Improved bounds for the randomized decision tree complexity of recursive majority[⋆]

Frédéric Magniez[1], Ashwin Nayak[2⋆⋆], Miklos Santha[1,3⋆⋆⋆], and David Xiao[1,4]

[1] LIAFA, Univ. Paris 7, CNRS; Paris, France. magniez@liafa.jussieu.fr
[2] C&O and IQC, U. Waterloo; and Perimeter Institute; Waterloo, ON, Canada. ashwin.nayak@uwaterloo.ca
[3] Centre for Quantum Technologies, National U. of Singapore. santha@lri.fr
[4] Univ. Paris-Sud; Orsay, France. dxiao@lri.fr

Abstract. We consider the randomized decision tree complexity of the recursive 3-majority function. For evaluating height h formulae, we prove a lower bound for the δ-two-sided-error randomized decision tree complexity of $(1-2\delta)(5/2)^h$, improving the lower bound of $(1-2\delta)(7/3)^h$ given by Jayram, Kumar, and Sivakumar (STOC '03). Second, we improve the upper bound by giving a new zero-error randomized decision tree algorithm that has complexity at most $(1.007) \cdot 2.64946^h$. The previous best known algorithm achieved complexity $(1.004) \cdot 2.65622^h$. The new lower bound follows from a better analysis of the base case of the recursion of Jayram *et al.* The new algorithm uses a novel "interleaving" of two recursive algorithms.

1 Introduction

Decision trees form a simple model for computing boolean functions by successively reading the input bits until the value of the function can be determined. In this model, the only cost is the number of input bits queried. Formally, a *deterministic decision tree algorithm* A on n variables is a binary tree in which each internal node is labeled with an input variable x_i, and the leaves of the tree are labeled by either 0 or 1. Each internal node has two outgoing edges, labeled either by 0 or 1. For every input $x = x_1 \dots x_n$, there is a unique path in the tree leading from the root to a leaf: if an internal node is labeled by x_i, we follow either the 0 or the 1 outgoing edge according to the value of x_i. The value of the algorithm A on input x, denoted by $A(x)$, is the label of the leaf on this unique path. The algorithm A *computes* a boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every input x, we have $A(x) = f(x)$.

[⋆] Partially supported by the French ANR Defis project ANR-08-EMER-012 (QRAC) and the European Commission IST STREP project 25596 (QSC).

[⋆⋆] Partially supported by NSERC Canada. Research at PI is supported by the Gvt of Canada through Industry Canada and by the Province of Ontario through MRI.

[⋆⋆⋆] Research at the Centre for Quantum Technologies is funded by the Singapore Ministry of Education and the National Research Foundation.

We define the *cost* $\mathrm{C}(A,x)$ of a deterministic decision tree algorithm A on input x as the number of input bits queried by A on x. Let $\mathcal{P}_f$ be the set of all deterministic decision tree algorithms which compute f. The *deterministic complexity* of f is $\mathrm{D}(f) = \min_{A\in\mathcal{P}_f} \max_{x\in\{0,1\}^n} \mathrm{C}(A,x)$. Since every function can be evaluated after reading all the input variables, $\mathrm{D}(f) \leq n$. In an extension of the deterministic model, we can also permit randomization in the computation.

A *randomized decision tree algorithm* A on n variables is a distribution over all deterministic decision tree algorithms on n variables. Given an input x, the algorithm first samples a deterministic tree $B \in_{\mathrm{R}} A$, then evaluates $B(x)$. The error probability of A in computing f is given by $\max_{x\in\{0,1\}^n} \Pr_{B\in_{\mathrm{R}} A}[B(x) \neq f(x)]$. The *cost* of a randomized algorithm A on input x, denoted also by $\mathrm{C}(A,x)$, is the expected number of input bits queried by A on x. Let $\mathcal{P}_f^\delta$ be the set of randomized decision tree algorithms computing f with error at most δ. The two-sided bounded error *randomized complexity* of f with error $\delta \in [0,1/2)$ is $\mathrm{R}_\delta(f) = \min_{A\in\mathcal{P}_f^\delta} \max_{x\in\{0,1\}^n} \mathrm{C}(A,x)$.

We write $\mathrm{R}(f)$ for $\mathrm{R}_0(f)$. By definition, for all $0 \leq \delta < 1/2$, it holds that $\mathrm{R}_\delta(f) \leq \mathrm{R}(f) \leq \mathrm{D}(f)$, and it is also known [1, 2, 12] that $\mathrm{D}(f) \leq \mathrm{R}(f)^2$, and that for all constant $\delta \in (0,1/2)$, $\mathrm{D}(f) \in \mathrm{O}(\mathrm{R}_\delta(f)^3)$ [7].

Considerable attention in the literature has been given to the randomized complexity of functions computable by read-once formulae, which are boolean formulae in which every input variable appears only once. For a large class of well balanced formulae with NAND gates the exact randomized complexity is known. In particular, let NAND_h denote the *complete* binary tree of height h with NAND gates, where the inputs are at the $n = 2^h$ leaves. Snir [11] has shown that $\mathrm{R}(\mathsf{NAND}_h) \in \mathrm{O}(n^c)$ where $c = \log_2\left(\frac{1+\sqrt{33}}{4}\right) \approx 0.753$. A matching $\Omega(n^c)$ lower bound was obtained by Saks and Wigderson [9], and extended to Monte-Carlo algorithms by Santha [10]. Since $\mathrm{D}(\mathsf{NAND}_h) = 2^h = n$ this implies that $\mathrm{R}(\mathsf{NAND}_h) \in \Theta(\mathrm{D}(\mathsf{NAND}_h)^c)$. Saks and Wigderson conjectured that *for every* boolean function f and constant $\delta \in [0,1/2)$, $\mathrm{R}_\delta(f) \in \Omega(\mathrm{D}(f)^c)$.

