Randomized versus Deterministic Decision Tree Size

Arkadev Chattopadhyay
Tata Institute of Fundamental Research
Mumbai, India
arkadev.c@tifr.res.in

Yogesh Dahiya
IMSc
Chennai, India
yogeshdahiya@imsc.res.is

Nikhil S. Mande
QuSoft and CWI
Amsterdam, The Netherlands
nikhil.s.mande@gmail.com

Jaikumar Radhakrishnan
TIFR, Mumbai, and
ICTS–TIFR, Bengaluru
India
jaikumar@tifr.res.in

Swagato Sanyal
IIT Kharagpur
India
swagato@cse.iitkgp.ac.in

ABSTRACT

A classic result of Nisan [SICOMP '91] states that the deterministic decision tree *depth* complexity of every total Boolean function is at most the cube of its randomized decision tree *depth* complexity. The question whether randomness helps in significantly reducing the *size* of decision trees appears not to have been addressed. We show that the logarithm of the deterministic decision tree size complexity of every total Boolean function on n input variables is at most the fourth power of the logarithm of its bounded-error randomized decision tree size complexity, ignoring a polylogarithmic factor in the input size. Our result has the following consequences: (1) The deterministic AND-OR query complexity of a total Boolean function is at most the fourth power of its randomized AND-OR query complexity, ignoring a polylog n factor, (2) The deterministic AND (OR) query complexity of a total Boolean function is at most the cube of its randomized AND (OR) query complexity, ignoring a polylog n factor. This answers a recent open question posed by Knop, Lovett, McGuire and Yuan [SIGACT News '21], (3) The notion of *rank* of a Boolean function was defined in a classic work of Ehrenfeucht and Haussler [Information and Computation'89] in the context of learning theory, and is characterized by the logarithm of decision tree size up to a logarithmic factor in the input size. Our results confirm a recent conjecture (ignoring a polylog n factor) of Cornelissen, Mande and Patro [FSTTCS '22], that asserted the equivalence of randomized and deterministic analogs of rank, upto polynomial factors, for all total Boolean functions, and (4) Combined with the above-mentioned work of Ehrenfeucht and Haussler, our result implies that the class of functions computable by randomized decision trees of polynomial size, is PAC-learnable in quasi-polynomial time. To obtain our main result on decision tree size, we use as an intermediate measure the *block number* of a Boolean function, studied first by Kulkarni and Tal [CJTCS'16], which can be thought of as a counting analog of *block sensitivity* of a Boolean function that played a central role in Nisan's result mentioned above.

CCS CONCEPTS

• **Theory of computation → Oracles and decision trees**; **Probabilistic computation**.

KEYWORDS

Boolean functions, derandomization, query complexity

ACM Reference Format:
Arkadev Chattopadhyay, Yogesh Dahiya, Nikhil S. Mande, Jaikumar Radhakrishnan, and Swagato Sanyal. 2023. Randomized versus Deterministic Decision Tree Size. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC '23), June 20–23, 2023, Orlando, FL, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564246.3585199

1 INTRODUCTION

Understanding the power of randomness is of central interest in computer science. There are two main resources for algorithms: time and space. The role randomness plays in reducing requirements of time and space is under intensive investigation, but a resolution seems quite out of reach of current techniques.

Decision trees are one of the most basic models of computation that has been studied for a long time. How much does randomness help algorithms in this model?

The two natural complexity measures of a decision tree are its depth, also called height, and its size. While depth has most probably been looked at more intensively, size also arises often, especially in the context of learning theory (see for example [7, 8, 17]). Small depth directly implies small size, i.e., a depth t decision tree has size at most 2^t. But the converse is well known to be false, as witnessed by the AND or OR function that can be computed with size $(n+1)$ but need depth also to be n. A classic work of Nisan [35], first published in the late eighties, showed that deterministic decision tree depth complexity is at most a cube of its randomized counterpart, for every total function.[1] The question about the extent of possible savings from the use of randomness in terms of the size of decision trees is natural. It remained unaddressed, somewhat surprisingly, since Nisan's work. It can be inferred from long-existing results that some savings is indeed possible. For instance, consider the function f

[1]Whether this cubic gap can be narrowed down further, is an area of active research with relatively recent breakthroughs [1, 34].

known as the AND-OR tree of size n. This function is represented by a monotone formula of depth $\log n$, with binary AND and OR gates alternating in the layers. Saks and Wigderson [37] showed that the randomized depth complexity of f is $\Theta(n^{0.753\cdots})$ which implies that the randomized size complexity of f is $2^{O(n^{0.753\cdots})}$. Snir's [38] early and elegant work already implies that the deterministic decision tree size of f is $2^{\Theta(n)}$. This was later rediscovered by Jukna, Razborov, Savický and Wegener [28] via a spectral technique. While this gap may appear significant, in the log-scale the gap is still polynomial, i.e.,

$$\log(\mathrm{DSize}^{\mathrm{dt}}(f)) \geq (\log(\mathrm{RSize}^{\mathrm{dt}}(f))^{1.32\ldots},$$ [2]

where $\mathrm{DSize}^{\mathrm{dt}}(f)$ and $\mathrm{RSize}^{\mathrm{dt}}(f)$ denote deterministic and bounded-error randomized decision tree complexity of f, respectively. In the log-scale, the largest known gap between randomized and deterministic size is quadratic (see Section C). Analogous to depth complexity, one may ask if the gap remains polynomial for every total function. Our main result answers this in the affirmative, ignoring polylog(n) factors.

THEOREM 1.1. *For every total Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$\log \mathrm{DSize}^{\mathrm{dt}}(f) = O((\log \mathrm{RSize}^{\mathrm{dt}}(f))^4 \log^3(n)).$$

1.1 Consequences

Theorem 1.1 complements Nisan's derandomization of decision tree depth. In this section, we describe three consequences of our main result, the first two of which are derandomization of depth in stronger models of decision trees. In such models, internal nodes of a tree are allowed to evaluate any function of the input bits from a given class of functions. The class of allowed functions defines the model. One such model is called Parity decision tree (PDT), where the class of allowed functions is just the set of all possible parities of the n input bits[3] PDT's have gained significant interest, see for example [13, 19, 24, 40]. Interestingly, it is well known that randomness can provide largest possible savings in depth (and size) for PDT's. A simple adversary-based argument shows that deterministic PDTs need depth n to compute the OR of n bits. However, randomized PDTs compute OR with bounded error in just $O(1)$ depth.

In a recent interesting survey, Knop, Lovett, McGuire and Yuan [29] advocate looking at other natural classes of allowed functions, like AND and OR, besides PARITY. They mainly argue that such models act as natural and insightful intermediate models between ordinary decision trees and Yao's 2-party communication model. The latter is of great independent interest, and can also be naturally thought of as a decision tree, known as a protocol tree, with each node evaluating an arbitrary function that depends on just Alice's or Bob's input bits. In particular, if there exists a PDT or AND/OR decision tree evaluating a function f in small depth, then there exists a protocol tree of at most twice the depth for computing the same function, irrespective of the partition of the input bits between the players.

The study of these models of decision trees also has well-known connections to combinatorial group testing. The natural quantum analog of AND decision trees has been studied in this context [3] with surprising applications in proving classical lower bounds [5]. In another direction, PDTs (and more generally parity decision DAGs) are closely related to proof systems that allow resolution over equations in $\mathbb{F}_2$, known as Res-Lin that is a generalization of ordinary resolution of clauses (see [36]). AND (OR) decision trees are a special case of linear threshold decision trees that are closely connected to the cutting planes proof system (see [25]). Both these proof systems are poorly understood [18, 26] and it is believed that a better understanding of PDT's and threshold decision trees, especially w.r.t. search problems, should lead to progress here.

1.1.1 (AND, OR)-decision trees. The two fundamental functions that are hard for decision tree depth are AND and OR, which are two of the most basic Boolean functions. It is thus natural to look at decision trees where query nodes can evaluate AND's or OR's of arbitrary subsets of input bits. Note that the model is equivalent up to a factor of 2 in query cost, to the model where AND (OR) alone is allowed, but over *literals* (i.e., possibly negated variables) and not just positive variables. Such decision trees have been studied for long, for example in the early work of Ben-Asher and Newman [4] and more recently in the context of graph problems, see for example [6, 9].

Observing that (AND, OR)-decision tree depth is equivalent to logarithm of ordinary decision tree size after ignoring polylogarithmic factors in the input size (see Lemma 4.3 and Lemma 4.4), we conclude from Theorem 1.1 that the randomized and deterministic depth in this model, denoted respectively by $\mathrm{R}^{(\wedge,\vee)\text{-dt}}(\cdot)$ and $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\cdot)$, are polynomially related as stated below formally.

THEOREM 1.2. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$\mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) = O(\mathrm{R}^{(\wedge,\vee)\text{-dt}}(f)^4 \log^7(n)).$$

1.1.2 AND-decision trees. A recent technique that has generated a lot of insight and solved several longstanding problems in two-party communication complexity is that of *lifting* decision tree depth complexity of a function/problem f to communication complexity of a composed problem. In this setting, each input bit of f is distributed among Alice and Bob via a gadget $g : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$. When the gadget g has a nice (obfuscating) property and its size b is appropriately large, lifting theorems assert that the randomized (deterministic) communication complexity of the composed problem is asymptotically $\mathrm{R}^{\mathrm{dt}}(f) \cdot \mathrm{R}^{\mathrm{cc}}(g)$ ($\mathrm{D}^{\mathrm{dt}}(f) \cdot \mathrm{D}^{\mathrm{cc}}(g)$). Observe that this corresponds roughly to the communication cost of the naive protocol that simulates the optimal decision tree algorithm for f, solving the i-th instance of g when the decision tree algorithm stipulates querying its i-th input bit. These theorems are attractive as they reduce the task of understanding communication complexity to that of decision tree depth complexity, the latter usually being a lot simpler. Several such theorems have been developed, see for example [12, 14, 21, 22, 24, 29], exploiting convenient properties of the gadget g. A well-known challenge in this area is to reduce the size of the gadget to a constant (see for example [33]), independent of the input length of f, and still prove such lifting theorems. There are essentially two one-bit gadgets, XOR and AND (OR).

