On the Monte Carlo Boolean Decision Tree Complexity of Read-Once Formulae

Miklos Santha*
CNRS, URA 410
Université Paris-Sud, LRI
91405 Orsay, France

Abstract

In the boolean decision tree model there is at least a linear gap between the Monte Carlo and the Las Vegas complexity of a function depending on the error probability. We prove for a large class of read-once formulae that this trivial speed-up is the best what a Monte Carlo algorithm can achieve. For every formula F belonging to that class we show that the Monte Carlo complexity of F with two sided error p is $(1-2p)R(F)$, and with one sided error p is $(1-p)R(F)$, where $R(F)$ denotes the Las Vegas complexity of F. The result follows from a general lower bound that we derive on the Monte Carlo complexity of these formulae. This bound is analogous to the lower bound due to Saks and Wigderson on their Las Vegas complexity.

Key words: boolean decision tree, randomized algorithm, read-once formula, lower bound.

1 Introduction

The boolean decision tree is an extremely simple model for computing boolean functions. It computes a function by repeatedly checking input bits until the value of the function can be determined. The cost associated with this computation is only the number of bits checked, all the other computations are free. Formally, a *deterministic decision tree algorithm* A on n variables is a binary tree whose internal nodes are labeled by some input variable x_i, $1 \leq i \leq n$. The leaves of the tree are labeled by the output values 0 or 1, and for every internal node,

*This research was supported by an Alexander von Humboldt Fellowship while the author visited the Max Planck Institut für Informatik, D-6600 Saarbrücken, and by the ESPRIT Working Group 7097 RAND.

one of the outgoing edge is labeled by the value 0, and the other by the value 1. For every input $w = w_1 \ldots w_n$, there is a unique root-leaf path in the tree which follows at every node, labeled by x_i, the edge out of the node corresponding to the value w_i. The value of the algorithm A on input w, which we denote by $A(w)$, is the output value of this root-leaf path. The algorithm A *computes* a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ if for every input w, we have $A(w) = f(w)$.

We define the *cost* $C(A, w)$ of a deterministic decision tree algorithm A on input w to be the number of input bits checked by A on w. Let $\mathcal{A}(f)$ be the set of all deterministic decision tree algorithms which compute f. The *deterministic complexity* of f is

$$D(f) = \min_{A \in \mathcal{A}(f)} \max_{w \in \{0,1\}^n} C(A, w).$$

Clearly, for every function f on n variables, $D(f) \leq n$. The first important result about the deterministic decision tree complexity was a linear lower bound for monotone graph properties obtained by Rivest and Vuillemen [8]. In another interesting development Nisan [7] proved that the CREW PRAM complexity of any function f is $\Theta(\log D(f))$.

As usual, we can also permit randomization in this computational model. A *randomized decision tree algorithm* P on n variables is a distribution over all deterministic decision tree algorithms on n variables. The value of the randomized algorithm P on w and the cost of P on w are random variables. The *expected cost* of the algorithm P on input w, which we denote also by $C(P, w)$, is the expected number of input bits checked by P on w. With respect to a function f, the algorithm P *errs* with probability p on w if $Pr[P(w) \neq f(w)] = p$. The error probability of P on w will be denoted by $E(P, w)$. We say that P *computes* f with *two sided error* p if for every input w, the algorithm errs with probability at most p. For $0 \leq p \leq 1/2$ let $\mathcal{R}_2^p(f)$ be the set of all randomized decision tree algorithms which compute f with two sided error p. We say that P *computes* f with *one sided error* p if for every w such that $f(w) = 0$, the algorithm is always correct; and for every w such that $f(w) = 1$, the algorithm errs with probability at most p. For $0 \leq p \leq 1$ let $\mathcal{R}_1^p(f)$ be the set of all randomized decision tree algorithms which compute f with one sided error p. The *randomized complexity* of f with *two sided error* p is

$$R_2^p(f) = \inf_{P \in \mathcal{R}_2^p(f)} \max_{w \in \{0,1\}^n} C(P, w),$$

and with *one sided error* p is

$$R_1^p(f) = \inf_{P \in \mathcal{R}_1^p(f)} \max_{w \in \{0,1\}^n} C(P, w).$$

Finally an important special case of these two complexities is the *zero error randomized complexity* of f which is defined as

$$R(f) = \inf_{P \in \mathcal{R}_2^0(f)} \max_{w \in \{0,1\}^n} C(P, w).$$

The one and two sided randomized complexities of f are also called *Monte Carlo* complexity, and the zero error randomized complexity is called *Las Vegas* complexity.