After further progress due to Heiman, Newman, and Wigderson [3] and Heiman and Wigderson [4], one would have hoped that the simple model of decision tree algorithms might shed more light on the power of randomness. But surprisingly, we know the exact randomized complexity of very few boolean functions. In particular, the randomized complexity of the recursive 3-majority function ($\mathsf{3\text{-}MAJ}_h$) is still open. This function, proposed by Boppana, was one of the earliest examples where randomized algorithms were found to be more powerful than deterministic decision trees [9]. It is a read-once formula on 3^h variables given by the complete ternary tree of height h whose internal vertices are majority gates. It is easy to check that $\mathrm{D}(\mathsf{3\text{-}MAJ}_h) = 3^h$, but there is a naive randomized recursive algorithm for $\mathsf{3\text{-}MAJ}_h$ that performs better: pick two random children of the root and recursively evaluate them, then evaluate the third child iff the value is not yet determined. This has zero-error randomized complexity $(8/3)^h$. However, it was already observed by Saks and Wigderson [9] that one can do even better than this naive algorithm. As for lower bounds, that reading 2^h variables is necessary for zero-error algorithms is easy to show. In spite of some similarities with the NAND_h function, no progress was reported on the randomized complexity of 3-MAJ for 17 years. In 2003, Jayram, Kumar, and Sivakumar [5] proposed an explicit randomized algorithm that achieves complexity $(1.004) \cdot 2.65622^h$, and beats the naive recursion. (Note, however, that the recurrence they derive in [5, Appendix B] is incorrect.) They also prove a $(1-2\delta)(7/3)^h$ lower bound for the δ-error randomized decision tree complexity of 3-MAJ_h. In doing so, they introduce a powerful combinatorial technique for proving decision tree lower bounds.

In this paper, we considerably improve the lower bound obtained in [5], by proving that $\mathrm{R}_\delta(\text{3-}\mathsf{MAJ}_h) \geq (1-2\delta)(5/2)^h$. We also improve the upper bound by giving a new zero-error randomized decision tree algorithm that has complexity at most $(1.007)2.64946^h$.

Theorem 1. *For all $\delta \in [0, 1/2]$, we have $(1-2\delta)(5/2)^h \leq \mathrm{R}_\delta(\text{3-}\mathsf{MAJ}_h) \leq (1.007)2.64946^h$.*

In contrast to the randomized case, the bounded-error *quantum* query complexity of 3-MAJ_h is known more precisely; it is in $\Theta(2^h)$ [8].

New lower bound. For the lower bound they give, Jayram *et al.* consider a complexity measure related to the distributional complexity of 3-MAJ_h with respect to a specific hard distribution (cf. Sect. 2.3). The focus of the proof is a relationship between the complexity of evaluating formulae of height h to that of evaluating formulae of height $h-1$. They derive a sophisticated recurrence relation between these two quantities, that finally implies that $\mathrm{R}_\delta(\text{3-}\mathsf{MAJ}_h) \geq (1-2\delta)(2+q)^h$, where $(1-2\delta)q^h$ is a lower bound on the probability p_h^δ that a randomized algorithm with error at most δ queries a special variable, called the "absolute minority", on inputs drawn from the hard distribution. They observe that any randomized decision tree with error at most δ must query at least one variable with probability $1-2\delta$. This variable has probability 3^{-h} of being the absolute minority, so $q \geq 1/3$, and the above lower bound follows.

We obtain the new lower bound by proving that $p_h^\delta \geq (1-2\delta)2^{-h}$, i.e., $q \geq 1/2$, which immediately implies the improved lower bound for $\mathrm{R}_\delta(\text{3-}\mathsf{MAJ}_h)$. We examine the relationship between p_h^δ and p_{h-1}^δ, by encoding a height $h-1$ instance into a height h instance, and using an algorithm for the latter. Analyzing our encoding requires understanding the behavior of all decision trees on 3 variables, and this can be done by exhaustively considering all such trees.

One can ask whether this is the best possible recurrence, and it may be possible to improve it by, say, encoding height $h-2$ instances into height h instances. Unfortunately we are unable to prove a claim analogous to Claim 3 in the case of such a recurrence, as the number of possible decision trees for 9 variables is too large to check exhaustively by hand. We nevertheless conjecture that such a claim exists for a two-level encoding scheme. More details will be provided in the journal version.

New algorithm. The naive algorithm and the algorithm of Jayram *et al.* are examples of *depth-k* recursive algorithms for 3-MAJ_h, for $k = 1, 2$, respectively. A depth-k recursive algorithm is a collection of subroutines, where each subroutine evaluates a node (possibly using information about other previously evaluated nodes), satisfying the following constraint: when a subroutine evaluates a node v, it is only allowed to call other subroutines to evaluate children of v at depth at most k, but is not allowed to call subroutines or otherwise evaluate children that are deeper than k. (Our notion of depth-1 is identical to the terminology "directional" that appears in the literature. In particular, the naive recursive algorithm is a directional algorithm.)