[2]We observe in Corollary A.4 that the AND-OR tree does not witness a stronger separation than this.
[3]Note that this is a generalization of ordinary decision trees as they can be simulated by merely evaluating the parity of just the relevant bit.

It is not difficult to see that when Xor is the gadget, then a communication protocol can simulate a PDT for f, a more powerful model than ordinary decision trees. This explains why the Eq function, a composition of And and Xor, has small randomized communication complexity: And has an $O(1)$-depth randomized PDT. For the Xor gadget, a deterministic lifting theorem[4] was relatively recently developed by Hatami, Hosseini and Lovett [24]. No randomized lifting theorem is known for the Xor gadget.

For the And gadget, similarly, a communication protocol can simulate an And-decision tree (ADT). Very recently, Knop, Lovett, McGuire and Yuan [30] lifted deterministic ADT depth complexity of f to deterministic communication complexity of $f \circ$ And. They left open the problem of proving a randomized lifting theorem. In a follow-up survey [29] it was argued that this inability is linked to our lack of understanding of basic questions about randomized ADT complexity. They conjectured, however, that the randomized and deterministic ADT depth complexity of total functions are polynomially related. Using a key inequality we establish in the proof of our main result, Theorem 1.1, we are able to confirm this conjecture as stated below.

Theorem 1.3. *For every Boolean function* $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{D}^{\wedge\text{-dt}}(f) = O(\mathrm{R}^{\wedge\text{-dt}}(f)^3 \log^4(n))$.

En route to proving the above, we establish a characterization, up to polynomial gap, of the randomized ADT depth complexity in terms of natural combinatorial quantities associated with a Boolean function. At an input $x \in \{0,1\}^n$, a block $B \subseteq [n]$, is called a sensitive 0-block if $x_i = 0$ for every $i \in B$, and $f(x) \neq f(x \oplus 1_B)$, where $x \oplus 1_B$ is obtained from x by switching values of the bits in B to 1. A set of indices $H \subseteq [n]$ is called a 0-hitting set for f at x if it intersects every sensitive 0-block of x. The 0-hitting set complexity of f, denoted by $\mathrm{HSC}_0(f)$, is the smallest number r, such that for every input $x \in \{0,1\}^n$, there exists a 0-hitting set of size at most r. A natural fractional relaxation leads to a quantity called the 0-fractional hitting set complexity, denoted by $\mathrm{FHSC}_0(f)$. We also consider the zero (one) cover number of f, denoted by $\mathrm{N}_0(f)$ ($\mathrm{N}_1(f)$), which is the minimum number of monochromatic subcubes needed to cover $f^{-1}(0)$ ($f^{-1}(1)$). The cover number of f, denoted by $\mathrm{N}(f)$, is defined to be $\mathrm{N}_0(f)+\mathrm{N}_1(f)$. We show that $\log \mathrm{N}(f)$ is bounded from above by $\widetilde{O}(\mathrm{R}^{\wedge\text{-dt}}(f)^2)$,[5] and that $\mathrm{FHSC}_0(f) = O(\mathrm{R}^{\wedge\text{-dt}}(f))$ (see Proof of Theorem 4.5 for proofs of these inequalities). Finally we also show that $\mathrm{D}^{\wedge\text{-dt}}(f) = \widetilde{O}(\mathrm{FHSC}_0(f) \log \mathrm{N}(f))$ (Claim 4.7). Combining these together, we obtain the following combinatorial characterization of deterministic and randomized ADT depth complexity.

Theorem 1.4. *For every Boolean function* $f : \{0,1\}^n \to \{0,1\}$,

$$\max\left\{\Omega(\mathrm{FHSC}_0(f)), \widetilde{\Omega}(\log \mathrm{N}(f)^{1/2})\right\} = \mathrm{R}^{\wedge\text{-dt}}(f)$$
$$\leq \mathrm{D}^{\wedge\text{-dt}}(f) = O(\mathrm{FHSC}_0(f) \cdot \log \mathrm{N}(f)).$$

[4]Ideally, lifting theorems should work even when f is not only a total function but just a partial function or a relation. Many applications, particularly in circuit and proof complexity, require this generality of a lifting theorem. There are no such general lifting theorems known with a constant-size gadget.

[5]Throughout this paper we use notations $\widetilde{O}(\cdot), \widetilde{\Theta}(\cdot), \widetilde{\Omega}(\cdot)$ to hide polylogarithmic factors in the input size.

Theorem 1.4 says that a function is hard for randomized ADTs if and only if its cover number is large or it has a large 0-fractional hitting set complexity. We note below two simple functions, one of which has large cover number and small 0-fractional hitting set complexity, and the other has small cover number but large 0-fractional hitting set complexity.

- $f = \mathrm{Or}_n$: $\log \mathrm{N}(f) = O(\log n), \mathrm{FHSC}_0(f) = n, \mathrm{R}^{\wedge\text{-dt}}(f) = \Omega(n)$.
- $f = \mathrm{And}_n \circ \mathrm{Or}_2$: $\log \mathrm{N}(f) = \Omega(n), \mathrm{FHSC}_0(f) = 2, \mathrm{R}^{\wedge\text{-dt}}(f) = \widetilde{\Omega}(n)$.

On the other hand, the And function has small cover number, as well as small 0-fractional hitting set complexity. This is not surprising as $\mathrm{D}^{\wedge\text{-dt}}(\mathrm{And}) = 1$.

1.1.3 Decision tree rank. Another measure of decision trees that has attracted interest recently is its decision tree *rank* [15, 16], which measures the tree's branching complexity; it is defined to be the height of the largest complete binary tree that can be embedded in the given tree. This notion was introduced by Ehrenfeucht and Haussler [17] in the context of learning theory. The rank of a Boolean function f is defined as the minimum rank of a decision tree computing f. One can show that rank characterizes the logarithm of decision tree size up to a logarithmic factor (see [16, Proposition 2.5], for example). An analogous statement is easily seen to hold true for randomized decision tree size and the natural randomized variant of rank. Cornelissen, Mande and Patro [15] conjectured that rank and randomized rank are polynomially related for all total Boolean functions. By the equivalence of rank with the logarithm of decision tree size mentioned above, Theorem 1.1 resolves this conjecture in the affirmative up to a polylog n factor. Ehrenfeucht and Haussler showed that the class of functions on n variables with decision tree size polynomial in n is learnable from random examples in quasi-polynomial time. An immediate consequence of Theorem 1.1 is that even the class of functions on n variables with *randomized* decision tree size polynomial in n is learnable from random examples in quasi-polynomial time.

1.2 Our Techniques

As far as we are aware, the only previous work achieving full derandomization for a model of decision trees is due to Nisan's [35], where he used beautiful argument by defining the *block sensitivity* of a Boolean function f. The block sensitivity of f at an input x is the maximum number of disjoint sensitive blocks of f at x. Another key notion required in Nisan's argument is that of *certificate complexity*. The certificate complexity of f at x is the minimum number of bits of x one needs to reveal to force f to a constant. The certificate complexity of f is the maximum certificate complexity over all inputs. Nisan's argument is completed essentially[6] by two facts: first, the certificate complexity of f at any input x is bounded from above by the square of the block sensitivity of f. Using the older known fact, re-discovered by many authors in different contexts [10, 23, 39], showing that the square of the certificate complexity is an upper bound for deterministic depth, yields the fourth power of block sensitivity as an upper bound on the deterministic depth

[6]Here, for simplicity, we sketch the ideas achieving a fourth power gap, and not the cubic gap that Nisan proved.

of f. Second, Nisan proved that block sensitivity is a valid lower bound on the randomized depth complexity of f.[7] This is simply because f embeds at x a promised Or of arity as much as its block sensitivity at x.

Block sensitivity is a very influential notion, but is not helpful for us in proving randomized decision tree size lower bounds. For example, Or and And have the largest possible block sensitivity, and even sensitivity, of n and yet have deterministic decision tree size merely $O(n)$. More generally, any measure certified by specifying the evaluation of f at s points of the Boolean cube can yield at most a lower bound of s on decision tree size. This is because there always exists a deterministic decision tree of size s that computes f restricted to those s points, as we describe now. We will construct a tree such that each leaf corresponds to a unique point amongst the s given points. Start with a single node, and as long as there exists a leaf ℓ which is reached by two distinct points $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$, query a variable i such that $x_i \neq y_i$ at ℓ. This way, ℓ is replaced by two other leaves, and x and y are separated. Continuing this way, we eventually end up with a tree with exactly s leaves, one for each point. We may then label the leaves with the function values of the corresponding points.

Consider the Majority function, at the all zero point. Its block sensitivity there is just 2. However, there are exponentially many, i.e., $\binom{n}{n/2}$, minimally sensitive blocks. These minimal blocks form an anti-chain with respect to inclusion. Further, their minimality implies[8] that each point in the cube obtained by flipping the bits of such a block yields a point where every (flipped) bit is sensitive. Our argument has two parts.

- First, we show (Lemma 3.2) that any large set S of such points, by itself, ensures that the randomized decision tree size is $\widetilde{\Omega}(|S|)$. This holds more generally. Hence, a randomized decision tree computing a function f that has a large number of minimally sensitive blocks at some input x will be forced to have a large size. To formalize this, we define the *block number* of a function f as the largest integer r for which there exists a point x where f has r minimally sensitive blocks. To the best of our knowledge, this notion was first studied by Kulkarni and Tal [31] to show relationships between different query complexity measures.
- In the second part of our argument, we deal with functions that have small block number. These are functions f, which at every point x, have *few* minimally sensitive blocks. An example of such an f with large randomized size complexity is Parity. In the second part of the argument (Lemma 3.3), we prove, using a simple boosting argument that for such an f, one would be able to extract a cover for both the ones and zeroes of f by monochromatic subcubes, using about as many subcubes as the size of the randomized decision tree. This is sufficient to construct a deterministic decision tree for f of not too large size, thanks to a classical result of Ehrenfeucht and Haussler [17] from the late eighties. This result states that there exists a deterministic decision tree of size at most $n^{\log^2 \mathrm{N}(f)}$, where $\mathrm{N}(f)$ is the minimal size of a cover of f by monochromatic subcubes.

