Improved bounds for the randomized decision tree complexity of recursive majority

Frédéric Magniez, Ashwin Nayak, Miklos Santha, David Xiao

Electronic Colloquium on Computational Complexity, Report No. 192 (2010)

Improved bounds for the randomized decision tree complexity of recursive majority*

Frédéric Magniez† Ashwin Nayak‡ Miklos Santha§ David Xiao¶

Abstract

We consider the randomized decision tree complexity of the recursive 3-majority function. For evaluating a 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 *et al.* (STOC '03). We also state a conjecture which would further improve the lower bound to $(1-2\delta)2.54355^h$.

Second, we improve the upper bound by giving a new zero-error randomized decision tree algorithm that has complexity at most $(1.007)\cdot 2.64946^h$, improving on the previous best known algorithm, which achieved $(1.004)\cdot 2.65622^h$.

Our lower bound follows from a better analysis of the base case of the recursion of Jayram *et al.*. Our algorithm uses a novel "interleaving" of two recursive algorithms.

*Partially supported by the French ANR Defis program under contract ANR-08-EMER-012 (QRAC project) and the European Commission IST STREP Project Quantum Computer Science (QSC) 25596.

†LIAFA, Univ. Paris 7, CNRS; F-75205 Paris, France. magniez@liafa.jussieu.fr

‡Department of Combinatorics and Optimization, and Institute for Quantum Computing, University of Waterloo; and Perimeter Institute for Theoretical Physics. Address: 200 University Ave. W., Waterloo, ON, N2L 3G1, Canada. Email: ashwin.nayak@uwaterloo.ca. Research supported in part by NSERC Canada. Research at Perimeter Institute for Theoretical Physics is supported in part by the Government of Canada through Industry Canada and by the Province of Ontario through MRI. Work done in part while visiting LRI—CNRS, Univ Paris-Sud, Orsay, France, and Centre for Quantum Technologies, National University of Singapore, Singapore.

§LIAFA, Univ. Paris 7, CNRS; F-75205 Paris, France; and Centre for Quantum Technologies, National University of Singapore, Singapore 117543; santha@lri.fr. Research at the Centre for Quantum Technologies is funded by the Singapore Ministry of Education and the National Research Foundation.

¶LIAFA, Univ. Paris 7, CNRS; F-75205 Paris, France; and Univ. Paris-Sud, F-91405 Orsay, France. dxiao@lri.fr

ISSN 1433-8092

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 with certainty. The cost associated with this computation is the number of input bits queried, all other computations are free. 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, for some $1 \leq i \leq n$. The leaves of the tree are labeled by one of the output values 0 or 1, and for every internal node, one of the outgoing edges is labeled by 0 and the other by the 1. For every input $x = x_1 \dots x_n$, there is a unique path in the tree leading from the root to one of the leaves, which follows, at every node, the outgoing edge whose label coincides with the value of the input bit corresponding to the label of the node. 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 $A(x) = f(x)$.

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 $0 \leq \delta \leq 1/2$, it holds that $\mathrm{R}^\delta(f) \leq \mathrm{R}(f) \leq \mathrm{D}(f)$, and it is also known [1, 2, 10] that $\mathrm{D}(f) \leq \mathrm{R}(f)^2$, and that for all constant $\delta \in (0, 1/2)$, $\mathrm{D}(f) < \mathrm{O}(\mathrm{R}^\delta(f)^3)$ [7].

Considerable attention in the literature has been given to the randomized complexity of functions computable by read-once formulae, that is by 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 [9] has shown that $\mathrm{R}(\mathsf{NAND}_h) = \mathrm{O}(n^c)$ 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 [8]. Since $\mathrm{D}(\mathsf{NAND}_h) = 2^h = n$ this implies that $\mathrm{R}(\mathsf{NAND}_h) = \Theta(\mathrm{D}(\mathsf{NAND}_h)^c)$, and Saks and Wigderson have also conjectured that this is the largest gap between deterministic and randomized complexity.

Conjecture 1.1 (Saks and Wigderson [8])**.** *For every boolean function f and constant $\delta \in [0, 1/2)$, $\mathrm{R}^\delta(f) = \Omega(\mathrm{D}(f)^c)$.*

For the randomized complexity of read-once threshold formula of depth d, Heiman, Newman, and Wigderson [4] proved a lower bound of $\Omega(n/2^d)$. Heiman and Wigderson [3] proved that the randomized complexity of every read-once formula f is at least $\Omega(\mathrm{D}(f)^{0.51})$.

After these initial successes one would have hoped that the simple model of decision tree algorithms migh 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 example where randomized algorithms were found to be more powerful than deterministic decision trees [8]. It is a read-once formula

on 3^h variable given by the complete ternary tree of height h whose internal vertices are majority gates. The deterministic decision tree complexity of $\mathsf{3\text{-}MAJ}_h$ is easily seen to be 3^h. There is a naive randomized recursive algorithm for $\mathsf{3\text{-}MAJ}_h$ that picks two random children of the root and recursively evaluates them, then evaluates the third child iff the value is not determined by the previously evaluated two children. It is easy to check that this has randomized complexity $(8/3)^h$. It was already observed by Saks and Wigderson [8] that this algorithm is not optimal. In spite of some similarities with the NAND_h function, no progress was reported on the randomized complexity of $\mathsf{3\text{-}MAJ}$ for 17 years, beyond the bounds derived for arbitrary boolean functions [4, 3]. 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 analysis in [5, Appendix B] is incorrect.) More importantly, they also prove a $(1-2\delta)(7/3)^h$ lower bound for the δ-error randomized decision tree complexity of $\mathsf{3\text{-}MAJ}_h$. 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 $\mathrm{R}^\delta(\mathsf{3\text{-}MAJ}_h) \geq (1-2\delta)(5/2)^h$. In the appendix we also state a conjecture which would further raise the lower bound to $(1-2\delta)2.54355^h$. 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 1.2. *For all $\delta \in [0, 1/2]$, we have $(1-2\delta)(5/2)^h \leq \mathrm{R}^\delta(\mathsf{3\text{-}MAJ}_h) \leq (1.007)2.64946^h$.*