We present an improved depth-two recursive algorithm. To evaluate the root of the majority formula, we recursively evaluate one grandchild from each of two distinct children of the root. The grandchildren "give an opinion" about the values of their parents. The opinion guides the remaining computation in a natural manner: if the opinion indicates that the children are likely to agree, we evaluate the two children in sequence to confirm the opinion, otherwise we evaluate the third child. If at any point the opinion of the nodes evaluated so far changes, we modify our future computations accordingly. A key innovation is the use of an algorithm optimized to compute the value of a *partially evaluated* formula. In our analysis, we recognize when incorrect opinions are formed, and take advantage of the fact that this happens with smaller probability.

We do not believe that the algorithm we present here is optimal. Indeed, we conjecture that even better algorithms exist that follow the same high level intuition applied for depth-k recursion for $k > 2$. However, it seems new insights are required to analyze the performance of deeper recursions, as the formulas describing their complexity become unmanageable for $k > 2$.

Organization. We prepare the background for our main results Sect. 2. In Sect. 3 we prove the new lower bound for 3-MAJ. The new algorithm for the problem is described and analyzed in Sect. 4.

2 Preliminaries

We write $u \in_{\mathrm{R}} D$ to state that u is sampled from the distribution D. If X is a finite set, we identify X with the uniform distribution over X, and so, for instance, $u \in_{\mathrm{R}} X$ denotes a uniform element of X.

2.1 Distributional Complexity

A variant of the randomized complexity we use is *distributional* complexity. Let $\mathcal{D}_n$ be the set of distributions over $\{0,1\}^n$. The *cost* $\mathrm{C}(A, D)$ of a randomized decision tree algorithm A on n variables with respect to a distribution $D \in \mathcal{D}_n$ is the expected number of bits queried by A when x is sampled from D and over the random coins of A. The *distributional complexity* of a function f on n variables for δ two-sided error is $\Delta_\delta(f) = \max_{D \in \mathcal{D}_n} \min_{A \in \mathcal{P}_f^\delta} \mathrm{C}(A, D)$. The following observation is a well established route to proving lower bounds on worst case complexity.

Proposition 2. $\mathrm{R}_\delta(f) \geq \Delta_\delta(f)$.

2.2 The 3-MAJ_h Function and the Hard Distribution

Let $\mathsf{MAJ}(x)$ denote the boolean majority function of its input bits. The ternary majority function 3-MAJ_h is defined recursively on $n = 3^h$ variables, for every $h \geq 0$. We omit the height h when it is obvious from context. For $h = 0$ it is the identity function. For $h > 0$, let x be an input of length n and let $x^{(1)}, x^{(2)}, x^{(3)}$ be the first, second, and third $n/3$ variables of x. Then

$$\text{3-}\mathsf{MAJ}(x) = \mathsf{MAJ}(\text{3-}\mathsf{MAJ}(x^{(1)}),\ \text{3-}\mathsf{MAJ}(x^{(2)}),\ \text{3-}\mathsf{MAJ}(x^{(3)})).$$

In other terms, 3-MAJ_h is defined by the read-once formula on the complete ternary tree T_h of height h in which every internal node is a majority gate and the leaves are the input variables. For every node v in T_h different from the root, let $P(v)$ denote the parent of v. We say that v and w are siblings if $P(v) = P(w)$. For any node v in T_h, let $Z(v)$ denote the set of variables associated with the leaves in the subtree rooted at v. We say that a node v is at depth d in T_h if the distance between v and the root is d. The root is therefore at depth 0, and the leaves are at depth h.

We now define recursively, for every $h \geq 0$, the set $\mathcal{H}_h$ of *hard inputs* of height h. The hard inputs consist of instances for which at each node v in the ternary tree, one child of v has value different from the value of v. For $b \in \{0, 1\}$, let $\mathcal{H}_h^b = \{x \in \mathcal{H}_h : \text{3-}\mathsf{MAJ}_h(x) = b\}$. The *hard distribution* on inputs of height h is defined to be the uniform distribution over $\mathcal{H}_h$.

For an $x \in \mathcal{H}_h$, the *minority path* $M(x)$ is the path, starting at the root, obtained by following the child whose value disagrees with its parent. For $0 \leq d \leq h$, the node of $M(x)$ at depth d is called the depth d minority node, and is denoted by $M(x)_d$. We call the leaf $M(x)_h$ of the minority path the *absolute minority* of x, and denote it by $m(x)$.

2.3 The Jayram-Kumar-Sivakumar Lower Bound

For a deterministic decision tree algorithm B computing 3-MAJ_h, let $L_B(x)$ denote the set of variables queried by B on input x. Recall that $\mathcal{P}^{\delta}_{\text{3-}\mathsf{MAJ}_h}$ is the set of all randomized decision tree algorithms that compute 3-MAJ_h with two-sided error at most δ. Jayram *et al.* define the function $I^{\delta}(h, d)$, for $d \leq h$:

$$I^{\delta}(h, d) = \min_{A \in \mathcal{P}^{\delta}_{\text{3-}\mathsf{MAJ}_h}} \mathbb{E}_{x \in_{\mathrm{R}} \mathcal{H}_h, B \in_{\mathrm{R}} A}[|Z(M(x)_d) \cap L_B(x)|].$$

In words, it is the minimum over algorithms computing 3-MAJ_h, of the expected number of queries below the dth level minority node, over inputs from the hard distribution. Note that $I^{\delta}(h, 0) = \min_{A \in \mathcal{P}^{\delta}_{\text{3-}\mathsf{MAJ}_h}} \mathrm{C}(A, \mathcal{H}_h)$, and therefore by Proposition 2, $\mathrm{R}_{\delta}(\text{3-}\mathsf{MAJ}_h) \geq I^{\delta}(h, 0)$.