In particular, combining the above two parts yields the following interesting relationship between the cover number and randomized size complexity, as stated in Theorem 3.1, part (a):

$$\log \mathrm{N}(f) = \widetilde{O}\left(\log^2 \mathrm{RSize}^{\mathrm{dt}}(f)\right). \tag{1}$$

The derandomization of depth for (And, Or)-decision trees follows directly, once we observe that such depth complexity of f is equivalent, ignoring a factor of $\log n$, to the logarithm of its ordinary decision tree size complexity. This equivalence is established using a simple tree-balancing argument that is possible as we have both And and Or queries available.

It requires more work to deal with decision trees that have just And (Or) queries. The main reason is that And queries on unnegated variables are not sufficient to equate depth with the logarithm of ordinary size. This is illustrated by the fact, already mentioned before in Section 1.1.2, that the Or function has ordinary decision tree size only n and, yet, the ADT depth complexity of Or is full, i.e., n. However, as observed by Loff and Mukhopadhyay [32], the ADT depth complexity is equivalent, upto polylog(n) factors, to the zero-depth complexity of ordinary decision trees. The zero-depth complexity of f is the smallest number r for which there exists a decision tree computing f in which every path from the root to a leaf encounters no more than r zeroes. Our main contribution here is an adaptation of Ehrenfeucht and Haussler's argument to prove a convenient deterministic zero-depth upper bound in terms of the fractional zero-hitting set complexity of f and its cover number, as stated in Theorem 1.4. The second important observation is that the fractional zero-hitting set complexity is a lower bound on the randomized zero-depth complexity of f. Noting the fact that a decision tree of zero-depth d is also an ordinary decision tree of $O(n^d)$ size, we complete the derandomization of ADT depth using Equation (1).

We remark here that the notion of fractional zero-hitting set complexity was introduced in [30]. However, the approach taken by the authors in that paper was algebraic, in the sense that they focused on the sparsity of the unique polynomial over the AND basis representing f. In fact, in their later survey [29], they suggested looking at the sparsity needed by polynomials to *approximate* f to tackle randomized ADT depth complexity. Our proof of their conjecture (our Theorem 1.3), however, deviates from this approach completely and involves purely combinatorial arguments.

2 PRELIMINARIES

This section formally introduces the required notation, different query models and various complexity measures used in our proofs.

All logarithms in this paper are taken base 2. We use notations $\widetilde{O}(\cdot), \widetilde{\Theta}(\cdot), \widetilde{\Omega}(\cdot)$ to hide polylogarithmic factors in the input size (and not just polylogarithmic factors in the argument). For a bit $b \in \{0,1\}$, $\bar{b}$ denotes the bit $1-b$. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let $\bar{f} : \{0,1\}^n \to \{0,1\}$ be the Boolean function defined as $\bar{f}(x) = 1 - f(x)$ for all $x \in \{0,1\}^n$. For a positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For $x, y \in \{0,1\}^n$ we say $x \leq y$ if $x_i \leq y_i$ for all $i \in [n]$. A Boolean function f is *monotone* if $x \leq y$

[7]It was later shown [2] that the square root of block sensitivity is also a lower bound on the *quantum* analog of depth complexity (which is better known as quantum query complexity), showing that deterministic depth complexity and quantum query complexity are also polynomially related for all total Boolean functions.

[8]This property of minimally sensitive blocks is a well-known fact that Nisan's argument also exploited.

implies $f(x) \leq f(y)$. A maxterm of a monotone Boolean function, $f : \{0,1\}^n \to \{0,1\}$, is a minimal set of variables $S \subseteq [n]$ such that setting the variables in S to 0 forces f to 0. A minterm of f is a minimal set of variables which, when set to 1 forces f to 1.

2.1 Combinatorial Measures of Boolean Functions

Definition 2.1 (Subcube). *A subcube of dimension d (equivalently, codimension $n-d$) is a subset of $\{0,1\}^n$ obtained by fixing $n-d$ of the variables. In other words, a subcube is a set of all inputs consistent with a partial assignment of n bits. For $b \in \{0,1\}$, a subcube A is said to be b-monochromatic with respect to f if $f(x) = b$ for all x in A. The codimension of a subcube A is denoted by* $\text{co-dim}(A)$.

Definition 2.2 (Cover number). *For a function $f : \{0,1\}^n \to \{0,1\}$ and $b \in \{0,1\}$, we define its b-cover number, denoted by $\mathrm{N}_b(f)$, as*

$$\min\{k : \exists b\text{-monochromatic subcubes wrt } f,\ S_1, \ldots S_k \text{ such that } \bigcup_{i\in[k]} S_i = f^{-1}(b)\}.$$

The cover number of f, denoted by $\mathrm{N}(f)$, is defined to be $\mathrm{N}_0(f) + \mathrm{N}_1(f)$.

For $x \in \{0,1\}^n$ and $B \subseteq [n]$, let 1_B denote the n-bit string that is 1 on bits in B and 0 otherwise.

Definition 2.3 (Sensitive block, block sensitivity). *Let $f : \{0,1\}^n \to \{0,1\}$. A set $B \subseteq [n]$ is a* sensitive block *of f at input x if $f(x \oplus 1_B) \neq f(x)$; a sensitive block at x is* minimal *if no proper subset of it is a sensitive block at x. The block sensitivity of f at x, denoted by $\mathrm{bs}(f,x)$, is the maximum r for which there exist disjoint blocks, $B_1, B_2, \ldots, B_r$, each of which is a sensitive block of f at x. The block sensitivity of f, denoted by $\mathrm{bs}(f)$, is defined as*

$$\mathrm{bs}(f) = \max_{x\in\{0,1\}^n} \mathrm{bs}(f,x).$$

An index $i \in [n]$ is sensitive for x with respect to f if $\{i\}$ is a sensitive block of f at x. For a function $f : \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, let $\mathcal{W}(f,x)$ denote the set of all minimal sensitive blocks of f at x. A set $B \subseteq [n]$ is a *sensitive 0-block of f at x* if $f(x \oplus 1_B) \neq f(x)$ and $x_i = 0$ for each $i \in B$. Let $\mathcal{W}_0(f,x)$ denote the set of all minimal sensitive 0-blocks of f at x.

Next we define hitting set complexity and a new measure which we call *block number*. For sets $S, F \subseteq [n]$, we say that S *hits* F if $S \cap F \neq \emptyset$.

Definition 2.4 (Block number, hitting set complexity). *For a function $f : \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, we define the* block number *of f at x, denoted by $\#\mathrm{bs}(f,x)$, to be the number of elements in $\mathcal{W}(f,x)$; the* block number *of f is*

$$\#\mathrm{bs}(f) = \max_{x\in\{0,1\}^n} \#\mathrm{bs}(f,x).$$

The hitting set complexity *of f at x, denoted by $\mathrm{HSC}(f,x)$, is the minimum size of a set $S \subseteq [n]$ that hits each $F \in \mathcal{W}(f,x)$. The* hitting set complexity *of f, denoted by $\mathrm{HSC}(f)$, is defined as*

$$\mathrm{HSC}(f) = \max_{x\in\{0,1\}^n} \mathrm{HSC}(f,x).$$

We remark here that, to the best of our knowledge, the notion of block number (in a different language) was first studied by Kulkarni and Tal [31].

Next we define 0-hitting set complexity.

Definition 2.5 (0-hitting set complexity, same as [30, Definition 2.10]). *For $f : \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, the* 0-hitting set complexity *of f at x, denoted $\mathrm{HSC}_0(f,x)$, is the minimum size of a set $S \subseteq [n]$ that hits each $F \in \mathcal{W}_0(f,x)$. The* 0-hitting set complexity *of f, denoted by $\mathrm{HSC}_0(f)$, is defined as*

$$\mathrm{HSC}_0(f) = \max_{x\in\{0,1\}^n} \mathrm{HSC}_0(f,x).$$

To study And decision trees (see Section 2.2), Knop, Lovett, McGuire and Yuan [30] considered a fractional version of hitting set complexity, which we define below.

Definition 2.6 (0-fractional hitting set complexity, same as [30, Definition 2.14]). *For $f : \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$, the* 0-fractional hitting set complexity *of f at x, denoted $\mathrm{FHSC}_0(f,x)$, is $1/p$, where $p > 0$ is the largest number such that there exists a distribution $\mathcal{D}$ on indices $i \in [n]$ with the property that $\Pr_{i\sim\mathcal{D}}[i \in F] \geq p$ for each $F \in \mathcal{W}_0(f,x)$. Define $\mathrm{FHSC}_0(f) = \max_{x\in\{0,1\}^n} \mathrm{FHSC}_0(f,x)$.*

Lemma 2.7. *Fix $x \in \{0,1\}^n$, and let $(W_1, W_2, \ldots, W_t) \in \mathcal{W}_0(f,x)$ be a sequence of blocks (not necessarily different). There there is an index j^* such that $|\{\ell : j^* \in W_\ell\}| \geq t\mathrm{FHSC}_0(f)$.*

Proof. Let $\mathrm{FHSC}_0(f) = 1/p$. From Definition 2.6, we obtain a distribution $\mathcal{D}$ on indices in $[n]$ such that $\Pr_{j\sim D}[j \in W] \geq p$, for each $W \in \mathcal{W}_0(f,x)$. By linearity of expectation, $\mathrm{E}_{j\in D}[|\{\ell : j \in W_\ell\}|] \geq pt$, so there must be a choice j^* for which the conclusion of the lemma holds. □

2.2 Decision Trees for Boolean Functions

A deterministic decision tree (DT) on n variables is a binary tree whose internal nodes are labeled by variables and leaves are labeled $\{0,1\}$. Every internal node has a left child and a right child; the edge leading to the left child is labeled 0 and the edge leading to the right child is labeled 1. On an input $x \in \{0,1\}^n$, the tree's computation proceeds from the root down to a leaf as follows. When the computation reaches a certain node, the variable associated with that node is *queried*: if the value obtained is 0, the computation moves to the left child, otherwise it moves to the right child. The output of a decision tree T on input x, denoted by $T(x)$, is the label of leaf node reached by this computation. We say that a decision tree T computes the function $f : \{0,1\}^n \to \{0,1\}$ (or T is a decision tree for f), if $T(x) = f(x)$ for all $x \in \{0,1\}^n$. The deterministic decision tree depth complexity of f, is

$$\mathrm{D}^{\mathrm{dt}}(f) := \min_{T:T \text{ computes } f} \mathrm{Depth}(T).$$

Analogously the deterministic decision tree size complexity of f is

$$\mathrm{DSize}^{\mathrm{dt}}(f) := \min_{T:T \text{ computes } f} \mathrm{Size}(T),$$

where $\mathrm{Size}(T)$ is the number of leaves in T.