For every p, we have the following obvious relationships between these complexity measures:

$$R_2^p(f) \leq R_1^p(f) \leq R(f) \leq D(f).$$

Blum and Impagliazzo [1], Hartmanis and Hemachandra [3], and Tardos [11] observed independently that there is at most a quadratic gap between the deterministic complexity and the Las Vegas complexity. More precisely, for every f we have

$$D(f) \leq (R(f))^2.$$

Nisan [7] showed that the deterministic complexity can not exceed more than polynomially the Monte Carlo complexity either. He proved that for for every function f, and for every constant $0 < p < 1/2$ we have

$$D(f) = O((R_2^p(f))^3),$$

and for every constant $0 < p < 1$ we have

$$D(f) = O((R_1^p(f))^2).$$

Several results were obtained about the Las Vegas complexity of specific functions, in particular monotone graph properties and read-once formulae. Subsequently Yao [13], King [6], and Hajnal [2] proved non trivial lower bounds for monotone graph properties, the best known result $\Omega(n^{2/3})$ is due to Hajnal. The general conjecture is that their Las Vegas complexity is actually $\Theta(n)$.

A *read-once formula* on n variables is a binary tree with n leaves, whose internal nodes are labeled with AND or OR, and whose leaves are labeled with distinct variables. Every read-once formula F computes in a natural way a boolean function on n variables, which is also denoted by F. Saks and Wigderson [9] have derived sharp upper and lower bounds on the Las Vegas complexity of several read-once formulae. For a large class of read-once formulae, which we call *well balanced* and will define precisely in the next section, these bounds are matching, and consequently equal to the exact Las Vegas complexity. In particular they have shown that the zero error randomized complexity of the uniform alternating read-once formula is $\Theta(n^\alpha)$, where $\alpha = \log_2(\frac{1+\sqrt{33}}{4}) \approx 0.753$. The $O(n^\alpha)$ upper bound for this particular function was previously obtained by Snir [10]. Saks and Wigderson also conjectured that actually this is the largest possible gap between the deterministic and the Las Vegas complexity, that is for every f,

$$R(f) = \Omega((D(f)^\alpha).$$

This conjecture is still open.

In a read-once *threshold* formula threshold functions with arbitrary fan-in are also permitted in the internal nodes of the tree. Heiman, Newman and

Wigderson [4] proved a lower bound of $n/2^d$ for the Las Vegas complexity of any read-once threshold formula of depth d. In another recent development Heiman and Wigderson [5] established a non-trivial lower bound for read-once formulae without any balancing condition. They proved that the Las Vegas complexity of every read-once formula is at least $n^{0.51}$.

In this paper we examine the Monte Carlo complexity of read-once formulae versus their Las Vegas complexity. It is easy to see that the Monte Carlo complexity of every non-trivial function is smaller than its Las Vegas complexity since a Monte Carlo algorithm can output within the limits of its error probability a (maybe erroneous) value without any computation. However, this gives only a gain of a constant factor over the Las Vegas complexity. We will show that for well balanced read-once formulae this trivial speed-up is the best what a Monte Carlo algorithm can achieve. We will derive a general lower bound on the Monte Carlo complexity of read-once formulae which is analogous to the lower bound obtained by Saks and Wigderson on their Las Vegas complexity. In the case of well balanced formulae this lower bound will match the upper bound which can be obtained by combining the general linear speed-up with the upper bound of Saks and Wigderson on their Las Vegas complexity. More precisely, we get the following results:

Theorem *For every well balanced read-once formula F, and for every $0 \leq p \leq 1/2$ we have*

$$R_2^p(F) = (1-2p)R(F),$$

and for every $0 \leq p \leq 1$ we have

$$R_1^p(F) = (1-p)R(F).$$

The outline of our proof is that of Saks and Wigderson. The lower bound relies in both cases on a so called Shrinking Lemma. In both cases the lemma establishes an inequality between the randomized complexities of two read once formulae F and F' where F' is obtained from F by truncation. In their proof Saks and Wigderson used Yao's duality result [12] about the Las Vegas and the distributional complexity of a function.