We define $p_h^{\delta} = I^{\delta}(h, h)$, which is the minimal probability that a δ-error algorithm A queries the absolute minority of a random hard x of height h.

Jayram *et al.* prove a recursive lower bound for $I^{\delta}(h, d)$ using information theoretic arguments. A more elementary proof can be found in Ref. [6].

Theorem 3 (Jayram, Kumar, Sivakumar [5]). *For all $0 \le d < h$:*

$$I^\delta(h,d) \ge I^\delta(h,d+1) + 2I^\delta(h-1,d).$$

A simple computation using their recursion gives $I^\delta(h,0) \ge \sum_{i=0}^{h} \binom{h}{i} 2^{h-i} p_i^\delta$. Putting this together with the fact that $\mathrm{R}_\delta(\text{3-MAJ}_h) \ge I^\delta(h,0)$, we get the following corollary:

Corollary 4. *Let $q, a > 0$ such that $p_i^\delta \ge a \cdot q^i$ for all $i \in \{0,1,2,\ldots,h\}$. Then $\mathrm{R}_\delta(\text{3-MAJ}_h) \ge a(2+q)^h$.*

As mentioned in Sect. 1, Jayram *et al.* obtain the $(1-2\delta)(7/3)^h$ lower bound from this corollary by observing that $p_h^\delta \ge (1-2\delta)(1/3)^h$.

3 Improved Lower Bound

Theorem 5. *For every error $\delta > 0$ and height $h \ge 0$, we have $p_h^\delta \ge (1-2\delta)2^{-h}$.*

Proof. We prove this theorem by induction. Clearly, $p_0^\delta \ge 1-2\delta$. It then suffices to show that $2p_h^\delta \ge p_{h-1}^\delta$ for $h \ge 1$. We do so by reduction as follows: let A be a randomized algorithm that achieves the minimal probability p_h^δ for height h formulae. We construct a randomized algorithm A' for height $h-1$ formulae such that the probability that A' errs is at most δ, and A' queries the absolute minority with probability at most $2p_h^\delta$. Since p_{h-1}^δ is the minimum probability of querying the absolute minority over all randomized algorithms on inputs of height $h-1$ with error at most δ, this implies that $2p_h^\delta \ge p_{h-1}^\delta$.

We now specify the reduction. For the sake of simplicity, we omit the error δ in the notation. We use the following definition:

Definition 6 (One level encoding scheme). *A* one level encoding scheme *is a bijection $\psi : \mathcal{H}_{h-1} \times \{1,2,3\}^{3^{h-1}} \to \mathcal{H}_h$, such that for all (y,r) in the domain, $\text{3-MAJ}_{h-1}(y) = \text{3-MAJ}_h(\psi(y,r))$.*

Let $c : \{0,1\} \times \{1,2,3\} \to \mathcal{H}_1$ satisfying $b = \mathsf{MAJ}(c(b,s))$ for all inputs (b,s). Define the one level encoding scheme ψ induced *by c as follows: $\psi(y,r) = x \in \mathcal{H}_h$ such that for all $1 \le i \le 3^{h-1}$, $(x_{3i-2}, x_{3i-1}, x_{3i}) = c(y_i, r_i)$.*

To define A', we use the one level encoding scheme ψ induced by the following function: $c(y,1) = y01$, $c(y,2) = 1y0$, and $c(y,3) = 01y$.

On input y, algorithm A' picks a uniformly random string $r \in \{1,2,3\}^{3^{h-1}}$, and runs A on $x = \psi(y,r)$. Observe that A' has error at most δ as $\text{3-MAJ}_{h-1}(y) = \text{3-MAJ}_h(\psi(y,r))$ for all r, and A has error at most δ. We claim now:

$$2 \Pr_{A,\, x \in_{\mathrm{R}} \mathcal{H}_h}[A(x) \text{ queries } x_{m(x)}] \quad \ge \quad \Pr_{A',\, (y,r) \in_{\mathrm{R}} \mathcal{H}'_h}[A'(y,r) \text{ queries } y_{m(y)}] \qquad (1)$$

where $\mathcal{H}'_h$ is the uniform distribution over $\mathcal{H}_{h-1} \times \{1,2,3\}^{3^{h-1}}$.

We prove this inequality by taking an appropriate partition of the probabilistic space of hard inputs $\mathcal{H}_h$, and prove Eq. 1 separately, on each set in the partition. For $h = 1$, the two classes of the partition are $\mathcal{H}_1^0$ and $\mathcal{H}_1^1$. For $h > 1$, the partition consists of the equivalence classes of the relation $\sim$ defined by $x \sim x'$ if $x_i = x'_i$ for all i such $P(i) \neq P(m(x))$ in the tree T.