A randomized query algorithm/decision tree $\mathcal{A}$ is a distribution $\mathcal{D}_{\mathcal{A}}$ over deterministic decision trees. The computation of $\mathcal{A}$ on input x proceeds by sampling a deterministic decision tree T according to $\mathcal{D}_{\mathcal{A}}$, and outputs the label of the leaf reached by T on x. $\mathcal{A}$ computes f with error at most ε if for every input x, the probability that $A(x) = f(x)$ is at least $1 - \varepsilon$. The cost of the randomized algorithm is measured by the number of worst-case queries made by $\mathcal{A}$ on any input x, i.e., the maximum depth over all decision trees in the support of the distribution. The ε-error randomized decision tree complexity of f, denoted $\mathrm{R}_{\varepsilon}^{\mathrm{dt}}(f)$, is the minimum cost of such a randomized decision tree. The randomized decision tree size of f, denoted by $\mathrm{RSize}_{\varepsilon}^{\mathrm{dt}}(f)$, is defined similarly. That is,

$$\mathrm{RSize}_{\varepsilon}^{\mathrm{dt}}(f) = \min_{\substack{\mathcal{A} \text{ computes } f \\ \text{with error } \leq \varepsilon}} \max_{T : \mathcal{D}_{\mathcal{A}}(T) > 0} \mathrm{Size}(T).$$

Each internal node of a standard decision tree corresponds to a variable being queried. Generalizing the class of permitted queries gives rise to many variants of decision trees that have been considered in different contexts. In each variant, the relevant deterministic and randomized decision tree complexities are defined just as above.

In our work, we are interested in three such classes.

- When the permitted queries are AND of (non-negated) variables, we refer to such trees as AND-*decision trees (ADT)*. We let $\mathrm{D}^{\wedge\text{-dt}}(\cdot)$ and $\mathrm{R}_{\varepsilon}^{\wedge\text{-dt}}(\cdot)$ denote the deterministic and ε-error randomized decision tree complexities, respectively, in this model.
- When the permitted queries are AND of a subset of variables (AND query) or an OR of a subset of variables (OR query), we refer to such trees as (AND, OR)-*decision trees ((*AND, OR*)-DT)*. We let $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\cdot)$ and $\mathrm{R}_{\varepsilon}^{(\wedge,\vee)\text{-dt}}(\cdot)$ denote the deterministic and ε-error randomized decision tree complexities, respectively, in this model.
- When the permitted queries are just variables, but the depth of a path is measured by the number of queries answered as 0 on the path, we refer to deterministic and ε-error randomized decision tree complexities as $\mathrm{D}^{0\text{-dt}}(\cdot)$ and $\mathrm{R}_{\varepsilon}^{0\text{-dt}}(\cdot)$, respectively. We use the term '0-depth' of a decision tree to refer to the longest root-to-leaf path where only 0-labelled edges contribute to the length.

Throughout this paper, when we drop ε from the subscript of a randomized decision tree (possibly with generalized queries as above) measure, we assume $\varepsilon = 1/3$.

DEFINITION 2.8 (QUERY TREE, QUERY SET, TRANSCRIPT). *A deterministic query tree on n variables is a rooted binary tree where every node is labelled by a variable. Unlike a decision tree, a query tree does not return the value of any function at the leaves. (It represents a strategy for querying variables with the goal of finding a certificate.) By dropping its leaves, we regard a decision tree for a function as a query tree. A randomized query tree is modelled as a distribution over deterministic query trees. For a query tree T and an input $x \in \{0,1\}^n$, let $Q_T(x)$ be the set of indices of variables queried by T on input x; the transcript of T on input x is the sequence $\sigma_T(x) = (\langle i_t, x[i_t]\rangle : t = 1, 2, \ldots)$, where i_t is the index of the input bit of x probed by T in the t-th step. For a randomized query tree T, let $T^{\otimes \ell}$ be the randomized query tree corresponding to performing ℓ independent computations of the query tree T. We use $\mathrm{Size}(T)$ to denote the number of distinct transcripts of T over all inputs $x \in \{0,1\}^n$.*

We define the notion of rank of a decision tree introduced by Ehrenfeucht and Haussler [17]. We also define randomized rank, a natural randomized variant of the same studied by Cornelissen, Mande and Patro [15].

DEFINITION 2.9 (DECISION TREE RANK AND RANDOMIZED RANK). *Let T be a binary decision tree. Define the rank of T recursively as follows. For a leaf node a, let $\mathsf{rank}(a) = 0$. For an internal node u with children v, w, let*

$$\mathsf{rank}(u) = \begin{cases} \max\{\mathsf{rank}(v), \mathsf{rank}(w)\} & \text{if } \mathsf{rank}(v) \neq \mathsf{rank}(w) \\ \mathsf{rank}(v) + 1 & \text{if } \mathsf{rank}(v) = \mathsf{rank}(w). \end{cases}$$

Define $\mathsf{rank}(T)$ to be the rank of the root of T. Define the randomized rank *of a randomized decision tree to be the maximum rank of a deterministic decision tree in its support.*

DEFINITION 2.10 (RANK OF A BOOLEAN FUNCTION). *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, define the rank of f, which we denote by $\mathsf{rank}(f)$, by $\mathsf{rank}(f) = \min_{T : T \text{ computes } f} \mathsf{rank}(T)$. Analogously define the randomized rank of f, which we denote by $\mathsf{rrank}(f)$, to be the minimum randomized rank of a randomized decision tree that computes f to error $1/3$.*

The following rank and size relationship is known for Boolean functions.

PROPOSITION 2.11 ([17, LEMMA 1]). *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$\mathsf{rank}(f) \leq \log \mathrm{DSize}^{\mathrm{dt}}(f) \leq \mathsf{rank}(f) \log\left(\frac{en}{\mathsf{rank}(f)}\right),$$

$$\mathsf{rrank}(f) \leq \log \mathrm{RSize}^{\mathrm{dt}}(f) \leq \mathsf{rrank}(f) \log\left(\frac{en}{\mathsf{rrank}(f)}\right)$$

Ehrenfeucht and Haussler [17] only deal with deterministic trees. The analogous result holds for randomized trees because we may apply the relevant deterministic result to all trees in the support of the given randomized tree.

2.3 Required Results

This section contains some required results that give bounds on some combinatorial measures and on certain decision tree measures of Boolean functions defined in the earlier two subsections.

Cover number gives a lower bound on deterministic decision tree size. Each leaf of a decision tree corresponds to a monochromatic subcube generated by the partial assignments defined by the root to leaf path. Thus a deterministic decision tree computing a function f induces a monochromatic subcube partition of f, giving the following.

PROPOSITION 2.12. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{DSize}^{\mathrm{dt}}(f) \geq \mathrm{N}(f)$.*

In the other direction, Ehrenfeucht and Haussler [17, Lemma 1, Lemma 6] gave the following upper bound on the logarithm of the decision tree size of a function f. The version we state below can be found in [27, Theorem 14.32], for example.

THEOREM 2.13 ([27, THEOREM 14.32]). *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\log \mathrm{DSize}^{\mathrm{dt}}(f) = O(\log^2 \mathrm{N}(f) \cdot \log n)$.*

Lemma 2.14. *Suppose $f : \{0,1\}^n \to \{0,1\}$ and $y \in \{0,1\}^n$. Suppose T is an ε-error randomized decision tree for f, $B \in \mathcal{W}(f,y)$ is a sensitive block of f at y, and $W \subseteq \mathcal{W}(f,y)$ is a set of sensitive blocks of f at y. Then,*

(a) $\Pr[Q_T(y) \cap B = \emptyset] \le 2\varepsilon$.
(b) $\Pr[Q_{T^{\otimes \ell}}(y) \cap B = \emptyset] \le (2\varepsilon)^\ell$.
(c) $\Pr[Q_{T^{\otimes \ell}}(y)$ *is a hitting set for* $W] \ge 1 - (2\varepsilon)^\ell |W|$.

Proof. For part (a), we have

$$\begin{aligned} 1 &= \Pr[f(y) \ne f(y \oplus 1_B)] \\ &\le \Pr[f(y) \ne T(y)] + \Pr[T(y) \ne T(y \oplus 1_B)] \\ &\quad + \Pr[T(y \oplus 1_B) \ne f(y \oplus 1_B)] \\ &\le 2\varepsilon + \Pr[T(y) \ne T(y \oplus 1_B)]. \end{aligned}$$

Thus, $\Pr[T(y) \ne T(y \oplus 1_B)] \ge 1 - 2\varepsilon$. Then,

$$\begin{aligned} \Pr[Q_T(y) \cap B = \emptyset] &\le \Pr[T(y) = T(y \oplus 1_B)] \\ &= 1 - \Pr[T(y) \ne T(y \oplus 1_B)] \le 2\varepsilon. \end{aligned}$$

Part (b) follows from part (a) because $T^{\otimes \ell}$ corresponds to ℓ independent computations performed on T; in particular, $Q_{T^{\otimes \ell}}(y) = Q_1 \cup Q_2 \cup \cdots \cup Q_\ell$, where the Q_i are independent and each has the same distribution as $Q_T(y)$. By a union bound over all $B \in W$, we have $\Pr[\exists B \in W : Q_{T^{\otimes \ell}}(y) \cap B = \emptyset] \le (2\varepsilon)^\ell |W|$. Part (c) immediately follows. □

Proposition 2.15. *$Q \subseteq [n]$ contains a certificate for f at x iff Q intersects every block in $\mathcal{W}(f,x)$.*

Proof. Suppose Q contains a certificate for f at x. If Q does not intersect a block B in $\mathcal{W}(f,x)$, then $y = x \oplus 1_B$ and x agree on all the positions in Q but $f(x) \ne f(y)$—a contradiction; thus Q intersects every block in $\mathcal{W}(f,x)$. Next suppose Q intersects every block in $\mathcal{W}(f,x)$. Suppose for some y, we have $f(x) \ne f(y)$; we will show that x and y differ in some position in Q. Now, $x \oplus y$ is a characteristic vector of a sensitive block of f at x, and must include in it a block in $\mathcal{W}(f,x)$; Q intersects this block, so x and y differ in a position in Q. □

And query complexity is tightly related to 0-depth query complexity, which was introduced by Loff and Mukhopadhyay [32].