New lower bound. For the lower bound they give, Jayram *et al.* consider a complexity measure related to the distributional complexity of $\mathsf{3\text{-}MAJ}_h$ with respect to a specific hard distribution (cf. Section 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 $\mathrm{R}^\delta(\mathsf{3\text{-}MAJ}_h) \geq (1-2\delta)(2+q)^h$, 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 the "absolute minority" on inputs drawn from the hard distribution. (The absolute minority is the unique leaf in the recursive majority tree over a hard instance such that the path leading from this leaf to the root has alternating values.) Jayram *et al.* observe that any randomized decision tree with error at most δ queries at least one variable with probability at least $1-2\delta$. 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 $\mathrm{R}^\delta(\mathsf{3\text{-}MAJ}_h)$. The obvious method for deriving a such a lower bound is to consider the queries that an algorithm makes beyond the first. This approach quickly runs aground, as it requires an analysis of the hard distribution conditioned upon values of a subset of the variables, which seems intractable. Instead, we examine the relationship between p_h^δ and p_{h-1}^δ, by embedding a height $h-1$ instance into one with height h and using an algorithm for the latter. Unlike the embedding used by Jayram *et al.* (which runs into the same difficulty as the obvious approach), the new embedding reduces the analysis to understanding the behavior of decision trees on 3 variables, which can be done by hand. In the appendix, we give a conjecture that would further improve the lower bound by relating p_h^δ to p_{h-2}^δ, although a complete analysis seems out of reach as it involves examining all possible decision trees on 9 variables.

New algorithm. The new algorithm we design arises from a more nuanced application of the intuition behind the two known algorithms. One way of viewing the naive recursive algorithm is that it strives to avoid evaluating the minority child of a node. A more fruitful view is that it attempts to make an informed opinion on the value of a node by computing the value of a random child. The algorithm described in the appendix of [5] can also be viewed in this light, and performs notably better, achieving complexity $(1.004) \cdot 2.65622^h$.

The algorithms mentioned above are examples of *depth-k* recursive algorithms for $\mathsf{3\text{-}MAJ}_h$, 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.)

The algorithm we present is an improved depth-two recursive algorithm. It recursively computes the values of two grandchildren from distinct children, to form an opinion on the values of the corresponding children. The opinion guides the remaining computation in a natural manner, i.e., if the opinion indicates that the two children are likely to be majority children, we evaluate the children in sequence to confirm the opinion. At any point, if the value of a child refutes it, we update our opinion, and 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. The rest of the article is organized as follows. We prepare the background for our main results in Section 2. In Section 3 we prove the new lower bound for 3-MAJ. We conjecture a better lower bound in Section A in the appendix. The new algorithm for the problem is described and analyzed in Section 4. A formal description of the algorithm occurs in Section B in the appendix.

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.

Fact 2.1. $\mathrm{R}^{\delta}(f) \geq \Delta^{\delta}(f)$.

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 $h \geq 0$. For $h = 0$ it is the identity function. For $h > 0$,

$$\begin{aligned}&\text{3-MAJ}_h(x_1 \dots x_{3^h})\\ &= \mathsf{MAJ}(\text{3-MAJ}_{h-1}(x_1 \dots x_{3^{h-1}}), \text{3-MAJ}_{h-1}(x_{3^{h-1}+1} \dots x_{2\cdot 3^{h-1}}), \text{3-MAJ}_{h-1}(x_{2\cdot 3^{h-1}+1} \dots x_{3^h})).\end{aligned}$$

If the height h of the formula is clear from the context, we drop the subscript from 3-MAJ$_h$.

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, (or equivalently, of length 3^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 $\mathcal{H}_h^b = \{x \in \mathcal{H}_h : \mathsf{3\text{-}MAJ}_h(x) = b\}$. 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 $0 \le d \le h$, 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 $\mathsf{3\text{-}MAJ}_h$, let $L_B(x)$ denote the set of variables queried by B on input x. Recall that $\mathcal{P}^{\delta}_{\mathsf{3\text{-}MAJ}_h}$ is the set of all randomized decision tree algorithms that compute $\mathsf{3\text{-}MAJ}_h$ with two-sided error at most δ. Jayram *et al.* define the function $I^{\delta}(h,d)$, for $d \le h$, as follows:

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

The expectation is taken over the choice of $B \in_{\mathrm{R}} A$ and the choice of input x. In words, it is the minimum over algorithms computing $\mathsf{3\text{-}MAJ}_h$, of the expected number of queries below the dth level minority node, over inputs from the hard distribution. Note that $I^{\delta}(h,0) = \min_{A \in \mathcal{P}^{\delta}_{\mathsf{3\text{-}MAJ}_h}} \mathrm{C}(A, \mathcal{H}_h)$, and therefore by Fact 2.1:

$$\mathrm{R}^{\delta}(\mathsf{3\text{-}MAJ}_h) \ge I^{\delta}(h,0). \tag{1}$$

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

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 2.2 (Jayram, Kumar, Sivakumar [5])**.** *For all $0 \le d < h$, it holds that*

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

A simple computation using their recursion gives $I(h,0) \ge \sum_{i=0}^{h} \binom{h}{i} 2^{h-i} p_i^{\delta}$. Putting this together with Equation 1 we get the following corollary:

Corollary 2.3. *Let $q, a > 0$ such that $p_i^{\delta} \ge a \cdot q^i$ for all $i \in \{0,1,2,\ldots,h\}$. Then $\mathrm{R}^{\delta}(\mathsf{3\text{-}MAJ}_h) \ge a(2+q)^h$.*

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

3 Improved Lower Bound

Theorem 3.1. *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, $p_0^{\delta} \ge 1-2\delta$, therefore, it 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 will omit the error δ in the notation. We use the following definition:

Definition 3.2 (One level encoding scheme)**.** A *one level encoding scheme* is a map ψ which is a bijection for every $h \ge 1$, mapping $\mathcal{H}_{h-1} \times \{1,2,3\}^{3^{h-1}}$ to $\mathcal{H}_h$, such that for every (y,r) in its domain with $y \in \mathcal{H}_{h-1}$, $\mathsf{3\text{-}MAJ}_{h-1}(y) = \mathsf{3\text{-}MAJ}_h(\psi(y,r))$.