Let $\mathcal{D}_n$ be the set of distributions over $\{0,1\}^n$. The *average cost* $C(A,d)$ of a deterministic decision tree algorithm A on n variables with respect to a distribution $d \in \mathcal{D}_n$ is defined by

$$C(A,d) = \sum_{w \in \{0,1\}^n} d(w)C(A,w).$$

The *distributional complexity* of a function f is

$$\Delta(f) = \sup_{d \in \mathcal{D}_n} \min_{A \in \mathcal{A}(f)} C(A,d).$$

Yao's duality theorem states that the Las Vegas and the distributional complexities of f are equal, that is

$$R(f) = \Delta(f).$$

Using this theorm Saks and Wigderson proved the Shrinking Lemma for the Las Vegas complexity by showing the corresponding inequality between the distributional complexities of the formulae in the "dual world" of input distributions and deterministic algorithms.

We can not extend their proof in the dual world for the Monte Carlo complexity since there is no known duality theorem for Monte Carlo algorithms. Indeed, Yao proved only an inequality between the Monte Carlo complexity and the distributional complexity with error. Let d be a distribution over $\{0,1\}^n$, and let $0 \leq p \leq 1$. We say that a deterministic decision tree algorithm A *computes* a function f with *error* p *with respect* to d if $\sum_{A(w) \neq f(w)} d(w) = p$. Let $\mathcal{A}_d^p(f)$ denote the set of algorithms which compute f with error at most p with respect to d. The *distributional complexity* of f with *error* p is

$$\Delta^p(f) = \sup_{d \in \mathcal{D}_n} \min_{A \in \mathcal{A}_d^p(f)} C(A, d).$$

For every $0 \leq p \leq 1/2$, Yao established the inequality

$$R_2^p(f) \geq \Delta^{2p}(f)/2.$$

It is still open if there exists a similar inequality in the opposite direction. The lack of a duality result for the complexity with error prevents us from working in the dual world: an inequality between the distributional complexities with error of the two formulae does not necessarily translates into an inequality between their Monte Carlo complexities. Therefore we prove the extended form of the Shrinking Lemma for Monte Carlo complexity in the "primal world" of randomized algorithms and deterministic inputs. The extended lemma contains the original one as a special case.

In Section 2 we review some definitions and results of Saks and Wigderson which we will use later. Section 3 contains our upper bounds and the reduction of our lower bounds to the Shrinking Lemma which is proven in Section 4. We mention an open problem in Section 5.

2 Previous results on the Las Vegas complexity

Let F be a read-once formula. The left subtree of F will be denoted by F_L, and its right subtree by F_R. A read-once formula is *alternating* if the labels AND and OR are assigned to alternating levels of the tree. Read-once formulae are by definition monotone. This does not cause loss of generality for our purposes since negations can be pushed down to the inputs, and renaming negative literals as positive ones does not change the decision tree complexities. A simple adversary argument shows that every deterministic boolean decision tree in the worst case evaluates all the variables of a read-once formula.

Fact 1 *For every read-once formula F on n variables we have:*

$$D(F) = n.$$

Saks and Wigderson [9] have derived upper and lower bounds on the Las Vegas complexity of read-once formulae. For several important classes of formulae the upper and lower bounds are actually matching and therefore their common value is the exact Las Vegas complexity. To state their result it is useful to define a function ψ on four variables:

$$\psi(a_0, b_0, a_1, b_1) = \frac{a_0 a_1 + b_0 b_1 + a_1 b_1}{a_1 + b_1}.$$

Definition: For every read-once formula F let $u(F) = \max\{u_0(F), u_1(F)\}$, where the functions u_0 and u_1 are defined inductively. If F contains a single node then $u_0(F) = u_1(F) = 1$. Otherwise