Claim 2.16 ([32], [30, Claim 4.4]). *For every Boolean function f, we have $\mathrm{D}^{0\text{-dt}}(f) \le \mathrm{D}^{\wedge\text{-dt}}(f) \le \mathrm{D}^{0\text{-dt}}(f)\log n$ and $\mathrm{R}^{0\text{-dt}}(f) \le \mathrm{R}^{\wedge\text{-dt}}(f) \le \mathrm{R}^{0\text{-dt}}(f)\log n$.*

While Loff and Mukhopadhyay [32], and Knop, Lovett, McGuire and Yuan [30] only deal with deterministic trees, the analogous result for randomized trees can easily be seen to hold true by applying the relevant deterministic result to all trees in the support of the relevant randomized tree.

3 RANDOMIZED VERSUS DETERMINISTIC SIZE COMPLEXITY

In this section, we establish upper bounds for $\mathrm{N}(f)$ and $\mathrm{DSize}^{\mathrm{dt}}(f)$ in terms of $\mathrm{RSize}^{\mathrm{dt}}(f)$. We present these upper bounds in Theorem 3.1 below. In the next section, we will make use of part (a) of this theorem to obtain similar results for decision trees with (And,Or) queries and And queries.

Theorem 3.1 (Main result). *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$, we have the following.*

(a) $\log \mathrm{N}(f) = O((\log \mathrm{RSize}^{\mathrm{dt}}(f))^2 \log(n))$.
(b) $\log \mathrm{DSize}^{\mathrm{dt}}(f) = O((\log \mathrm{RSize}^{\mathrm{dt}}(f))^4 \log^3(n))$.

As we stated in the introduction, the block number of a Boolean function plays a central role in our argument. In the following two lemmas the block number appears first in the lower bound and then in an upper bound.

Lemma 3.2. *Let $f : \{0,1\}^n \to \{0,1\}$. Then, for all $x \in \{0,1\}^n$, we have $\left(\frac{1}{2}\right) \#\mathrm{bs}(f,x) \le \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f)^{2\log n}$.*

Lemma 3.3. *Let $f : \{0,1\}^n \to \{0,1\}$. Then,*

$$\mathrm{N}(f) \le n \cdot \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f)^{2\log \#\mathrm{bs}(f)}.$$

Before justifying these lemmas formally, we first verify that our main result follows from them.

Proof of Theorem 3.1. The theorem clearly holds if f is a constant functions. So we assume that f is not constant; in particular, we assume that $\mathrm{RSize}^{\mathrm{dt}}_{1/3}(f) \ge 2$. From Lemma 3.3 and Lemma 3.2 we obtain

$$\begin{aligned} \log \mathrm{N}(f) &\le \log n + \log \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f) \cdot 2\log \#\mathrm{bs}(f); \\ &\le \log n + 2\log \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f)(\log \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f) \cdot 2\log n + 1) \\ &= O((\log \mathrm{RSize}^{\mathrm{dt}}_{1/3}(f))^2 \cdot \log n). \end{aligned}$$

Part (b) follows by combining part (a) with Theorem 2.13. □

We now prove the lemmas.

Proof of Lemma 3.2. Let T be an $(1/3)$-error randomized decision tree for f. Fix $x \in \{0,1\}^n$. For $B \in \mathcal{W}(f,x)$, let $y_B = x \oplus 1_B$. Since B is minimal, we have $f(y_B) \ne f(y_B \oplus 1_{\{i\}})$ for all $i \in B$; that is, $\{i\} \in \mathcal{W}(f, y_B)$. With $\ell = \lfloor 2\log n \rfloor$ (assume n is large) and $W = \{\{i\} : i \in B\}$, we conclude from Lemma 2.14 (c) that

$$\Pr[Q_{T^{\otimes \ell}}(y_B) \supseteq B] \ge 1 - \left(\frac{2}{3}\right)^\ell |B| \ge 1 - \left(\frac{1}{2n}\right)|B| \ge \frac{1}{2}.$$

Thus, using linearity of expectation, we may fix a deterministic tree T^* in the support of $T^{\otimes \ell}$ such that for at least half of the $\#\mathrm{bs}(f,x)$ choices for B, we have $Q_{T^*}(y_B) \supseteq B$; let W' be the set of these B. For each $B \in W'$, let $\sigma_B = \sigma_{T^*}(y_B)$ be the transcript of T^* on input y_B. We make two remarks.

(a) Since the blocks in $\mathcal{W}(f,x)$ are minimal, for distinct $B, B' \in \mathcal{W}(f,x)$, there is an $i \in B \setminus B'$; for this i, $y_B[i] = x[i] + 1 \ne x[i] = y_{B'}[i] \pmod 2$. (There is a similar $i' \in B' \setminus B$, but we won't need it.) Thus, for distinct $B, B' \in W'$, we have $\sigma_B \ne \sigma'_B$.
(b) Since T^* is obtained by picking ℓ trees in the support of T and running them one after the other, the total number of possibilities for σ_B is at most $\mathrm{Size}(T)^\ell$.

Since T was an arbitrary $(\frac{1}{3})$-error decision tree for f, we conclude from the above remarks that $\mathrm{RSize}^{\mathrm{dt}}_{1/3}(f)^{2\log n} \ge |W'| \ge \frac{1}{2}\#\mathrm{bs}(f,x)$. □

Proof of Lemma 3.3. Let T be a $(\frac{1}{3})$-error randomized decision tree for f. Fix $x \in \{0,1\}^n$, and let $W = \mathcal{W}(f,x)$. Recall from Proposition 2.15 that a set Q contains a certificate for f at x iff Q intersects every set in $\mathcal{W}(f,x)$. Now, set $\ell = 2\log \#\mathrm{bs}(f)$ and conclude from Lemma 2.14 (c) that

$$\begin{aligned}&\Pr[Q_{T^{\otimes\ell}}(x) \text{ contains a certificate for } f \text{ at } x]\\ =&\Pr[Q_{T^{\otimes\ell}}(x) \text{ is a hitting set for } \mathcal{W}(f,x)]\\ \geq&\ 1-\left(\frac{2}{3}\right)^{\ell}\#\mathrm{bs}(f) > \frac{1}{2}.\end{aligned}$$

Let $T_1, T_2, \ldots, T_n$ be n trees each picked with the same distribution as $T^{\otimes\ell}$. Then, using a union bound over all $x \in \{0,1\}^n$, we conclude that with non-zero probability, for all $x \in \{0,1\}^n$, there is an i such that $Q_{T_i}(x)$ contains a certificate of x. Fix a choice $T_1^*, T_2^*, \ldots, T_n^*$ of deterministic query trees in the support of $T^{\otimes\ell}$ such that for all x, there is an i such that $Q_{T_i^*}(x)$ includes a certificate of f at x. Let $\sigma_i(x) = \sigma_{T_i^*}(x)$ be the transcript of T_i^* on input x; let $C_{i,x} = \{y : \sigma_i(x) = \sigma_i(y)\}$. Note that each $C_{i,x}$ is a subcube; furthermore, if $Q_{T_i^*}(x)$ contains a certificate for x, then $C_{i,x}$ is monochromatic. For every i, the number of possibilities for $\sigma_i(x)$ and hence $C_{i,x}$ is at most $\mathrm{Size}(T)^{\ell}$. Then,

$$\{C_{i,x} : i \in [n], x \in \{0,1\}^n \text{ and } Q_{T_i^*}(x) \text{ contains a certificate for } x\}$$

is a cover for f with at most $n \cdot \mathrm{Size}(T)^{\ell}$ subcubes. Since T was an arbitrary $\left(\frac{1}{3}\right)$-error protocol, the lemma follows. □

3.1 A Better Bound for Monotone Functions

In this section, we observe that for monotone functions we can obtain better bounds than those provided by Theorem 3.1.

Theorem 3.4 (Improvement for monotone functions). *Suppose $f : \{0,1\}^n \to \{0,1\}$ is monotone. Then, we have the following.*

(a) $\log \mathrm{N}(f) = O(\log \mathrm{RSize}^{\mathrm{dt}}(f)\log(n))$;
(b) $\log \mathrm{DSize}^{\mathrm{dt}}(f) = O((\log \mathrm{RSize}^{\mathrm{dt}}(f))^2 \log^3(n))$.

Proof. Note that part (b) follows from part (a) and Theorem 2.13. Part (a) clearly holds if f is a constant function, so we restrict attention to functions f that are not constant. Note that $\#\mathrm{bs}(f, 0^n) = \mathrm{N}_1(f)$, for we have a distinct minimal sensitive block from each minterm; similarly, $\#\mathrm{bs}(f, 1^n) = \mathrm{N}_0(f)$, for we have one minimal sensitive block from each maxterm. It follows that $\mathrm{N}(f) \leq 2\cdot\#\mathrm{bs}(f)$. Then,

$$\log \mathrm{N}(f) \leq \log(\#\mathrm{bs}(f)) + 1 = O(\log \mathrm{RSize}^{\mathrm{dt}}(f)\log n),$$

where the last inequality follows from Lemma 3.2. □

4 (AND,OR) QUERY COMPLEXITY AND AND QUERY COMPLEXITY

In the previous section, we established that for every total function, there is a deterministic decision tree whose size is bounded by a quasi-polynomial of the *size* of its smallest randomized decision tree (ignoring some polynomial factors involving $\log n$ in the exponent). Though this result referred to decision tree *size*, we now use it to derive consequences for the randomized and deterministic *depth* in the (And, Or) decision tree model, and in the And (Or) decision tree model.