Let $c : \{0,1\} \times \{1,2,3\} \to \mathcal{H}_1$ be a function which for every $(b,s) \in \{0,1\} \times \{1,2,3\}$ satisfies $b = \mathsf{MAJ}(c(b,s))$. The one level encoding scheme ψ *induced* by c is defined for each $h \geq 1$ as follows: $\psi(y,r) = x \in \mathcal{H}_h$ such that for all $1 \leq i \leq 3^{h-1}$

$$(x_{3i-2}, x_{3i-1}, x_{3i}) = c(y_i, r_i).$$

To define A', we use the one level encoding scheme ψ induced by the function $c : \{0,1\} \times \{1,2,3\} \to \mathcal{H}_1$ defined as

$$c(y,r) = \begin{cases} y01 & \text{if } r = 1, \\ 1y0 & \text{if } r = 2, \\ 01y & \text{if } r = 3. \end{cases} \quad \text{and} \tag{2}$$

On input y, by definition A' picks a uniformly random string $r = r_1 \dots r_{3^{h-1}}$ from $\{1,2,3\}^{3^{h-1}}$, and runs A on $x = \psi(y,r)$. Observe that A' has error at most δ since $\text{3-MAJ}_{h-1}(y) = \text{3-MAJ}_h(\psi(y,r))$ for all r, and A has error at most δ. We claim now:

$$2 \Pr_{A,\, x \in_R \mathcal{H}_h}[A(x) \text{ queries } x_{m(x)}] \quad \geq \quad \Pr_{A',\, (y,r) \in_R \mathcal{H}'_h}[A'(y,r) \text{ queries } y_{m(y)}], \tag{3}$$

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 Equation 3 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 Equation 3 follows from the following stronger statement: for every S, and for all B in the support of A, it holds that

$$2 \Pr_{x \in_R \mathcal{H}_h}[B(x) \text{ queries } x_{m(x)} \mid x \in S] \quad \geq \quad \Pr_{(y,r) \in_R \mathcal{H}'_h}[B'(y,r) \text{ queries } y_{m(y)} \mid \psi(y,r) \in S]. \tag{4}$$

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 $\text{3-MAJ}_1(x_1x_2x_3)$, for two reasons. Firstly, B_1 is derived from B, which may err on particular inputs. But even if $B(x)$ correctly computes $\text{3-MAJ}_h(x)$, 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. Equation 2). Furthermore, the distribution of $x_1x_2x_3$ when $x \in_R S$ 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 $x_1x_2x_3 \in_R \mathcal{H}_1^0$. Thus Equation 4 is equivalent to:

$$2 \Pr_{x \in_R \mathcal{H}_1^0}[B_1(x) \text{ queries } x_{m(x)}] \quad \geq \quad \Pr_{x \in_R \mathcal{H}_1^0}[B_1(x) \text{ queries } x_{r_1} \text{ where } (0,r_1) = \psi^{-1}(x)]. \tag{5}$$

Observe that Equation 5 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:

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

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:

- Query x_1
- If $x_1 = 0$, stop
- If $x_1 = 1$, query x_2 and stop.

Claim 3.3. *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 we 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, 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 with each 0 changed to 1 and vice versa.

Therefore Equation 5 holds for every B_1, which implies the theorem. □

Combining Corollary 2.3 and Theorem 3.1, we obtain the following.

Corollary 3.4. $R^\delta(3\text{-}MAJ_h) \geq (1-2\delta)(5/2)^h$.

We conjecture that this can be improved to $R^\delta(3\text{-}MAJ_h) \geq (1-2\delta)2.54355^h$. See Section A for details.

4 Improved depth-two algorithm

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

As before, we identify the formula $3\text{-}MAJ_h$ 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 $3\text{-}MAJ_h$, 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 Figure 1; a formal description is given as **Algorithm 2** in the appendix (Section B). 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 Figure 2 and is described more formally as **Algorithm 3** in the appendix (Section B). It takes two arguments v, y_1, and completes the evaluation of the

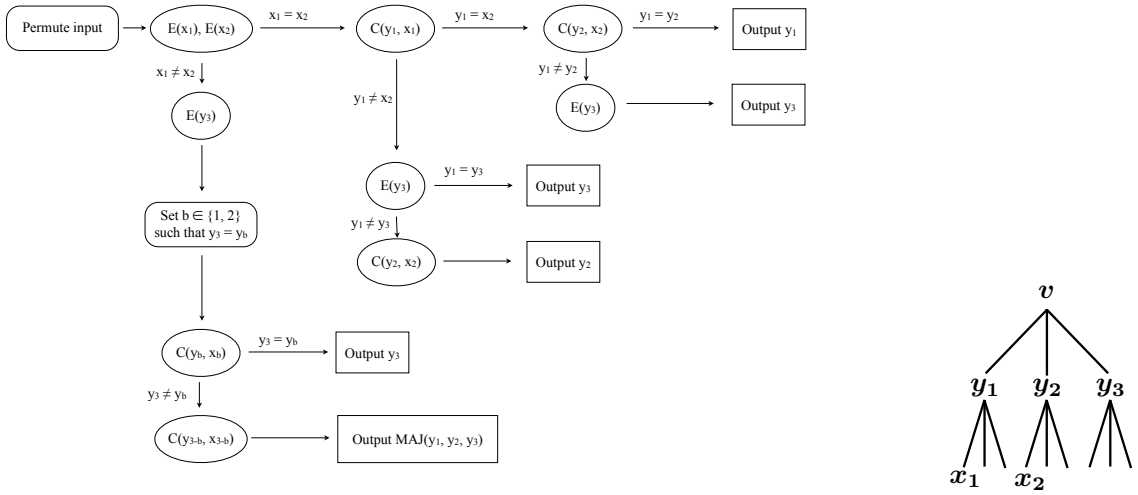


Figure 1: Pictorial representation of algorithm EVALUATE on a subformula of height $h(v) \geq 2$ rooted at v. It is abbreviated by the letter 'E' when called recursively on descendants of v. The letter 'C' abbreviates the second algorithm COMPLETE.

subformula $\mathsf{3\text{-}MAJ}_h$ rooted at 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].