Because ψ is a bijection, observe that this also induces a partition of (y, r), where $(y, r) \sim (y', r')$ iff $\psi(y, r) \sim \psi(y', r')$. Also observe that every equivalence class contains three elements. Let S be an equivalence class of $\sim$. Then Eq. 1 follows from the following stronger statement: for every S, and for all B in the support of A, it holds that

$$\begin{aligned} 2 \Pr_{x \in_{\mathrm{R}} \mathcal{H}_h} &[B(x) \text{ queries } x_{m(x)} \mid x \in S] \\ &\geq \Pr_{(y,r) \in_{\mathrm{R}} \mathcal{H}'_h} [B'(y, r) \text{ queries } y_{m(y)} \mid \psi(y, r) \in S] \ , \end{aligned} \tag{2}$$

where B' is the algorithm that computes $x = \psi(y, r)$ and then evaluates $B(x)$.

The same proof applies to all sets S, but to simplify the notation, we consider a set S that satisfies the following: for $x \in S$, we have $m(x) \in \{1, 2, 3\}$ and that $x_{m(x)} = 1$. Observe that for each $j > 3$, the jth bits of all three elements in S coincide. Therefore, the restriction of B to the variables (x_1, x_2, x_3), when looking only at the three inputs in S, is a well-defined decision tree on three variables. We call this restriction B_1, and formally it is defined as follows: for each query x_j made by B for $j > 3$, B_1 simply uses the value of x_j that is shared by all $x \in S$ and that we hard-wire into B_1; for each query x_j made by B where $j \in \{1, 2, 3\}$, B_1 actually queries x_j. Note that the restriction B_1 does not necessarily compute $\textsf{3-MAJ}_1(x_1x_2x_3)$, for two reasons. Firstly, B_1 is derived from B, which may err on particular inputs. But even if $B(x)$ correctly computes $\textsf{3-MAJ}_h(x)$, it might happen that B never queries any of x_1, x_2, x_3, or it might query one and never query a second one, etc.

For any $x \in S$, recall that we write $(y, r) = \psi^{-1}(x)$. It holds for our choice of S that $m(y) = 1$ because we assumed $m(x) \in \{1, 2, 3\}$ and also $y_1 = y_{m(y)} = 0$ because we assumed $x_{m(x)} = 1$.

Observe that, for inputs $x \in S$, B queries $x_{m(x)}$ iff B_1 queries the minority among x_1, x_2, x_3. Also, $B'(y, r)$ queries $y_{m(y)}$ iff $B_1(\psi(0, r_1))$ queries x_{r_1} (cf. definition of c used by A'). Furthermore, the distribution of $x_1x_2x_3$ when $x \in_{\mathrm{R}} S$ is uniform over $\mathcal{H}_1^0$. Similarly, the distribution of r_1 over uniform (y, r) conditioned on $\psi(y, r) \in S$ is identical to that of $(0, r_1) = \psi^{-1}(x_1x_2x_3)$ for $x_1x_2x_3 \in_{\mathrm{R}} \mathcal{H}_1^0$. Thus Eq. 2 is equivalent to:

$$\begin{aligned} 2 \Pr_{x \in_{\mathrm{R}} \mathcal{H}_1^0} &[B_1(x) \text{ queries } x_{m(x)}] \\ &\geq \Pr_{x \in_{\mathrm{R}} \mathcal{H}_1^0} [B_1(x) \text{ queries } x_{r_1} \text{ where } (0, r_1) = \psi^{-1}(x)] \ . \end{aligned} \tag{3}$$

Observe that Eq. 3 holds trivially if B_1 makes no queries, since then both sides equal 0. Therefore it is enough to consider only the case where B_1 makes at least one query. For any decision tree algorithm Q on three bits, which makes at least one query, we define the number ρ_Q as:

$$\rho_Q = \frac{\Pr_{x \in_{\mathrm{R}} \mathcal{H}_1^0}[Q(x) \text{ queries } x_{m(x)}]}{\Pr_{x \in_{\mathrm{R}} \mathcal{H}_1^0}[Q(x) \text{ queries } x_{r_1} \text{ where } (0, r_1) = \psi^{-1}(x)]} \ .$$

Note that the denominator is at least 1/3, since Q queries x_{r_1} when x is such that r_1 is the index of the first query. We prove that ρ_Q is always at least 1/2, by describing a decision tree algorithm Q' which minimizes ρ_Q. The algorithm Q' is defined as follows: first query x_1, if $x_1 = 0$, stop, else if $x_1 = 1$, query x_2 and stop.

Claim. The algorithm Q' gives $\rho_{Q'} = 1/2$, and this is the minimal possible ρ_Q among all deterministic decision tree algorithms making at least one query.

To prove the claim we first evaluate $\rho_{Q'}$. The numerator equals 1/3 since the minority is queried only when $x = 100$, while the denominator equals 2/3 since x_{r_1} is queried when x is 001 or 100.

Let now be Q any algorithm which makes at least one query, we prove that $\rho_Q \geq 1/2$. Without loss of generality, we may suppose that the first query is x_1. We distinguish two cases.