4.1 Randomized versus Deterministic (And, Or) Query Complexity

The following is our main theorem in this subsection, that shows that randomness does not offer much advantage in the (And,Or) decision tree model.

Theorem 4.1. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$\mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) = O(\mathrm{R}^{(\wedge,\vee)\text{-dt}}(f)^4 \log^7(n)).$$

The translation is achieved by showing that for all functions, the logarithm of the *size* of the best ordinary decision tree and the depth of an (And,Or) decision tree are closely related.

Lemma 4.2. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$, we have*

$$\log \mathrm{DSize}^{\mathrm{dt}}(f)/(2\log n) \leq \mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) \leq 4\log \mathrm{DSize}^{\mathrm{dt}}(f), \quad (2)$$

$$\log \mathrm{RSize}^{\mathrm{dt}}(f)/(2\log n) \leq \mathrm{R}^{(\wedge,\vee)\text{-dt}}(f) \leq 4\log \mathrm{RSize}^{\mathrm{dt}}(f). \quad (3)$$

Proof of Theorem 4.1. We use the second inequality of Equation (2) and the first inequality of Equation (3). We then have

$$\begin{aligned}\mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) &\leq 4\log \mathrm{DSize}^{\mathrm{dt}}(f)\\ &= O((\log \mathrm{RSize}^{\mathrm{dt}}(f))^4(\log n)^3)\\ &= O(\mathrm{R}^{(\wedge,\vee)\text{-dt}}(f)^4 \log^7(n)).\end{aligned}$$

The first line above follows from the second inequality in Equation (2), the second line by Theorem 3.1 (b), and the last line by the first inequality in Equation (3). □

It remains to establish Lemma 4.2. We need two observations, which we state in the following lemmas. We first observe that if (And,Or) queries are allowed, then an arbitrary decision tree can be *balanced* so that the depth of the resulting (And,Or) decision tree is logarithmic in the size of the original tree.

Lemma 4.3. *Suppose T is an ordinary decision tree for a function f. Then, there is an* (And, Or) *decision tree C for f such that.*

$$\mathrm{Depth}^{(\wedge,\vee)\text{-dt}}(C) \leq 4\log \mathrm{DSize}^{\mathrm{dt}}(T).$$

This lemma follows using a tree-balancing argument that has been used in various settings (Boolean formulas, communication protocols, etc.) previously. A formal justification appears in the full version of our paper [11]. Note that Lemma 4.3 immediately implies the second inequality in Equation (2) and also Equation (3). To justify the first inequality, we show that (And,Or) decision trees can be converted to ordinary decision trees whose size is bounded by an exponential in the depth of the original tree. Our justification of Lemma 4.2 will be complete once we show the following.

Lemma 4.4. *For every* (And, Or) *decision tree C of depth d computing f, there is an ordinary decision tree T computing f such that $\log \mathrm{Size}(T) \leq d\log\left(\frac{ne}{d}\right)$.*

Again we provide a formal justification in the full version of our paper [11]. An informal argument goes as follows. Replace each And query in the tree by a skewed ordinary decision tree for computing And. Label the edges corresponding to the value 0 as Red and the other edges as Blue. Similarly, replace each Or query by a skewed ordinary decision tree, but this time label the edge with label 1 as Red and the edges with label 0 as Blue. In the resulting tree, we may eliminate nodes where a variable queried earlier is queried again. Thus we obtain an ordinary decision tree with depth at most n and at most d Red edges on any root-to-leaf path. The number of paths to a leaf in this tree can be bounded by the number of sequences in $\{\text{Red}, \text{Blue}\}^n$ with at most d Red edges; any such sequence can be completed to a sequence with exactly $n+d$ elements, with exactly d Reds; thus, the number of leaves is at most $\binom{n+d}{d} \leq \left(\frac{2en}{d}\right)^d$.

4.2 Randomized versus Deterministic And Query Complexity

In the previous subsection, we saw an application of our main result to (And,Or) decision tree depth. That application crucially depended on our ability to balance decision trees by using (And,Or) queries. With just And queries, we cannot use the same argument. Nevertheless, in this section, we show that randomness does not offer much advantage even in the And decision tree model.

Theorem 4.5. *For every Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\mathrm{D}^{\wedge\text{-dt}}(f) = O(\mathrm{R}^{\wedge\text{-dt}}(f)^3 \log^4(n))$.*

Recall from Claim 2.16 that And decision tree depth complexity is bounded from below and above (up to an $O(\log n)$ factor) by 0-depth complexity, both in the deterministic and randomized settings. In this section, we will work mainly with 0-depth complexity and use this connection to derive our conclusion on And decision tree depth complexity.

Once again, our proof relies on two observations. First, a randomized decision tree of small 0-depth provides us a way of efficiently hitting the 0-blocks of f at any input. We formalize this using the notion of 0-fractional hitting set complexity.

Lemma 4.6. $\mathrm{FHSC}_0(f) = O(\mathrm{R}^{0\text{-dt}}(f))$.

Our next observation allows us to build 0-depth decision trees from 0-fractional hitting sets. This argument, which can be regarded as the analog of the result of Ehrenfeucht and Haussler [17] (see Theorem 2.13), is the main technical argument of this section.

Lemma 4.7. $\mathrm{D}^{0\text{-dt}}(f) = O(\mathrm{FHSC}_0(f)(\log \mathrm{N}_0(f) + \log \mathrm{N}_1(f)))$.

Before we formally justify these observations, let us see how they imply Theorem 4.5.

Proof of Theorem 4.5. We have

$$\begin{aligned}
\mathrm{D}^{\wedge\text{-dt}}(f) &\leq \mathrm{D}^{0\text{-dt}}(f)\log n && \text{(Claim 2.16)}\\
&= O(\mathrm{FHSC}_0(f)\cdot(\log \mathrm{N}_0(f) + \log \mathrm{N}_1(f)))\log n && \text{(Claim 4.7)}\\
&= O(\mathrm{R}^{0\text{-dt}}(f)\cdot(\log \mathrm{N}_0(f) + \log \mathrm{N}_1(f))\log n) && \text{(Claim 4.6)}\\
&= O(\mathrm{R}^{0\text{-dt}}(f)\cdot \log \mathrm{N}(f)\cdot \log n)\\
&= O(\mathrm{R}^{0\text{-dt}}(f)\cdot (\log \mathrm{RSize}^{\mathrm{dt}}(f))^2\cdot \log^2 n) && \text{(Theorem 3.1 (a))}\\
&\overset{(*)}{=} O(\mathrm{R}^{0\text{-dt}}(f)^3\cdot \log^4 n) && \text{(Equation (4))}\\
&= O(\mathrm{R}^{\wedge\text{-dt}}(f)^3\cdot \log^4 n). && \text{(Claim 2.16)}
\end{aligned}$$

The inequality marked (*) holds because a decision tree T on n variables (that queries no variable twice on the same path) with 0-depth d has at most $\binom{n+d}{d}$ leaves (see Lemma 4.4); thus

$$\log \mathrm{RSize}^{\mathrm{dt}}(f) = O(\mathrm{R}^{0\text{-dt}}(f)\log n). \tag{4}$$

□

Proof of Claim 4.7. We first present an informal overview. Suppose $\mathrm{FHSC}_0(f) = 1/p$. Then, by Lemma 2.7, for every input z and (multi-)set W of sensitive 0-blocks of f at z, there is a variable that hits at least a fraction p of the blocks in W. Our strategy for building the decision tree for f is as follows. After having queried some variables, we restrict attention to the unset variables, and the function induced on these variables. If the restricted function is a constant function, then we announce the value. Otherwise, we present a specific *reference input* (consistent with the answers received to the queries so far) and consider a (multi-)set W of sensitive 0-blocks of this input. The property above ensures that there is a variable that lies in a fraction p of these blocks. We query this variable; the choice of the variable will ensure that if this bit is 0 (in the actual input, not in the reference input), then after querying this bit, the size of either the 0-cover or the 1-cover for the restricted function will be at most a factor $1-p$ of what it was before the current step. Thus, on any input, by the time $t^* = \lceil (1/p)\ln(\mathrm{N}_0(f)\cdot \mathrm{N}_1(f))\rceil$ bits read turn out to be zero, either the 0-cover number or the 1-cover number for the restricted function will have become zero, making the restricted function constant. To implement this strategy, we need to specify how the following are determined at each step. Suppose we wish to compute $f(x)$.

Reference input: Our reference input z will initially be set to $(0,0,\ldots,0)$. Suppose the input bit $x[i]$ is queried in a certain step; we obtain the next reference input by changing that bit of the current reference input from 0 to $x[i]$. Thus, at every stage, the reference input is consistent with the part of x discovered by previous queries; the variables not yet queried continue to be 0.

The set of 0-blocks: Let $S \subseteq [n]$ be the indices of variables queried so far and $\rho : S \to \{0,1\}$ be the assignment obtained for them. Restrict attention to the variables with indices not in S. Assume that this restricted function is not constant, that is, the optimal size of 0-cover and 1-cover are at least 1. Note that the reference input z assigns zeros to all the unset variables. We have two cases.

- Suppose $f(z) = 0$: For each subcube C in the 1-cover consistent with the queried variables (for $i \in S$, $x[i]$ is unset or set to $\rho(i)$ in C), identify it with a conjunction of the form
$$\bigwedge_{j\in B_C} x[j] \ \wedge \bigwedge_{j\in B'_C} \overline{x[j]}, \tag{5}$$
where $B_C = \{i : i \notin S \text{ and } x[i] \text{ is set to 1 in } C\}$ and $B'_C = \{i : i \notin S \text{ and } x[i] \text{ is set to 0 in } C\}$. Since $f(z) = 0$, each B_C must be non-empty; furthermore, each B_C is a sensitive 0-block for f at z. We define W to be the multiset $\{B_C : C \text{ is a subcube in the 1-cover consistent with the queried variables}\}$.
- Suppose $f(z) = 1$: We consider subcubes in the optimal 0-cover consistent with the queried variables, and identify each of them with a conjunction of the form (5). We obtain a multi-set W of sensitive 0-blocks for the reference input z as before.