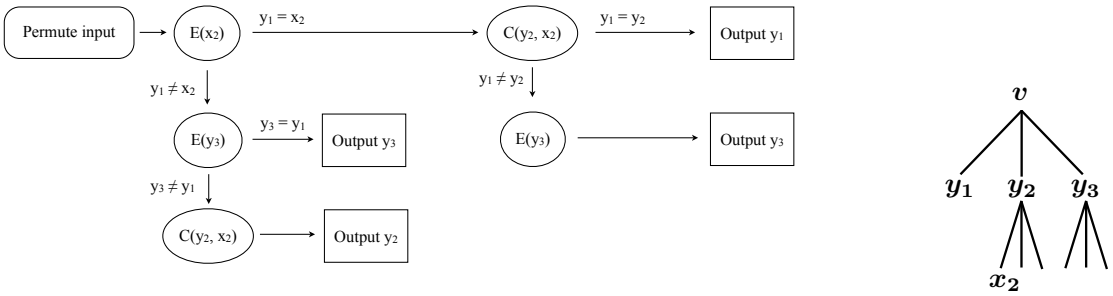


Figure 2: Pictorial representation of algorithm COMPLETE on a subformula of height $h \geq 1$ rooted at v one child y_1 of which has already been evaluated. It is abbreviated by the letter 'C' when called recursively on descendants of v. Calls to EVALUATE are denoted 'E'.

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. 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 4.1. *We have*

$$S^{\mathrm{m}}(1) = 2, \quad S^{\mathrm{M}}(1) = \frac{3}{2}, \quad T(0) = 1, \text{ and } \quad T(1) = \frac{8}{3}.$$

For all $h \geq 1$, we have

$$S^{\mathrm{M}}(h) \leq S^{\mathrm{m}}(h) \qquad \text{and} \qquad S^{\mathrm{M}}(h) \leq T(h). \tag{6}$$

Finally, for all $h \geq 2$, we have

$$S^{\mathrm{m}}(h) = T(h-2) + T(h-1) + \frac{2}{3} S^{\mathrm{M}}(h-1) + \frac{1}{3} S^{\mathrm{m}}(h-1), \tag{7}$$

$$S^{\mathrm{M}}(h) = T(h-2) + \frac{2}{3} T(h-1) + \frac{1}{3} S^{\mathrm{M}}(h-1) + \frac{1}{3} S^{\mathrm{m}}(h-1), \quad \text{and} \tag{8}$$

$$T(h) = 2\,T(h-2) + \frac{23}{27} T(h-1) + \frac{26}{27} S^{\mathrm{M}}(h-1) + \frac{18}{27} S^{\mathrm{m}}(h-1). \tag{9}$$

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 that Equations (7), (8), (9) for height h, since the values of the coefficients immediately imply that Inequalities (6) holds for h as well.

Equation (7). 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. (7). 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 (6) 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. (7). We follow the same convention and appeal to this kind of argument also while deriving the other two recurrence relations.

Equation (8). 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. (8). 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. (7), applying Inequalities (6) 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	probability	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	probability	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 (6) 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 (6) 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. (8), 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:

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

Equation (9). 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.

For the remaining complexity, we consider two possible cases, depending on whether the three children y_1, y_2, y_3 of v are equal. If $y_1 = y_2 = y_3$, assuming that the children of y_1 are not all equal, and the same for the children of y_2, we have

$T(h)$ given $y_1 = y_2 = y_3$		
event	probability	complexity
$y_1 = x_1, y_2 = x_2$	$4/9$	$2S^{\mathrm{M}}(h-1)$
$y_1 = x_1, y_2 \neq x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_1 \neq x_1, y_2 = x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_1 \neq x_1, y_2 \neq x_2$	$1/9$	$T(h-1) + S^{\mathrm{m}}(h-1)$

As before, the complexities are in non-decreasing order, and we observe that Inequalities (6) for height $h-1$ implies that in a worst case input the children of y_1 are not all equal, and the same for the children of y_2.

If y_1, y_2, y_3 are not all equal, we have three events $y_1 = y_2 \neq y_3$, $y_1 \neq y_2 = y_3$ and $y_3 = y_1 \neq y_2$ each of which occurs with probability $1/3$. This leads to the following analyses

$T(h)$ given $y_1 = y_2 \neq y_3$		
event	probability	complexity
$y_1 = x_1, y_2 = x_2$	$4/9$	$2S^{\mathrm{M}}(h-1)$
$y_1 = x_1, y_2 \neq x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 = x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 \neq x_2$	$1/9$	$T(h-1) + 2S^{\mathrm{m}}(h-1)$

$T(h)$ given $y_1 \neq y_2 = y_3$		
event	probability	complexity
$y_1 = x_1, y_2 = x_2$	$4/9$	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_1 = x_1, y_2 \neq x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 = x_2$	$2/9$	$T(h-1) + S^{\mathrm{M}}(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 \neq x_2$	$1/9$	$T(h-1) + 2S^{\mathrm{m}}(h-1)$

$T(h)$ given $y_3 = y_1 \neq y_2$		
event	probability	complexity
$y_1 = x_1, y_2 = x_2$	4/9	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_1 = x_1, y_2 \neq x_2$	2/9	$T(h-1) + S^{\mathrm{M}}(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 = x_2$	2/9	$T(h-1) + S^{\mathrm{m}}(h-1)$
$y_1 \neq x_1, y_2 \neq x_2$	1/9	$T(h-1) + 2S^{\mathrm{m}}(h-1)$

In all three events, we observe that Inequalities (6) for height $h-1$ implies that in a worst case input, the children of y_1 are not all equal, and the same for the children of y_2.