If Q makes a second query when the first query is evaluated to 0 then the numerator is at least 2/3 since for the second query there is also an x for which $m(x)$ is the index of this query. But the denominator is at most 1, and therefore in that case $\rho_Q \geq 2/3$. If Q does not make a second query when the first query is evaluated to 0 then the denominator is at most 2/3 since for $x = 010$, we have $r_1 = 3$, but x_3 is not queried. Since the numerator is at least 1/3, we have in that case $\rho_Q \geq 1/2$.

To handle a general S, we replace $\{1, 2, 3\}$ with $m(x)$ and its two siblings. For S such that $x \in S$ satisfies $x_{m(x)} = 0$, the optimal algorithm Q' is the same as the one described above, except that each 0 is changed to 1 and vice versa.

Therefore Eq. 3 holds for every B_1, which implies the theorem. □

Combining Corollary 4 and Theorem 5, we obtain the following.

Corollary 7. $\mathrm{R}_\delta(3\text{-}\mathsf{MAJ}_h) \geq (1 - 2\delta)(5/2)^h$.

4 Improved Depth-Two Algorithm

In this section, we present a new zero-error algorithm for computing $3\text{-}\mathsf{MAJ}_h$. For the key ideas behind it, we refer the reader to Sect. 1.

As before, we identify the formula $3\text{-}\mathsf{MAJ}_h$ with a complete ternary tree of height h. In the description of the algorithm we adopt the following convention. Once the algorithm has determined the value b of the subformula rooted at a node v of the formula $3\text{-}\mathsf{MAJ}_h$, we also use v to denote this bit value b.

The algorithm is a combination of two depth-2 recursive algorithms. The first one, Evaluate, takes a node v of height $h(v)$, and evaluates the subformula rooted at v. The interesting case, when $h(v) > 1$, is depicted in Fig. 1. The first step, permuting the input, means applying a random permutation to the children y_1, y_2, y_3 of v and independent random permutations to each of the three sets of grandchildren.

The second algorithm, Complete, is depicted in Fig. 2. It takes two arguments v, y_1, and completes the evaluation of the subformula $3\text{-}\mathsf{MAJ}_h$ rooted at

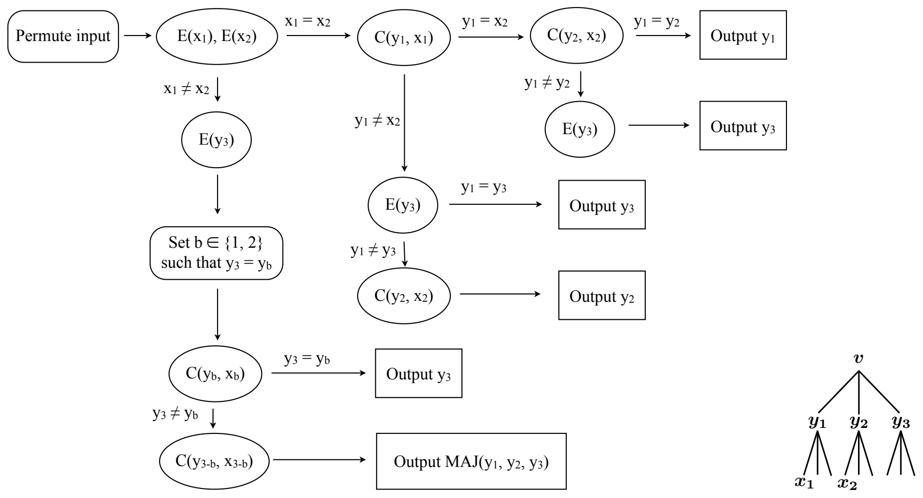

Fig. 1. Pictorial representation of algorithm Evaluate on a subformula of height $h(v) \geq 2$ rooted at v. It is abbreviated by the letter 'E' when called recursively on descendants of v. The letter 'C' abbreviates the second algorithm Complete.

node v, where $h(v) \geq 1$, and y_1 is a child of v whose value has already been evaluated. The first step, permuting the input, means applying a random permutation to the children y_2, y_3 of v and independent random permutations to each of the two sets of grandchildren of y_2, y_3. Note that this is similar in form to the depth 2 algorithm of [5].

To evaluate an input of height h, we invoke Evaluate(r), where r is the root. The correctness of the two algorithms follows by inspection—they determine the values of as many children of the node v as is required to compute the value of v.

For the complexity analysis, we study the expected number of queries they make for a worst-case input of fixed height h. (*A priori*, we do not know if such an input is a hard input as defined in Section 2.2.) Let $T(h)$ be the worst-case complexity of Evaluate(v) for v of height h. For Complete(v, y_1), we distinguish between two cases. Let y_1 be the child of node v that has already been evaluated. The complexity given that y_1 is the minority child of v is denoted by S^{m}, and the complexity given that it is a majority child is denoted by S^{M}.

The heart of our analysis is the following set of recurrences that relate T, S^{M} and S^{m} to each other.