The bit to be queried next: By construction, each block in W is a sensitive 0-block for the reference input z. So by the property stated at the outset, there is a variable $x[i]$ that belongs to at least a fraction p of the blocks in W (counted with multiplicity). Note that if this variable is queried and the input bit $x[i]$ is revealed to be zero, then the subcubes C in the cover corresponding to blocks in W where $i \in B_C$ do not contribute to the cover for the new restricted function.

At each step, if the value of f has not yet been determined, we determine the next variable to be queried as described above. This completes the description of our strategy. It remains to bound the 0-depth of the decision tree corresponding to this strategy. Let N_0^t and N_1^t denote the number of subcubes surviving (that is, consistent with values the variables already queried) after t queries. Note that surviving subcubes form a cover for the restricted function, although it need not be optimal. Consider the quantity $M_t = \mathrm{N}_0^t \cdot \mathrm{N}_1^t$. Initially, $M_0 = \mathrm{N}_0(f) \cdot \mathrm{N}_1(f)$; and in general $M_t \leq M_0(1-p)^t$. Set $t^* = \lceil (1/p) \ln(\mathrm{N}_0(f) \cdot \mathrm{N}_1(f)) \rceil$. Suppose we read at least $t^* + 1$ zeros; then $M_{t^*} < 1$, that is, $M_{t^*} = 0$ (one of the covers has become empty); but then we would have stopped when we read t^* zeros—a contradiction. It follows that the 0-depth of the decision tree underlying this strategy is $O((1/p)\log(\mathrm{N}_0(f) \cdot \mathrm{N}_1(f))$. □

Proof of Claim 4.6. Let T be an ε-error randomized decision tree for f with 0-depth d. We will show that for each $x \in \{0,1\}^n$, we have

$$\mathrm{FHSC}_0(f,x) \leq \frac{d}{1-2\varepsilon}. \tag{6}$$

Our claim will follow from this, because T was fixed arbitrarily. By Lemma 2.14, for each block $B \in \mathcal{W}_0(f,x)$, we have $\Pr[Q_T(x) \cap B \neq \emptyset] \geq 1-2\varepsilon$. Let $Q_T^0(x) = \{i \in Q_T(x) : x[i] = 0\}$; note that $|Q_T^0(x)| \leq d$. Since B is a zero block of f at x, we have that $x[i] = 0$ for each $i \in B$; thus, $\Pr[Q_T^0(x) \cap B \neq \emptyset] \geq 1-2\varepsilon$. Let the index I be chosen uniformly from $Q_T^0(x)$. Then, for each $B \in \mathcal{W}_0(f,x)$, we have $\Pr[I \in B] \geq (1-2\varepsilon)/d$. This establishes Equation (6): we take the distribution of I as the distribution D required in the definition of $\mathrm{FHSC}_0(f,x)$ (see Definition 2.6). □

5 CONCLUSIONS

Nisan's result from the late eighties [35] can be viewed as the analog of derandomizing time for decision trees. The complexity measure of logarithm of decision tree size may naturally be thought of as the analog of space for decision trees. In this light, we would like to view our main theorem as providing a derandomization of space in the decision tree model. Nisan's result upper bounds the deterministic depth complexity of a function f by the cube of its randomized depth. It remains an outstanding problem to determine if this gap is achieved by some total function. Relatively recently, a new total function was defined by Göös, Pitassi and Watson [21] to separate deterministic partition number and communication complexity by a quadratic factor, solving a longstanding problem. A modification of this function, which we refer to henceforth as the modified GPW function, was shown in [1] to achieve a quadratic gap between the bounded-error randomized and deterministic decision tree depth. In particular, the modified GPW function has deterministic depth $\Omega(n)$ but randomized depth $\Theta(\sqrt{n})$. No better separation is known for depth for any function.

It becomes natural to wonder if our Theorem 1.1 obtains the optimal relationship between the logarithms of randomized and deterministic decision tree size. Interestingly, even though the modified GPW function provided a breakthrough in increasing the known gap between randomized and deterministic depth, it turns out that both its deterministic and randomized size is $2^{\widetilde{\Theta}(\sqrt{n})}$. We observe however (see Section C), that lifting a function with the XOR gadget lifts a deterministic-randomized depth separation to a deterministic-randomized log size separation. Thus, the largest known separation between the logarithms of randomized and deterministic decision tree size is quadratic as well.

For ADT's, again it can be shown that the modified GPW function fails to generate any advantage for randomized algorithms w.r.t. depth, however lifting it by the XOR gadget again gives a quadratic separation. It would be interesting to find either functions yielding larger gaps or to narrow down the gap from cubic given by our Theorem 1.3.

ACKNOWLEDGMENTS

We thank Shachar Lovett for useful comments on an earlier draft of this paper, and Weiqiang Yuan for pointing out to us that the best possible separation between logarithm of deterministic decision tree size and logarithm of randomized decision tree size, and between deterministic and randomized ADT's, must be at least quadratic. AC was partially supported by the Science & Engineering Research Board of the DST, India, through the MATRICS grant MTR/2019/001633. NSM was supported by the Dutch Research Council (NWO), as part of the Quantum Software Consortium programme (project number 024.003.037). JR was partially supported by the Science & Engineering Research Board of the DST, India, through the MATRICS grant MTR/2019/001226. AC and JR were partially supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500. SS was supported through an ISIRD grant by SRIC, IIT Kharagpur.

A LOWER BOUNDS FOR RANDOMIZED DECISION TREE SIZE

In this section we first observe that randomized communication complexity gives a lower bound on randomized decision tree size. As a consequence, using a communication lower bound due to Göös and Jayram [20], we conclude a lower bound on the randomized decision tree size of the And-Or tree on n variables, nearly matching the upper bound implied by the following result.

Theorem A.1 ([37]). *Let $f : \{0,1\}^n \to \{0,1\}$ be the* And-Or *tree on n variables. Then,* $\mathrm{R}^{\mathrm{dt}}_{1/3}(f) = \Theta(n^{0.753\ldots})$.

This immediately implies

$$\mathrm{RSize}^{\mathrm{dt}}_{1/3}(f) = 2^{O(n^{0.753\ldots})}. \tag{7}$$

After this, we give a general template to prove randomized decision tree size lower bounds and demonstrate its use by showing lower bounds for specific classes of Boolean functions.

A.1 Communication Complexity

We refer the reader to the full version of our paper [11] for the relevant preliminaries in communication complexity. We require the following theorem due to Göös and Jayram [20, Theorem 2.3].

Theorem A.2. *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the* And-Or *tree on $2n$ inputs, where Alice gets the first bit of every bottom gate and Bob gets the rest. Then, $\mathrm{R}^{\mathrm{cc}}_{\varepsilon}(F) = \widetilde{\Omega}(n^{0.753\dots})$.*

A.2 Lower Bounds via Communication Complexity

We observe the following.

Lemma A.3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function such that $\mathrm{R}^{\mathrm{cc}}_{\varepsilon}(f) \geq c$ under some partition of the inputs. Then, $\mathrm{RSize}^{\mathrm{dt}}_{\varepsilon}(f) = 2^{\Omega(c)}$.*

Proof. Consider an arbitrary partition of inputs between Alice and Bob. Consider a randomized decision tree of size s computing f to error ε. One obtains a communication protocol for f in the following way: Alice and Bob sample a tree from the underlying distribution of the randomized decision tree using public randomness. They then evaluate this tree from the root down using a communication protocol; if a node has the label of a variable held by Alice (Bob), she (he) sends its value to Bob (Alice) and they continue in the relevant subtree. When they reach a leaf they output the value at that leaf. Note that the randomized protocol thus obtained has exactly the same structure as the randomized decision tree. It is well known that communication protocols can be balanced (using the same ideas as in the proof of Lemma 4.3). We thus obtain a randomized protocol of cost $O(\log s)$ that computes f to error ε. This proves the lemma. □

Along with Theorem A.2, this implies the following, yielding near-tightness of the bound in Equation (7).

Corollary A.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be the* And-Or *tree on n inputs. Then, $\mathrm{RSize}^{\mathrm{dt}}_{1/3}(f) = 2^{\widetilde{\Omega}(n^{0.753\dots})}$.*

In view of the above, one might ask if it is necessary for the randomized communication complexity of a function to be large under some partition of inputs in order for its randomized decision tree size to be large. Specifically, can one show that the randomized decision tree size of the well-studied Equality function is large, even though its randomized communication complexity is a constant? In Section A.3 we show that the randomized decision tree size of Equality is indeed large, as an application of a more general template that we introduce to prove lower bounds against randomized decision tree size.

A.3 A Template for Proving Randomized Decision Tree Size Lower Bounds

In this section, we give a set of sufficient conditions which imply randomized decision tree size lower bounds. In particular, we can use these to recover Theorem 3.1 for the particular case of monotone functions and we can also use these in other cases to prove better randomized size lower bounds. We start by giving some definitions. For a subcube A, its 0-codimension is the number of variables fixed to 0 by A and its 1-codimension is the number of variables fixed to 1 by A. We define a notion of spreadness for a distribution on $\{0,1\}^n$. For a set $S \subseteq \{0,1\}^n$ and a distribution μ on $\{0,1\}^n$, let $\mu(S) = \Pr_{x\sim\mu}[x \in S]$.

Definition A.5 (Spreadness). *A probability distribution μ on $\{0,1\}^n$ is*

- *(ℓ,κ)-spread if for a subcube S of codimension ℓ, $\mu(S) \leq \kappa$.*
- *$(\ell,\kappa,1)$-spread if for a subcube S of 1-codimension ℓ, $\mu(S) \leq \kappa$.*
- *$(\ell,\kappa,0)$-spread if for a subcube S of 0-codimension ℓ, $\mu(S) \leq \kappa$.*

Next, we define a notion of sensitivity of a distribution. For $b \in \{0,1\}$, a set $B \subseteq [n]$ is a *sensitive b-block of f at input x* if $f(x \oplus 1_B) \neq f(x)$ and $x_i = b$ for each $i \in B$. The *b-block sensitivity of f at x*, denoted by $\mathrm{bs}_b(f,x)$, is the maximum integer r for which there exist r disjoint sensitive b-blocks of f at x.