Applying Inequalities (6) for height $h-1$, it follows that line by line the complexities in the last three tables are at least the complexities in the table for the case $y_1 = y_2 = y_3$. Therefore the worst case also corresponds to an input in which y_1, y_2, y_3 are not all equal. We conclude Eq. (9) as before, by taking the expectation of the complexities in the last three tables. □

Theorem 4.2. *$T(h), S^{\mathrm{M}}(h)$, and $S^{\mathrm{m}}(h)$ are all in $\mathrm{O}(\alpha^h)$, where $\alpha \leq 2.64946$.*

Proof. We make an ansatz that $T(h) \leq a\,\alpha^h$, $S^{\mathrm{M}}(h) \leq b\,\alpha^h$, 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

$$2 \leq c\alpha, \quad \frac{3}{2} \leq b\alpha, \quad 1 \leq a, \quad \text{and} \quad \frac{8}{3} \leq a\alpha. \tag{10}$$

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 and Lemma 4.1, we have

$$\begin{aligned} S^{\mathrm{m}}(h) &\leq a\,\alpha^{h-2} + a\,\alpha^{h-1} + \frac{2b}{3}\,\alpha^{h-1} + \frac{c}{3}\,\alpha^{h-1}, \\ S^{\mathrm{M}}(h) &\leq a\,\alpha^{h-2} + \frac{2a}{3}\,\alpha^{h-1} + \frac{b}{3}\,\alpha^{h-1} + \frac{c}{3}\,\alpha^{h-1}, \quad \text{and} \\ T(h) &\leq 2a\,\alpha^{h-2} + \frac{23a}{27}\,\alpha^{h-1} + \frac{26}{27}\,\alpha^{h-1} + \frac{18}{27}\,\alpha^{h-1}. \end{aligned}$$

These would imply the required bounds on $S^{\mathrm{m}}(h), S^{\mathrm{M}}(h), T(h)$ if

$$\begin{aligned} a + \frac{3a+2b+c}{3}\alpha &\leq c\,\alpha^2, \\ a + \frac{2a+b+c}{3}\alpha &\leq b\,\alpha^2, \quad \text{and} \\ 2a + \frac{23a+26b+18c}{27}\alpha &\leq a\,\alpha^2. \end{aligned} \tag{11}$$

The choice $\alpha = 2.64946$, $a = 1.007$, $b = 0.55958\,a$, and $c = 0.75582\,a$ satisfies all the inequalities (10, 11), so that the proof by induction holds. □

References

[1] M. Blum and R. Impagliazzo. General oracle and oracle classes. In *Proc. FOCS '87*, pages 118–126, 1987.

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

[3] R. Heiman and A. Wigderson. Randomized versus deterministic decision tree complexity for read-once boolean functions. In *Proc. Structure in Complexity Theory '91*, pages 172–179, 1991.

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

[5] T. Jayram, R. Kumar, and D. Sivakumar. Two applications of information complexity. In *Proc. STOC '03*, pages 673–682, 2003.

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

[7] N. Nisan. CREW PRAMs and decision trees. In *Proc. STOC '89*, pages 327–335, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8.

[8] M. Saks and A. Wigderson. Probabilistic boolean decision trees and the complexity of evaluating game trees. In *Proc. FOCS '86*, pages 29–38, 1986.

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

[10] G. Tardos. 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.

A A conjectured better lower bound

Our proof from the previous section proceeds by proving a recurrence, using a one level encoding scheme, for the minimal probability that an algorithm queries the absolute minority bit. One can ask whether this is the best possible recurrence, and the following theorem hints that it may be possible to improve it by using higher level encoding schemes. Unfortunately we are unable to prove a claim analogous to Claim 3.3 in the case of such a recurrence, as the number of possible decision trees to minimize over is too large to handle by enumeration by hand. We nevertheless have a candidate for the minimal decision tree algorithm, which is the natural extension of the algorithm given in Theorem 3.1 for the one level recursion, and we leave as an open question whether or not our candidate indeed achieves the minimum.

As before, the base case satisfies $p_0^\delta \geq 1 - 2\delta$. In the following, we omit δ from the notation when convenient.

Definition A.1. A *two-level encoding scheme* is a map ψ for every $h \geq 2$, from $\mathcal{H}_{h-2} \times \Omega$ to $\mathcal{H}_h$ (Ω is a space of random coins for the encoding), satisfying for every (y, ω):

$$\mathsf{3\text{-}MAJ}_{h-1}(y) = \mathsf{3\text{-}MAJ}_h(\psi(y, \omega)).$$

Theorem A.2. *Assuming Conjecture A.3, for every $h \geq 0$, we have*

$$p_h^\delta \geq (1 - 2\delta)(\sqrt{13/47})^h > (1 - 2\delta)0.5259^h.$$

Proof. The proof follows the same structure as that of Theorem 3.1 but using two levels recursion: we show that $(47/13)p_h \geq p_{h-2}$. To prove this, for every deterministic algorithm A for height h formulae, we construct a randomized algorithm A' for height $h-2$ formulae such that the probability that A' queries the absolute minority is at most 47/13-times the probability that A queries the absolute minority.