$$\begin{array}{rcl} u_1(F) & = & u_1(F_L) + u_1(F_R), \\ u_0(F) & = & \max\{u_0(F_L), u_0(F_R), \\ & & \psi(u_0(F_L), u_0(F_R), u_1(F_L), u_1(F_R))\} \end{array}$$

if the root of F is labeled by AND, and

$$\begin{array}{rcl} u_0(F) & = & u_0(F_L) + u_0(F_R), \\ u_1(F) & = & \max\{u_1(F_L), u_1(F_R), \\ & & \psi(u_1(F_L), u_1(F_R), u_0(F_L), u_0(F_R))\} \end{array}$$

if the root of F is labeled by OR.

Definition: For every read-once formula F let $l(F) = \max\{l_0(F), l_1(F)\}$, where the functions l_0 and l_1 are defined inductively. If F contains a single node then $l_0(F) = l_1(F) = 1$. Otherwise

$$\begin{array}{rcl} l_1(F) & = & l_1(F_L) + l_1(F_R), \\ l_0(F) & = & \min\{l_0(F_L) + l_1(F_R), l_0(F_R) + l_1(F_L), \\ & & \psi(l_0(F_L), l_0(F_R), l_1(F_L), l_1(F_R))\} \end{array}$$

if the root of F is labeled by AND, and

$$\begin{array}{rcl} l_0(F) & = & l_0(F_L) + l_0(F_R), \\ l_1(F) & = & \min\{l_1(F_L) + l_0(F_R), l_1(F_R) + l_0(F_L), \\ & & \psi(l_1(F_L), l_1(F_R), l_0(F_L), l_0(F_R))\} \end{array}$$

if the root of F is labeled by OR.

Saks and Wigderson proved that the expected cost of the best Las Vegas algorithm for F is at most $u_0(F)$ on every w with $F(w) = 0$, and similarly its expected cost is at most $u_1(F)$ on every w with $F(w) = 1$. It follows that the Las Vegas complexity is at most $u(F)$. The major result of their paper is that a similar lower bound result holds with the function $l(F)$.

Theorem 1 (Saks and Wigderson) *For every read-once formula F we have*

$$l(F) \le R(F) \le u(F).$$

Definition: A read-once formula F is called *well balanced* if $u(F) = l(F)$.

Corollary 1 *For every well balanced read-once formula F the common value $u(F) = l(F)$ is the exact Las Vegas complexity of F.*

Saks and Wigderson observed that several alternating read-once formulae are well balanced. These formulae include uniform trees, in general trees where the difference between the depths of any two leaves is at most one, Fibonacci trees and binomial trees.

3 Monte Carlo complexity

First we show that a Las Vegas algorithm can always be transformed into a Monte Carlo algorithm with slightly better expected cost. Therefore the Monte Carlo complexity of a function is always somewhat smaller than its Las Vegas complexity.

Theorem 2 *For every function f, and for every $0 \le p \le 1/2$ we have*

$$R_2^p(f) \le (1-2p)R(f),$$

and for every $0 \le p \le 1$ we have

$$R_1^p(f) \le (1-p)R(f).$$

Proof: Let P_f be the optimal Las Vegas algorithm for f. By definition, let P_2^p be the randomized algorithm which outputs 0 with probability p, outputs 1 with probability p, and simulates P_f with probability $1-2p$. Similarly, let P_1^p be the algorithm which outputs 0 with probability p, and simulates P_f. with probability $1-p$. Clearly, the maximum error probability of both algorithms is bounded by p and P_1^p never errs on inputs with $f(w) = 0$. Moreover, for every input w we have