Lemma 8. *It holds that $S^{\mathrm{m}}(1) = 2$, $S^{\mathrm{M}}(1) = \frac{3}{2}$, $T(0) = 1$, and $T(1) = \frac{8}{3}$.*

For all $h \geq 1$, it holds that

$$S^{\mathrm{M}}(h) \leq S^{\mathrm{m}}(h) \qquad \text{and} \qquad S^{\mathrm{M}}(h) \leq T(h) \ . \tag{4}$$

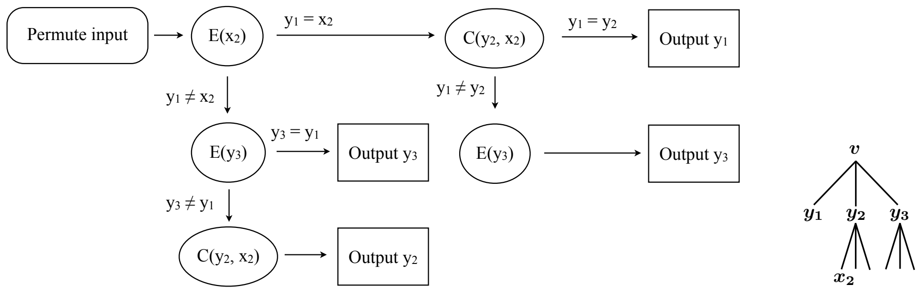

Fig. 2. Pictorial representation of algorithm COMPLETE on a subformula of height $h \geq 1$ rooted at v one child y_1 of which has already been evaluated. It is abbreviated by the letter 'C' when called recursively on descendants of v. Calls to EVALUATE are denoted 'E'.

Finally, for all $h \geq 2$, it holds that

$$S^{\mathrm{m}}(h) = T(h-2) + T(h-1) + \frac{2}{3}\,S^{\mathrm{M}}(h-1) + \frac{1}{3}\,S^{\mathrm{m}}(h-1)\ , \tag{5}$$

$$S^{\mathrm{M}}(h) = T(h-2) + \frac{2}{3}\,T(h-1) + \frac{1}{3}\,S^{\mathrm{M}}(h-1) + \frac{1}{3}\,S^{\mathrm{m}}(h-1)\ , \quad \textit{and} \tag{6}$$

$$T(h) = 2\,T(h-2) + \frac{23}{27}\,T(h-1) + \frac{26}{27}\,S^{\mathrm{M}}(h-1) + \frac{18}{27}\,S^{\mathrm{m}}(h-1)\ . \tag{7}$$

Proof. We prove these relations by induction. The bounds for $h \in \{0,1\}$ follow immediately by inspection of the algorithms. To prove the statement for $h \geq 2$, we assume the recurrences hold for all $l < h$. Observe that it suffices to prove Equations (5), (6), (7) for height h, since the values of the coefficients immediately imply that Inequalities (4) holds for h as well.

Equation (5). Since COMPLETE(v, y_1) always starts by computing the value of a grandchild x_2 of v, we get the first term $T(h-2)$ in Eq. (5). It remains to show that the worst-case complexity of the remaining queries is $T(h-1) + (2/3)S^{\mathrm{M}}(h-1) + (1/3)S^{\mathrm{m}}(h-1)$.

Since y_1 is the minority child of v, we have that $y_1 \neq y_2 = y_3$. The complexity of the remaining steps is summarized in the next table in the case that the three children of node y_2 are not all equal. In each line of the table, the worst case complexity is computed given the event in the first cell of the line. The second cell in the line is the probability of the event in the first cell over the random permutation of the children of y_2. This gives a contribution of $T(h-1) + (2/3)S^{\mathrm{M}}(h-1) + (1/3)S^{\mathrm{m}}(h-1)$.

$S^{\mathrm{m}}(h)$ (we have $y_1 \neq y_2 = y_3$)		
event	probability	complexity
$y_2 = x_2$	2/3	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	1/3	$T(h-1) + S^{\mathrm{m}}(h-1)$

This table corresponds to the worst case, as the only other case is when all children of y_2 are equal, in which the cost is $T(h-1) + S^{\mathrm{M}}(h-1)$. Applying Inequality (4) for $h-1$, this is a smaller contribution than the case where the children are not all equal.

Therefore the worst case complexity for S^{m} is given by Eq. (5). We follow the same convention and appeal to this kind of argument also while deriving the other two recurrence relations.

Equation (6). Since COMPLETE(v, y_1) always starts by computing the value of a grandchild x_2 of v, we get the first term $T(h-2)$ in Eq. (6). There are then two possible patterns, depending on whether the three children y_1, y_2, y_3 of v are all equal. If $y_1 = y_2 = y_3$, we have in the case that all children of y_2 are not equal that:

$S^{\mathrm{M}}(h)$ if $y_1 = y_2 = y_3$		
event	probability	complexity
$y_2 = x_2$	2/3	$S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	1/3	$T(h-1)$

As in the above analysis of Eq. (5), applying Inequalities (4) for height $h-1$ implies that the complexity in the case when all children of y_2 are equal can only be smaller, therefore the above table describes the worst-case complexity for the case when $y_1 = y_2 = y_3$.

If y_1, y_2, y_3 are not all equal, we have two events $y_1 = y_2 \neq y_3$ or $y_1 = y_3 \neq y_2$ of equal probability as y_1 is a majority child of v. This leads to the following tables for the case where the children of y_2 are not all equal

$S^{\mathrm{M}}(h)$ given $y_1 = y_2 \neq y_3$		
event	prob.	complexity
$y_2 = x_2$	2/3	$S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	1/3	$T(h-1) + S^{\mathrm{m}}(h-1)$

$S^{\mathrm{M}}(h)$ given $y_1 = y_3 \neq y_2$		
event	prob.	complexity
$y_2 = x_2$	2/3	$T(h-1)$
$y_2 \neq x_2$	1/3	$T(h-1) + S^{\mathrm{m}}(h-1)$

As before, one can apply Inequalities (4) for height $h-1$ to see that the worst case occurs when the children of y_2 are not all equal.