To define A', we will use the two levels encoding scheme $\psi : \mathcal{H}_{h-2} \times \{1,2,3\}^{4\cdot 3^{h-2}} \to \mathcal{H}_h$, induced by the same function $c : \{0,1\} \times \{1,2,3\} \to \mathcal{H}_1$ we have used for the one level encoding scheme in Theorem 3.1. We recall that

$$c(y,r) = \begin{cases} y01 & \text{if } r = 1, \\ 1y0 & \text{if } r = 2, \\ 01y & \text{if } r = 3. \end{cases}$$

and we define the induced encoding ψ as follows: for every $1 \le i \le 3^{h-2}$,

$$\psi(y,R,r_1,r_2,r_3)_{9i-8} \dots \psi(y,R,r_1,r_2,r_3)_{9i} = \begin{cases} c(y_i,r_{1i}), c(0,r_{2i}), c(1,r_{3i}) & \text{if } R_i = 1, \\ c(1,r_{1i}), c(y_i,r_{2i}), c(0,r_{3i}) & \text{if } R_i = 2, \\ c(0,r_{1i}), c(1,r_{2i}), c(y_i,r_{3i}) & \text{if } R_i = 3. \end{cases}$$

Let $\Omega = \{1,2,3\}^{4\cdot 3^{h-2}}$, and we write $\omega \in \Omega$ as $\omega = (R, r_1, r_2, r_3)$. On input y, by definition A' samples uniform $\omega \in \Omega$ runs A on $x = \psi(y,\omega)$. As before, it suffices to prove that

$$(47/13) \Pr_{A, x \in \mathcal{H}_h}[A(x) \text{ queries } x_{m(x)}] \ge \Pr_{A', y \in \mathcal{H}_{h-2}, \omega \in \Omega}[A'(y,\omega) \text{ queries } y_{m(y)}] \tag{12}$$

We partition again $\mathcal{H}_h$, this time into sets of size 81. For $h = 2$, the two classes are $\mathcal{H}_2^0$ and $\mathcal{H}_2^1$. For $h > 2$, the partition consists of the equivalence classes of the relation defined by $x \sim x'$ if $x_i = x'_i$ for all i such $P(P(i)) \neq P(P(m(x)))$ in the tree T. Namely, an equivalence class consists of inputs that are identical everywhere except the 2-level subtree containing their absolute minority. We then prove that for every equivalence class S, and all B in the support of A, it holds that:

$$(47/13) \Pr_{x \in S}[B(x) \text{ queries } x_{m(x)}] \ge \Pr_{y,\omega}[B'(y,\omega) \text{ queries } y_{m(y)} \mid \psi(y,\omega) \in S]. \tag{13}$$

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

To prove this, suppose for simplicity of notation $m = m(x) \in \{1,\dots,9\}$ and $x_{m(x)} = 0$ for every $x \in S$. This implies that for all $x \in S$, if we set $(y,\omega) = \psi^{-1}(x)$, then $m(y) = 1$ and $y_1 = 0$. Let B_2 be the restriction of B to the first 9 bits, where for every query x_j, for $j > 9$, algorithm B_2 follows the outgoing edge of B according the common value of the jth bit of the elements in S, and for each query x_j where $j \in \{1,\dots,9\}$, B_2 also queries x_j.

Observe that $B(x)$ querying $x_{m(x)}$ corresponds to $B_2(x)$ querying $x_{m(x)}$, while $B'(y,\omega)$ querying $y_{m(y)}$ corresponds to $B_2(x)$ querying $x_{q(x)}$, where $q(x) = 3(R_1 - 1) + r_{R_1 1}$ and where $(0,(R_1,r_{11},r_{21},r_{31})) = \psi^{-1}(x)$. Namely, if $x = \psi(y,(R_1,r_{11},r_{21},r_{31}))$, then $q(x) \in \{1,\dots,9\}$ is the location where the encoding inserted y_1.

Therefore Equation 13 is equivalent to the following:

$$(47/13) \Pr_{x \in \mathcal{H}_2^0}[B_2(x) \text{ queries } x_{m(x)}] \ge \Pr_{x \in \mathcal{H}_2^0}[B_2(x) \text{ queries } x_{q(x)}] \tag{14}$$

For algorithms B_2 that query no nodes, Equation 14 is trivially satisfied as both sides equal 0. Therefore, we define for every decision tree algorithm Q on 9 bits which makes at least one query, ρ_Q as

$$\rho_Q = \frac{\Pr_{x \in \mathcal{H}_2^0}[Q(x) \text{ queries } x_{m(x)}]}{\Pr_{x \in \mathcal{H}_2^0}[Q(x) \text{ queries } x_{q(x)}]}. \tag{15}$$

Consider Algorithm 1 on 9 variables. (See also Figure 3 for a pictorial representation of the algorithm. In the figure, the symbol "$\perp$" means stop, and "ALL"' means to completely evaluate all variables, except the ones that cannot influence the output.)

We conjecture that the partial DT given in Algorithm 1 is the tree that minimizes the LHS of this inequality. (See also Figure 3 for a pictorial representation of the algorithm. In the figure, the symbol "$\perp$" means stop, and "ALL"' means to completely evaluate all variables, except ones that cannot influence the output.)

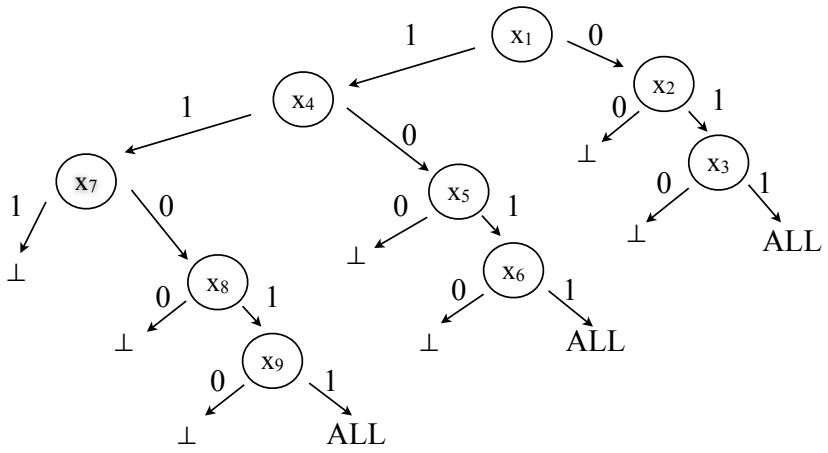


Figure 3: Picture of Algorithm 1

Conjecture A.3. *The algorithm given in Algorithm 1 (see also Figure 3) minimizes* ρ_{B_2}.