$$C(P_2^p, w) \le (1-2p)C(P_f, w)$$

and

$$C(P_1^p, w) \leq (1-p)C(P_f, w),$$

and the theorem follows. $\square$

In the definitions of the different complexities of a boolean function we assumed a unit cost for checking an input bit. To state the Shrinking Lemma, these notions shall be generalized to the case where the cost of checking an input bit is a function of the variable and its value. A *cost function pair* for n variables is a pair of functions (c_0, c_1) such that for $j = 0, 1$ we have $c_j : \{x_1, \ldots, x_n\} \rightarrow R^*$, where R^* denotes the set of non-negative reals. The cost of checking the variable x_i on input w with respect to the pair (c_0, c_1) is $c_j(x_i)$ if $w_i = j$, for $i = 1, \ldots, n$ and $j = 0, 1$. The expected cost of a randomized algorithm P on input w with respect to the pair (c_0, c_1), which we denote $C(P, w, c_0, c_1)$, and the randomized complexity of a function f with respect to (c_0, c_1) with two sided error p (respectively one sided error p), which we denote $R_2^p(f, c_0, c_1)$ (respectively $R_1^p(f, c_0, c_1)$), is the natural generalization of $C(P, w)$ and $R_2^p(f)$ (respectively $R_1^p(f)$).

We will now state the Shrinking Lemma. Let F be a read-once formula on n variables, and let (c_0, c_1) be a cost function pair for n variables. Let V be a node of F whose children are leaves, and suppose that these children are labeled by the variables y and z. Let F' be the read-once formula on $n-1$ variables obtained from F by deleting the children of V from F and labeling V with a new variable v. We define a cost function pair (c_0', c_1') for the $n-1$ variables of F' as follows: $c_0'(x) = c_0(x)$ and $c_1'(x) = c_1(x)$ if $x \neq v$. Otherwise

$$\begin{aligned} c_1'(v) &= c_1(y) + c_1(z), \\ c_0'(v) &= \min\{c_0(y) + c_1(z), c_0(z) + c_1(y), \\ &\quad \psi(c_0(y), c_0(z), c_1(y), c_1(z))\} \end{aligned}$$

if the node V is labeled by AND, and

$$\begin{aligned} c_0'(v) &= c_0(y) + c_0(z), \\ c_1'(v) &= \min\{c_1(y) + c_0(z), c_1(z) + c_0(y), \\ &\quad \psi(c_1(y), c_1(z), c_0(y), c_0(z))\} \end{aligned}$$

if the node V is labeled by OR.

Shrinking Lemma *Using the above notation, for every* $0 \leq p \leq 1/2$ *we have*

$$R_2^p(F, c_0, c_1) \geq R_2^p(F', c_0', c_1'),$$

and for every $0 \leq p \leq 1$ *we have*

$$R_1^p(F, c_0, c_1) \geq R_1^p(F', c_0', c_1').$$

The Shrinking Lemma will be proved in the next section. Our lower bounds follow easily from it.

Theorem 3 *For every read-once formula F, and for every $0 \leq p \leq 1/2$ we have*

$$R_2^p(F) \geq (1-2p)l(F),$$

and for every $0 \leq p \leq 1$ we have

$$R_1^p(F) \geq (1-p)l(F).$$

Proof : We apply the Shrinking Lemma inductively to F in an arbitrary way, until it shrinks to a single variable. It is easy to see that for every node V of F, when V becomes a leaf in this shrinking procedure, the respective costs for checking the variable corresponding to V will be $l_0(F_V)$ and $l_1(F_V)$, where F_V is the subformula of F with root V. Therefore the costs of checking the remaining single variable are $l_0(F)$ and $l_1(F)$. The expected cost in the worst case of any randomized algorithm with error p for computing a single variable with these costs is clearly $(1-2p)l(F)$ if the error can be two sided, and $(1-p)l(F)$ if the error is one sided. $\square$

Corollary 2 *For every well balanced read-once formula F, and for every $0 \leq p \leq 1/2$ we have*

$$R_2^p(F) = (1-2p)R(F),$$

and for every $0 \leq p \leq 1$ we have

$$R_1^p(F) = (1-p)R(F).$$

Proof : Theorems 1, 2 and 3. $\square$

4 The proof of the Shrinking Lemma

We will prove the Shrinking Lemma for two sided error, the one sided case is analogous. The proof follows by definition from the following claim:

Claim: *For every randomized algorithm P computing F, there exists a randomized algorithm P' computing F' such that for every input w' to F', the following two conditions hold:*
there exists an input w to F such that

$$C(P, w, c_0, c_1) \geq C(P', w', c_0', c_1'), \tag{1}$$

and there exists an input w to F such that

$$E(P, w) \geq E(P', w'). \tag{2}$$

Proof: We assume that V is labeled in F by AND, the other case is similar. Let P be a randomized algorithm computing F. Set the probabilities p_y and p_z as

$$p_y = \frac{c_1(y)}{c_1(y)+c_1(z)}, \quad p_z = \frac{c_1(z)}{c_1(y)+c_1(z)}.$$

Let us call the variables y and z *interesting.* We define the algorithm P' for F' as follows: P' simulates P as long as it checks only non interesting variables. In every occasion when for the first time P checks an interesting variable, P' makes a probabilistic choice about how to continue the computation of F'; these choices are made independently from each other. We call such a choice *interesting*, and also *y-choice* (respectively *z-choice*) when the variable checked is y (z). We will describe the y-choice, the z-choice can be obtained by exchanging the role of y and z in the description.

Description of the y-choice: P' choses with probability p_y the 1st possibility and with probability p_z the 2nd possibility where these possibilities are defined as follows.

- 1st possibility: P' checks v and continues the simulation of P by substituting the value of v for y. If P later checks z then P' continues the simulation according to $z = 1$ (that is branching out on the edge corresponding to $z = 1$);
- 2nd possibility: P' continues the simulation of P according to $y = 1$. If P later checks z then P' replaces this by checking v, and continues the simulation by substituting the value of v for y.

We will prove that for every input w' to F' the conditions of the Claim hold. This will be done in two cases, according to the value of the variable v in w'. The following two lemmas correspond to these cases. To simplify the notation we will not mark the cost function pair in the expressions for the expected costs. We also use the convention that concerning the computations of P on w the superscripts correspond to the value of y and the subscripts to the value of z.

Lemma 1 *Let w'_1 be an input for F' in which $v = 1$. We define w^1_1 to be the input for F in which $y = z = 1$ and the values of all the non interesting variables are the same as in w'_1. Then we have*

$$C(P, w^1_1) = C(P', w'_1), \tag{3}$$

and

$$E(P, w) = E(P', w'). \tag{4}$$

Proof: First we prove that the equality about the expected costs is satisfied. We need a few notations to express them. Let K be the expected cost of the algorithm P on w_1^1 before it checks any interesting variable. This is also the expected cost of P' on w_1' before P' makes an interesting choice. Let q_y (respectively q_z) denote the probability that P on w_1^1 checks an interesting variable and the first one is y (respectively z.) This is also the probability that P' on w_1' makes a y-choice (z-choice). Let K^1 (K_1) be the expected cost of P on w_1^1 after it checked the first interesting variable under the condition that this variable is y (z), not including the cost of the (eventual) checking of the second interesting variable. This is also the expected cost of P' on w_1' after the interesting choice was made under the condition that it was a y-choice (z-choice), not including the cost of checking of v. Finally let r^1 (r_1) be the probability that P on w_1^1 checks the second interesting variable under the condition that the first interesting variable checked is y (z). This is also the probability that P' on w_1' checks v under the condition that it made a y-choice and it chose the second possibility. With these notations, the expected cost of P on w_1^1 is:

$$\begin{aligned} C(P, w_1^1) \;\; = \;\; & K + q_y\{c_1(y) + K^1 + r^1 c_1(z)\} + \\ & q_z\{c_1(z) + K_1 + r_1 c_1(y)\}. \end{aligned}$$

This expression corresponds to the expected cost before checking the first interesting variable, to the cost of checking this variable, and to the cost after that. In the latest term the cost of checking the second interesting variable is expressed separately. The expected cost of P' on w_1' is:

$$\begin{aligned} C(P', w_1') \;\; = \;\; & K + \\ & q_y\{p_y(c_1'(v)+K^1)+p_z(K^1+r^1 c_1'(v))\}+ \\ & q_z\{p_z(c_1'(v)+K_1)+p_y(K_1+r_1 c_1'(v))\}. \end{aligned}$$

Using the definitions of p_y, p_z and $c_1'(v)$ we get that equation 3 is satisfied.