From the above tables, we deduce that the worst-case complexity occurs on inputs where y_1, y_2, y_3 are not all equal. This is because one can apply Inequalities (4) for height $h-1$ to see that, line by line, the complexities in the table for the case $y_1 = y_2 = y_3$ are upper bounded by the corresponding entries in each of the latter two tables. To conclude Eq. (6), recall that the two events $y_1 = y_2 \neq y_3$ and $y_1 = y_3 \neq y_2$ occur with probability 1/2 each:

$$
\begin{aligned}
S^{\mathrm{M}}(h) = T(h-2) &+ \frac{1}{2}\left[\frac{2}{3}\, S^{\mathrm{M}}(h-1) + \frac{1}{3}\left(T(h-1) + S^{\mathrm{m}}(h-1)\right)\right] \\
&+ \frac{1}{2}\left[\frac{2}{3}\, T(h-1) + \frac{1}{3}\left(T(h-1) + S^{\mathrm{m}}(h-1)\right)\right] .
\end{aligned}
$$

Equation (7). Since EVALUATE(v) starts with two calls to itself to compute x_1, x_2, we get the first term $2\,T(h-2)$ on the right hand side. The full analysis of Eq. (7) is similar to those of Eq. (5) and Eq. (6); we defer it to the journal article. □

Theorem 9. *$T(h), S^{\mathrm{M}}(h)$, and $S^{\mathrm{m}}(h)$ are all in $\mathrm{O}(\alpha^h)$, where $\alpha \leq 2.64946$.*

Proof. We make an ansatz $T(h) \leq a\,\alpha^h$, $S^{\mathrm{M}}(h) \leq b\,\alpha^h$, and $S^{\mathrm{m}}(h) \leq c\,\alpha^h$, and find constants a, b, c, α for which we may prove these inequalities by induction.

The base cases tell us that $2 \leq c\alpha, \frac{3}{2} \leq b\alpha, 1 \leq a, \quad \text{and} \frac{8}{3} \leq a\alpha$.

Assuming we have constants that satisfy these conditions, and that the inequalities hold for all appropriate $l < h$, for some $h \geq 2$, we derive sufficient conditions for the inductive step to go through.

By the induction hypothesis, Lemma 8, and our ansatz, it suffices to show

$$a + \tfrac{3a+2b+c}{3}\alpha \leq c\,\alpha^2 \qquad a + \tfrac{2a+b+c}{3}\alpha \leq b\,\alpha^2 \qquad 2a + \tfrac{23a+26b+18c}{27}\alpha \leq a\,\alpha^2 \quad (8)$$

The choice $\alpha = 2.64946$, $a = 1.007$, $b = 0.55958\,a$, and $c = 0.75582\,a$ satisfies the base case as well as all the Inequalities (8), so the induction holds. □

References

[1] Blum, M., Impagliazzo, R.: General oracle and oracle classes. In: Proc. FOCS '87. pp. 118–126 (1987)

[2] Hartmanis, J., Hemachandra, L.: One-way functions, robustness, and non-isomorphism of NP-complete sets. In: Proc. Structure in Complexity Theory '87. pp. 160–173 (1987)

[3] Heiman, R., Newman, I., Wigderson, A.: On read-once threshold formulae and their randomized decision tree complexity. In: Proc. Structure in Complexity Theory '90. pp. 78–87 (1990)

[4] Heiman, R., Wigderson, A.: Randomized versus deterministic decision tree complexity for read-once boolean functions. In: Proc. Structure in Complexity Theory '91. pp. 172–179 (1991)

[5] Jayram, T., Kumar, R., Sivakumar, D.: Two applications of information complexity. In: Proc. STOC '03. pp. 673–682 (2003)

[6] Landau, I., Nachmias, A., Peres, Y., Vanniasegaram, S.: The lower bound for evaluating a recursive ternary majority function: an entropy-free proof. Tech. rep., Department of Statistics, University of California, Berkeley, CA, USA, http://www.stat.berkeley.edu/110 (2006), undergraduate Research Report

[7] Nisan, N.: CREW PRAMs and decision trees. In: Proc. STOC '89. pp. 327–335. ACM, New York, NY, USA (1989)

[8] Reichardt, B.W., Špalek, R.: Span-program-based quantum algorithm for evaluating formulas. In: Proceedings of the 40th ACM Symposium on Theory of Computing. pp. 103–112. ACM, New York, NY, USA (2008)

[9] Saks, M., Wigderson, A.: Probabilistic boolean decision trees and the complexity of evaluating game trees. In: Proc. FOCS '86. pp. 29–38 (1986)

[10] Santha, M.: On the Monte Carlo boolean decision tree complexity of read-once formulae. Random Structures and Algorithms 6(1), 75–87 (1995), http://dx.doi.org/10.1002/rsa.3240060108

[11] Snir, M.: Lower bounds for probabilistic linear decision trees. Combinatorica 9, 385–392 (1990)

[12] Tardos, G.: Query complexity or why is it difficult to separate $\mathbf{NP}^A \cap \mathbf{coNP}^A$ from $\mathbf{P}^A$ by a random oracle. Combinatorica 9, 385–392 (1990)