Assuming this conjecture, we now prove that Algorithm 1 achieves $\rho_{B_2} = (13/47)$. We refer to Figure 4, where we annotate the decision tree of Figure 3 with two numbers (inside the boxes). For each node v, right-side number is the number of choices of ω such that the query at v is $x_{q(x)}$. The left-side count is the number of choices of $x_1, \ldots, x_9$ such that the query at v is the absolute minority among $x_1, \ldots, x_9$.

We give a brief explanation of the counts, and leave the complete verification to the reader. First we consider the left-side counts: we assumed that the input evaluates to 0, so since the tree has height 2, the absolute minority has value 0. Therefore, we see that the only queries that might query the absolute minority are x_1, x_4, x_7 (since for all other queries, either we are not in the minority subtree, or if we are in the minority subtree then we have already queried the sole 0 in that subtree). We can verify for example that there are 9 hard inputs on which x_1 is the absolute minority: $(x_1, \ldots, x_9) = 011\ 001\ 001$ and the eight other inputs obtained by permuting x_4, x_5, x_6 and permuting x_7, x_8, x_9.

For the right-side counts, we observe that in general they are significantly larger because $x_{q(x)}$ is a *majority* node. For instance, for the right-side count on the node marked x_3, there are 9 inputs on which $x_1 = 0$ and $x_2 = 1$, and $q(x) = 3$. Namely, when $R_1 = 1, r_{11} = 3$, and r_{21}, r_{31} are free (there are 9 possibilities).

The sum of the left-hand counts is 13 and the sum of the right-hand counts is 47. This gives a ratio of 13/47. To lift our assumptions on S, it suffices to look at the set of all grandchildren of $M(S)_{h-2}$ rather than leaves $\{1, \ldots, 9\}$, and if the value of $m(x) = 1$ for $x \in S$, then it suffices to use Algorithm 1 except flipping all the 0's to 1's and vice versa.

□

Theorem A.4. *Assuming Conjecture A.5, for every* $h \geq 0$, *we have*

$$p_h^\delta \geq (1 - 2\delta)(\sqrt{13/44})^h > (1 - 2\delta)0.54355^h.$$

Proof. This theorem uses a two-level encoding that is more symmetric than that of Theorem A.2. Let c be as in the proof of Theorem A.2. We build the following encoding, where $(b, R, r_1, r_2, r_3) \in \{0,1\}^{3^{h-2}} \times \{1,2,3\}^{4 \cdot 3^{h-2}}$:

$$\psi(y, b, R, r_1, r_2, r_3)_{9i-8} \ldots \psi(y, R, r_1, r_2, r_3)_{9i} = \begin{cases} c(y_i, r_{1i}), c(b, r_{2i}), c(1-b, r_{3i}) & \text{if } R_i = 1, \\ c(1-b, r_{1i}), c(y_i, r_{2i}), c(b, r_{3i}) & \text{if } R_i = 2, \\ c(b, r_{1i}), c(1-b, r_{2i}), c(y_i, r_{3i}) & \text{if } R_i = 3. \end{cases}$$

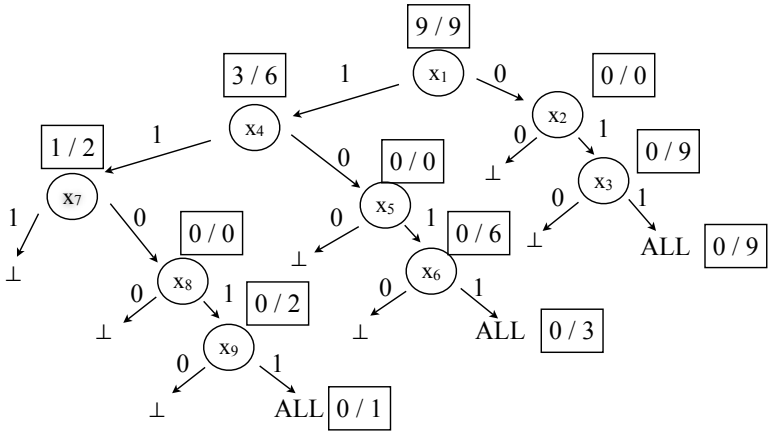


Figure 4: Algorithm 1, annotated for Theorem A.2

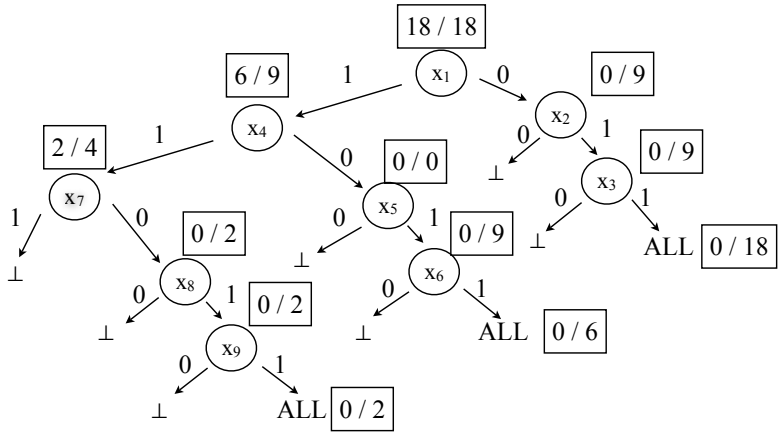


Figure 5: Algorithm 1, annotated for Theorem A.4

We note that this encoding ψ is no longer a bijection. However, one can make essentially the same argument as Theorem A.2 to show that it suffices to prove that the following ratio is at least 13/44 for all decision trees Q on 9 variables:

$$\rho_Q = \frac{\Pr_{x\in\mathcal{H}_2^0}[Q(x) \text{ queries } x_{m(x)}]}{\Pr_{x\in\mathcal{H}_{2,q(x)}^0}[Q(x) \text{ queries } x_{q(x)}]}. \tag{16}$$

where $q(x) = 3(R_1 - 1) + r_{R_1 1}$ and where $(0, (b_1, R_1, r_{11}, r_{21}, r_{31}))$ is uniformly sampled among the set of preimages $\psi^{-1}(x)$.