We now turn to the equality concerning the error probabilities. Let e be the probability that P on w_1^1 doesn't check an interesting variable and errs. This is also the probability that P' on w_1' does not make an interesting choice and errs. Let e^1 (e_1) be the probability that P errs on w_1^1 under the condition that P checks an interesting variable and the first one is y (z). This is the same probability that P' errs on w_1' under the condition that makes a y-choice (z-choice). Then we have

$$E(P, w_1^1) = e + q_y e^1 + q_z e_1,$$

and

$$E(P', w_1') = e + q_y(p_y e^1 + p_z e^1) + q_z(p_z e_1 + p_y e_1)$$

These two expressions are trivially equal. $\square$

Lemma 2 *Let w_0' be an input for F' in which $v = 0$. We define w_1^0 to be the input for F in which $y = 0$ and $z = 1$, and the values of all the non interesting* *variables are the same as in w'_0. Similarly, in w^1_0 let $y = 1$ and $z = 0$, and let all the other bits be as in w'_0. Then we have*

$$p_y C(P, w^0_1) + p_z C(P, w^1_0) \geq C(P', w'_0), \tag{5}$$

and

$$p_y E(P, w^0_1) + p_z E(P, w^1_0) = E(P', w'_0). \tag{6}$$

Proof: We introduce a few notations as previously. The expected cost of the algorithm P before checking an interesting variable on w^0_1 is the same as on w^1_0, let K denote this quantity. Let q_y (q_z) denote the probability that P on w^0_1 (or independently on w^1_0) checks an interesting variable and the first interesting variable checked is y $(z.)$ Let K^0 (K_1) be the expected cost of P on w^0_1 after checking the first interesting variable under the condition that this variable is y (z), not including the cost of the (eventual) checking of the second interesting variable. Similarly let K^1 (K_0) be the expected cost of P on w^1_0 after checking the first interesting variable under the condition that this variable is y (z), not including the cost of the (eventual) checking of the second interesting variable. Finally let r^1 (r_1) be the probability that P on w^1_0 (w^0_1) checks z (y) under the condition that the first interesting variable checked is y (z). To all these quantities, as in the Lemma 1, naturally correspond equal quantities in the computation of P' on w'_0.

We will bound from below $C(P, w^0_1)$ and $C(P, w^1_0)$ by not counting the cost of checking the second interesting variable in the case when the first interesting variable checked has value 0. Otherwise we count like in Lemma 1. We have then

$$\begin{aligned} C(P, w^0_1) \quad &\geq \quad K + q_y\{c_0(y) + K^0\} + \\ &\qquad q_z\{c_1(z) + K_1 + r_1 c_0(y)\}, \end{aligned}$$

and

$$\begin{aligned} C(P, w^1_0) \quad &\geq \quad K + q_z\{c_0(z) + K_0\} + \\ &\qquad q_y\{c_1(y) + K^1 + r^1 c_0(z)\}. \end{aligned}$$

The expected cost of P' on w'_0 is:

$$\begin{aligned} C(P', w'_0) \;&=\; K + \\ &\quad q_y\{p_y(c'_0(v)+K^0)+p_z(K^1+r^1 c'_0(v))\}+ \\ &\quad q_z\{p_z(c'_0(v)+K_0)+p_y(K_1+r_1 c'_0(v))\}. \end{aligned}$$

We will show that substituting these lower bounds for $C(P, w^0_1)$ and $C(P, w^1_0)$, and the expression for $C(P', w'_0)$, inequality 5 is satisfied. All the terms involving the letter K obviously cancel out. We show that on the left hand side the sum of the terms which are multiplied by q_y is at least as big as those on the right hand side, and that a similar result holds for q_z. After simplification this reduces to the following inequality:

$$p_y c_0(y) + p_z(c_1(y) + r^1 c_0(z)) - (p_y + p_z r^1)c'_0(v) \geq 0. \tag{7}$$

