as follows: $d_0'(0) = d_1'(0) = 1$ and

$$\begin{aligned} & \begin{pmatrix} d_0'(k+1) \\ d_1'(k+1) \end{pmatrix} \\ = & \begin{pmatrix} \frac{497256}{6341} & \frac{38927}{6341} \\ \frac{743}{10} & \frac{397}{80} \end{pmatrix} \begin{pmatrix} d_0'(k) \\ d_1'(k) \end{pmatrix}. \qquad (6) \end{aligned}$$

Obviously, $d_0'(k)$ and $d_1'(k)$ give the lower bounds on $d_0(f^{(k)})$ and $d_1(f^{(k)})$, respectively. Since the eigenvalues of the matrix in Eq. (6) are $\gamma_1 = 84.1772\cdots$ and $\gamma_2 = -0.7955\cdots$, we have $d_0(f^{(k)}) \geq d_0'(k) = \Theta(\gamma_1^k)$, and $d_1(f^{(k)}) \geq d_1'(k) = \Theta(\gamma_1^k)$, and hence $d(f^{(k)}) = \Omega(\gamma_1^k) = \Omega(N^{\log_{||V||}(\gamma_1)}) = \Omega(N^{0.88008\cdots})$, where N denotes the total number of input variables of $f^{(k)}$. On the other hand, we have shown that $R(f^{(k)}) = O(80^k) = O(N^{\log_{||V||}(80)}) = O(N^{0.86997\cdots})$. This implies $d(f^{(k)})/R(f^{(k)}) = \Omega(N^{0.0101})$, completing the proof of the theorem. □

It should be noted that our upper bound $R(f^{(k)}) = O(80^k)$ for $f^{(k)}$ is not far from optimal. We can show that $R(f^{(k)}) = \Omega(79^k)$ since $f^{(k)}$ will be the OR of 79^k variables under a suitable partial restriction.

The choice of the sequence V is totally based on numerical computation using a computer. Given any good sequence V, we can easily follow the proof of the above theorem. The randomized complexity $R(f_V^{(k)})$ is $O((V^* + 1)^k)$ for any good V. The computation of the matrix in Eq.(5) is just a routine work. In fact, it is uniquely determined by only the values of k and v_k, i.e., the length of a sequence and the maximum fan-in of OR gates. This makes the exhaustive search feasible. If we restrict ourselves to good sequences then it is conceivable that there is no better sequence even if we consider a longer (or shorter) sequence or a sequence including a larger integer.

For the non-restricted setting we choose the sequence $V = (2, 8, 9, 23, 47)$. The building block is now a depth-three (AND-AND-OR) formula

$$f \;=\; (f_1 \wedge f_2 \wedge f_3 \wedge f_5) \wedge f_4, \qquad (7)$$

where $f_i = \bigvee_{j\in[v_i]} x_{i,j}$. This is a read-once function on $||V|| = 89$ variables.

It is easy to check that $R(f) \leq 47$ by considering an undirectional algorithm that evaluates f as $((\mathrm{OR}_8 \wedge \mathrm{OR}_9) \wedge \mathrm{OR}_2) \wedge (\mathrm{OR}_{31} \wedge \mathrm{OR}_{47})$. Note that, in fact, $R(f) = 47$ since f *contains* OR_{47} as a subfunction. By an analogous argument to the proof of Theorem 8, we have $R(f^{(k)}) = O(47^k)$ and $d(f^{(k)}) \geq \Omega((49.3951\cdots)^k)$. This gives a slightly larger gap $d(f^{(k)})/R(f^{(k)}) = \Omega(N^{0.86882})/O(N^{0.85776}) = \Omega(N^{0.0110})$.

It is interesting to see whether this exponent (~ 0.01) can significantly be improved by considering a construction based on a formula generated by a non-good sequence. Let f be a read-once Boolean function that represented by a CNF formula. Let $d(f^{worst})$ ($d(f^{best})$, resp.) be the largest (smallest, resp.) value of the directional complexity $d(f^T)$ among all AND/OR formula f^T consisting only of gates of fan-in two that computes f.

As long as we follow the argument described in the proof of Theorem 8, the best achievable exponent would be at most $\alpha = \log_{||V||}\left(\frac{d(f_V^{worst})}{d(f_V^{best})}\right)$. By an exhaustive search for $k \le 5$ and $1 \le v_1 \le \cdots \le v_k \le 100$, we found that the sequence $V = (1, 1, 5, 11)$ gives the largest value $\alpha = 0.0167\cdots$. This is again a good sequence. An easy computation shows $d(f_V^{best}) = 11$ and $d(f_V^{worst}) = d((((\mathrm{OR}_1 \wedge \mathrm{OR}_1) \wedge \mathrm{OR}_{11}) \wedge \mathrm{OR}_5) = 11 + \frac{6}{11}$. This suggests that an approach based on the recursive use of a depth-two formula would not yield a significant improvement.

A final remark is on why we don't use this sequence in our construction. If we use this sequence $V = (1, 1, 5, 11)$ to define a building block of the construction (instead of Eq. (7)), then the matrix form corresponding to Eq. (6) is calculated to $\begin{pmatrix} 75/11 & 49/11 \\ 7 & 4 \end{pmatrix}$, and its largest eigenvalue is ~ 11.17. This only gives the exponent of $\log_{||V||}\left(\frac{11.17}{11}\right) \sim 0.0053$.

Acknowledgment

The author would like to thank anonymous reviewers for their comments that improved the presentation of the paper.

References

Barnum, H. & Saks, M. (2004), 'A Lower Bound on the Quantum Query Complexity of Read-Once Functions', *J. Comput. Syst. Sci.* **69**(2), 244–258.

Buhrman, H. & de Wolf, R. (2002), 'Complexity Measures and Decision Tree Complexity: A Survey', *Theoret. Comput. Sci.* **288**(1), 21–43.

Heiman, R., Newman, I. & Wigderson, A. (1993), 'On Read-Once Threshold Formulae and Their Randomized Decision Tree Complexity', *Theoret. Comput. Sci.* **107**(1), 63–76.

Heiman, R. & Wigderson, A. (1991), 'Randomized vs. Deterministic Decision Tree Complexity for Read-Once Boolean Functions', *Computational Complexity* **1** 311-329.

Jayram, T.S., Kumar, R. & Sivakumar, D. (2003), Two Applications of Information Complexity, *in* Proc. of STOC '03, 673–682.

Reichardt, B.W., & Špalek, R. (2008), Span-program-based Quantum Algorithm for Evaluating Formulas, *in* Proc. of STOC '08, 103–112.

Reichardt, B.W. (2009), 'Span-program-based Quantum Algorithm for Unbalanced Formulas', `arXiv:0907.1622 [quant-ph]`.

Reichardt, B.W. (2009), 'Faster Quantum Algorithm for Evaluating Game Trees', `arXiv:0907.1623 [quant-ph]`.

Saks, M. & Wigderson, A. (1986), Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees, *in* Proc. of FOCS '86, 29–38.

Snir, M. (1985), 'Lower Bounds for Probabilistic Linear Decision Trees', *Theoret. Comput. Sci.* **38**, 69–82.

Vereshchagin, N.K. (1998), 'Randomized Boolean Decision Trees: Several Remarks', *Theoret. Comput. Sci.* **207**, 329–342.