Conjecture A.5. *The algorithm of Figure 3 (see also Figure 3) minimizes ρ_Q, with value equal to* 13/44.

Our candidate algorithm minimizing Eq. (16) is also the same, and the counts (analogous to those given in Figure 4 for the the proof of Theorem A.2) are given in Figure 5. (To make the left and right-hand quantities comparable, we multiplied left-hand counts by 2. This is because the probability space of x is half the size of the probability space of $x, q(x)$). This leads to a ratio of 13/44. □

A.1 Intuition behind the conjecture

We believe the conjectures Conjecture A.3 and Conjecture A.5 because they represent a natural strategy for minimizing the ratio ρ_Q. Namely, the algorithm in Figure 3 encodes the strategy that we query the nodes in order, but we skip nodes that have increased probability of being the absolute minority. This occurs for instance with the first query x_1: if $x_1 = 0$ then we know that the next query cannot be the absolute minority (since the absolute minority has value 0 and it is the unique leaf in its subtree with value 0), so we are comfortable querying x_2. If $x_1 = 1$, then there is an increased probability that x_2 is the absolute minority, so we skip it and move to x_4, which is in the next subtree. This strategy also occurs at depth 1 from the root: if we evaluate $y_1 = 1$ then we know that the absolute minority must be a child of y_1, so we can safely evaluate all the children of y_2, y_3. On the other hand, if $y_1 = 0$, then there is the absolute minority must be a child of y_2 or y_3, and we stop evaluating in order to avoid evaluating the absolute minority.

B Formal description of the algorithms

In this section we present a formal description of the depth-two recursive algorithms for 3-MAJ$_h$ studied in Section 4. Pieces corresponding to $h \geq 2$ are depicted in Figures 1 and 2.

Algorithm 1 Conjectured optimal partial DT for Equation 15, see Figure 3

```
evaluate x_1
if x_1 = 0 then
  evaluate x_2
  if x_2 = 0 then
    stop
  else
    evalute x_3
    if x_3 = 0 then
      stop
    else
      exhaustively evaluate remaining variables
    end if
  end if
else
  evaluate x_4
  if x_4 = 0 then
    evaluate x_5
    if x_5 = 0 then
      stop
    else
      evaluate x_6
      if x_6 = 0 then
        stop
      else
        exhaustively evaluate remaining variables
      end if
    end if
  else
    evaluate x_7
    if x_7 = 0 then
      evaluate x_8
      if x_8 = 0 then
        stop
      else
        evaluate x_9
        if x_9 = 0 then
          stop
        else
          exhaustively evaluate remaining variables
        end if
      end if
    end if
  end if
end if
```

Algorithm 2 EVALUATE(v): evaluate a node v.

Input: Node v with subtree of height $h(v)$.
Output: the bit value 3-MAJ$_h(Z(v))$ of the subformula rooted at v

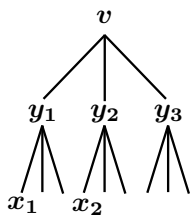
Let $h = h(v)$

{First base case: $h = 0$ (v is a leaf)}
if $h = 0$ **then**
  Query $Z(v)$ to get its value a
  return a
end if

Let y_1, y_2, y_3 be a uniformly random permutation of the children of v

{Second base case: $h = 1$}
if $h = 1$ **then**
  EVALUATE(y_1) and EVALUATE(y_2)
  if $y_1 = y_2$ **then**
    return y_1
  else
    return EVALUATE(y_3)
  end if
end if

{Recursive case}
Let x_1 and x_2 be chosen uniformly at random from the children of y_1 and y_2, respectively
{use the attached figure as a guide}
EVALUATE(x_1) and EVALUATE(x_2)



if $x_1 \neq x_2$ **then**
  EVALUATE(y_3)
  Let $b \in \{1, 2\}$ be such that $x_b = y_3$
  COMPLETE(y_b, x_b)
  if $y_b = y_3$ **then**
    return y_b
  else
    return COMPLETE(y_{3-b}, x_{3-b})
  end if
else $\{x_1 = x_2\}$
  COMPLETE(y_1, x_1)
  if $y_1 = x_1$ **then**
    COMPLETE(y_2, x_2)
    if $y_2 = x_2$ **then** $\{y_2 = y_1\}$
      return y_1
    else $\{y_2 \neq y_1\}$
      return EVALUATE(y_3)
    end if
  else $\{y_1 \neq x_1\}$
    EVALUATE(y_3)
    if $y_3 = y_1$ **then**
      return y_1
    else
      return COMPLETE(y_2, x_2)
    end if
  end if
end if

Algorithm 3 COMPLETE(v, y_1): finish the evaluation of the subformula rooted at node v

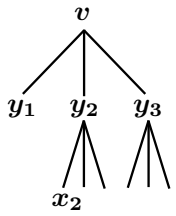
Input: Node v of height $h(v)$; child y_1 of v which has already been evaluated
Output: the bit value 3-MAJ$_h(Z(v))$

Let $h = h(v)$

Let y_2, y_3 be a uniformly random permutation of the two children of v other than y_1

{Base case}
if $h = 1$ **then**
  EVALUATE(y_2)
  if $y_2 = y_1$ **then**
    return y_1
  else
    return EVALUATE(y_3)
  end if
end if

{Recursive case}
Let x_2 be chosen uniformly at random from the children of y_2
{use the attached figure as a guide}
EVALUATE(x_2)



if $y_1 \neq x_2$ **then**
  EVALUATE(y_3)
  if $y_1 = y_3$ **then**
    return y_1
  else
    return COMPLETE(y_2, x_2)
  end if
else $\{y_1 = x_2\}$
  EVALUATE(y_2, x_2)
  if $y_1 = y_2$ **then**
    return y_1
  else
    return EVALUATE(y_3)
  end if
end if

ECCC ISSN 1433-8092
http://eccc.hpi-web.de