Since the left hand side of inequality 7 is a linear function of r^1, it is sufficient to check it for $r^1 = 0$ and $r^1 = 1$. For $r^1 = 0$, it reduces to

$$c'_0(v) \leq c_0(y) + c_1(z), \tag{8}$$

and for $r^1 = 1$ it reduces to

$$c'_0(v) \leq \psi(c_0(y), c_0(z), c_1(y), c_1(z)). \tag{9}$$

When we are looking at terms multiplied by q_z in both sides of inequality 5, we also need

$$c'_0(v) \leq c_0(z) + c_1(y). \tag{10}$$

The inequalities 8, 9 and 10 are satisfied by the definition of $c'_0(v)$.

Finally we prove the equality about the error probabilities. The probability that P on w_1^0 doesn't check an interesting variable and errs is the same as the probability that P on w_0^1 doesn't check an interesting variable and errs. Let e be that probability. Let e^0 (e_1) be the probability that P errs on w_1^0 under the condition that P checks an interesting variable and the first one is y (z). Similarly let e^1 (e_0) be the probability that P errs on w_0^1 under the condition that P checks an interesting variable and the first one is y (z). Then we have

$$E(P, w_1^0) = e + q_y e^0 + q_z e_1,$$

$$E(P, w_0^1) = e + q_y e^1 + q_z e_0,$$

and

$$E(P', w'_0) = e + q_y(p_y e^0 + p_z e^1) + q_z(p_z e_0 + p_y e_1).$$

These expressions clearly satisfy equality 6. $\square$

5 Open problem

In this paper we have shown that for some class of boolean functions, no Monte Carlo algorithm can achieve a greater than linear speed-up over the best Las Vegas algorithm. The obvious open question concerns the exact relationship in general between Monte Carlo and Las Vegas complexities. In particular, is it possible to find at all a function which has a superliner gap between these two measures?

Acknowledgments

We are thankful to Rudolf Fleischer and Hermann Jung for several helpful conversations.

References

[1] M. Blum and R. Impagliazzo (1987), Generic oracles and oracle classes, *Proceedings of 18th IEEE FOCS*, 118-126.

[2] P. Hajnal (1990), On the power of randomness in the decision tree model, *Proceedings of 5th Structure in Complexity Theory*, 66-77.

[3] J. Hartmanis and L. Hemachandra (1987), One-way functions, robustness, and non-isomorphism of NP-complete sets, *Proceedings of 2nd Structure in Complexity Theory*, 160-173.

[4] R. Heiman, I. Newman and A. Wigderson (1990), On read-once threshold formulae and their randomized decision tree complexity, *Proceedings of 5th Structure in Complexity Theory*, 78-87.

[5] R. Heiman and A. Wigderson (1991), Randomized versus deterministic decision tree complexity for read-once Boolean functions, *Proceedings of 6th IEEE Structure in Complexity Theory*, 170-179.

[6] V. King (1988), Lower bound on the complexity of graph properties, *Proceedings of 20th ACM STOC*, 468-476.

[7] N. Nisan (1989), CREW PRAMs and decision trees, *Proceedings of 21st ACM STOC*, 327-335.

[8] R. Rivest and J. Vuillemen (1978), On recognizing graph properties from adjacency matrices, *Theoretical Computer Science* 3, 371-384.

[9] M. Saks and A. Wigderson (1986), Probabilistic boolean decision trees and the complexity of evaluating game trees, *Proceedings of 27th IEEE FOCS*, 29-38.

[10] M. Snir (1985), Lower bounds for probabilistic linear decision trees, *Theoretical Computer Science* 38, 69-82.

[11] G. Tardos (1989), Query complexity or why is it difficult to separate $NP^A \bigcap co - NP^A$ from P^A by a random oracle A, *Combinatorica* 9 (4), 385-392.

[12] A. Yao (1977), Probabilistic computations: towards a unified measure of complexity, *Proceedings of 18th IEEE FOCS*, 393-400.

[13] A. Yao (1987), Lower bounds to randomized algorithms for graph properties, *Proceedings of 28th IEEE FOCS*, 393-400.