Definition A.6 (Sensitivity of a distribution). *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and $b \in \{0,1\}$, a distribution μ on $\{0,1\}^n$ is*

- *(ℓ,b)-sensitive w.r.t. f if for each input x in the support of μ, we have $f(x) = b$ and $\mathrm{bs}(f,x) \geq \ell$.*
- *$(\ell,b,0)$-sensitive w.r.t. f if for each input x in the support of μ, we have $f(x) = b$ and $\mathrm{bs}_0(f,x) \geq \ell$.*
- *$(\ell,b,1)$-sensitive w.r.t. f if for each input x in the support of μ, we have $f(x) = b$ and $\mathrm{bs}_1(f,x) \geq \ell$.*

We now show that the existence of a distribution that is spread and sensitive w.r.t. f implies large randomized decision tree size complexity.

Theorem A.7. *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let $b \in \{0,1\}$ and let μ_b be a distribution on $\{0,1\}^n$ which satisfies any of the following properties:*

1. *μ_b is (ℓ,b)-sensitive w.r.t. f and $(\lceil \ell/2\rceil,\kappa)$-spread.*
2. *There exists $c \in \{0,1\}$ such that μ_b is (ℓ,b,c)-sensitive w.r.t. f and $(\lceil \ell/2\rceil,\kappa,c)$-spread.*

Then $\mathrm{RSize}^{\mathrm{dt}}_{1/10}(f) = \Omega(1/\kappa)$.

Proof. We prove (1) above, the proof of (2) follows along similar lines. Without loss of generality, assume $b = 0$. Thus, μ_0 is a distribution over inputs in $f^{-1}(0)$ that is $(\ell,0)$-sensitive. Define μ_1, a distribution over $f^{-1}(1)$, as follows: Sample an input x according to μ_0 and let $B_1, B_2, \dots, B_r$ be disjoint sensitive block of f at x where $r = \mathrm{bs}(f,x)$. Sample a i from $[r]$ and return $x \oplus 1_{B_i}$. Define the distribution $\mu = (\mu_0 + \mu_1)/2$. We show below that $\mathrm{DSize}^{\mathrm{dt}}_{\mu,1/10}(f) = \Omega(1/\kappa)$, thereby showing $\mathrm{RSize}^{\mathrm{dt}}_{1/10}(f) = \Omega(1/\kappa)$. Let T be a deterministic decision tree that computes f correctly on at least a $(9/10)$-mass of inputs sampled according to μ. Since μ can be viewed as sampling according to μ_0 with probability $1/2$ and according to μ_1 with probability $1/2$, T must be correct on at least a $(4/5)$ mass of μ_0 as well as a $(4/5)$ mass of μ_1. Let L_0 and L_1 be set of all 0-leaves and 1-leaves of T, respectively. Define

$$\rho_0 = \sum_{v\in L_0} \Pr_{x\sim\mu_0}[x \text{ reaches } v], \quad \rho_1 = \sum_{v\in L_1} \Pr_{x\sim\mu_1}[x \text{ reaches } v].$$

That is, ρ_0 (ρ_1) denotes the μ_0-mass (μ_1-mass) captured by 0-leaves (1-leaves). By the argument above, both ρ_0 and ρ_1 must be at least

4/5. We observe following about the 0-paths (paths leading to 0-leaves).

(1) (Short Paths). Firstly, we show that the 0-paths of length less than $\lceil \ell/2 \rceil$ must capture less than 2/5 mass of μ_0, i.e., $\sum_{\substack{v \in L_0 \\ |v| < \lceil \ell/2 \rceil}} \Pr_{x \sim \mu_0}[x \text{ reaches } v] \leq 2/5$, where $|v|$ denotes the length of the path leading to v. Towards a contradiction, assume $\sum_{\substack{v \in L_0 \\ |v| < \lceil \ell/2 \rceil}} \Pr_{x \sim \mu_0}[x \text{ reaches } v] > 2/5$. Let S be a subcube with codimension less than $\lceil \ell/2 \rceil$ corresponding to a 0-leaf. For each x supported by μ_0 which lies in S, our assumption on μ_0 implies that $\mathrm{bs}(f,x) \geq \ell$. Thus, there must exist at least $\mathrm{bs}(f,x)/2$ blocks such that none of their variables are read on the path to this leaf. Flipping any one of these blocks gives an input supported by μ_1 and that still reaches this leaf. Hence $\mu_1(S) \geq (1/2)\mu_0(S)$. Thus,

$$\sum_{\substack{v \in L_0 \\ |v| < \lceil \ell/2 \rceil}} \Pr_{x \sim \mu_1}[x \text{ reaches } v] \geq \frac{1}{2} \cdot \sum_{\substack{v \in L_0 \\ |v| < \lceil \ell/2 \rceil}} \Pr_{x \sim \mu_0}[x \text{ reaches } v] > 1/5.$$

This contradicts the fact that T is correct on at least a 4/5 mass of μ_1.

(2) (Long Paths). Secondly, a 0-path of length at least $\lceil \ell/2 \rceil$ captures at most κ of μ_0 mass. This follows from the spreadness property of μ_0.

Since ρ_0 must be atleast 4/5, from the first bullet above, we have that 0-paths in T of length at least $\lceil \ell/2 \rceil$ must capture at least 2/5 of μ_0 mass. By the second bullet above, there must be at least $2/(5\kappa)$ 0-paths in T of length at least $\lceil \ell/2 \rceil$, proving an $\Omega(1/\kappa)$ lower bound on the number of leaves in T.

The analysis for (2) is along similar lines. The only difference will be in using sensitive c-blocks in place of sensitive blocks and defining short and long paths for the case analysis above based on c-codimension of the corresponding subcubes. □

Below we give some examples to showcase the usage of the above template to prove randomized decision tree size lower bounds for monotone Boolean functions. We refer the reader to the full version of our paper [11] for applications of our template to the Equality function and symmetric functions.

A.3.1 Monotone Functions. In this section, we use our template to give an alternative proof of Theorem 3.4. We prove part (a) of Theorem 3.4 as part (b) follows from part (a) and Theorem 2.13.

Proof. (Part (a) Theorem 3.4). We show that f has a sub-function (f under some restriction) g that has large randomized decision tree size. The statement of theorem trivially holds when $\mathrm{N}(f) \leq n^c$ for any fixed constant c, so we restrict our attention to the case when $\mathrm{N}(f) \geq n^c$ for some appropriate c to be fixed later. Without loss of generality, let $\mathrm{N}_1(f) \geq \mathrm{N}(f)/2$, the case when $\mathrm{N}_0(f) \geq \mathrm{N}(f)/2$ is similar. For a monotone function f, the subcubes corresponding to minterms of f form the unique minimum 1-cover for f. For a minterm M, let x_M be the input such that $x_i = 1$ for $i \in M$ and 0 otherwise. Let $S = \{x_M | M \text{ is a minterm of } f\}$. Since we have a distinct minimal sensitive block for 0^n from each minterm, $|S| = \mathrm{N}_1(f)$ and each $x \in S$ is a 1-input of f with $\mathrm{bs}_1(f,x) = |x|$, i.e., each 1-bit position of x is sensitive. For $i \in [n]$, let S_i denote inputs of Hamming weight i in S. By an averaging argument, there exists $j \in [n]$ such that $|S_j| \geq \mathrm{N}_1(f)/n$. We give an iterative procedure to find our desired sub-function g.

(1) Initialise the parameters $V = [n]$, $g = f$, $\ell = j$, $R = S_j$ and $k = \log(\mathrm{N}_1(f)/n)/(2(\lceil \log n \rceil + 1))$. We use V to denote the input domain of g. Throughout the process, R will be a set of 1-inputs of g over $\{0,1\}^V$ each of Hamming weight and 1-block sensitivity ℓ w.r.t. g.

(2) Find a subcube P over $\{0,1\}^V$ of 1-codimension $\lceil \ell/2 \rceil$ such that it contains more than 2^{-k} fraction of inputs in R. Let $Q \subseteq V$ be the set of variables fixed by P. If no such subcube exists, stop the procedure. Else, update V to $[n] \setminus Q$, ℓ to $\lfloor \ell/2 \rfloor$ and g to $g|_P$ (g restricted to P). For $V \subseteq T$ and $x \in \{0,1\}^T$, let x_V in $\{0,1\}^V$ denote the projection of x on to V. Update R to $\{x_V | x \in R \text{ and } x \text{ lies in } P\}$. Note that after this update, g is a function on $\{0,1\}^V$ and $R \subseteq \{0,1\}^V$. Also, each $x \in R$ is a 1-input of g, has Hamming weight ℓ, and each 1-bit position of x forms a sensitive 1-block for g i.e., $\mathrm{bs}_1(g,x) = \ell$.

(3) Repeat the above step.

We claim the above procedure can not run forever and must terminate within the second step's ($\lceil \log n \rceil + 1$)'th repetition. The procedure starts with each input in R having Hamming weight ℓ, and with every repetition of the second step, the updated R contains inputs with Hamming weight halved from the previous iteration. So if the second step repeats ($\lceil \log n \rceil + 1$) times, R can have only one input, namely 0^n. On the other hand, the size of R only reduces by a factor of 2^{-k} with each repetition of the second step because of the way we choose the subcube P, and so by the end of ($\lceil \log n \rceil + 1$) repetitions, the size of R must be

$$\begin{aligned}|R| \geq |S_j| 2^{-k(\lceil \log n \rceil + 1)} &\geq (\mathrm{N}_1(f)/n) 2^{-k(\lceil \log n \rceil + 1)} \\ &= (\mathrm{N}_1(f)/n)^{1/2} \geq n^{\frac{c-1}{2}}.\end{aligned}$$

So if we choose $c > 1$, the size of $|R|$ will be strictly larger than 1, contradicting our earlier conclusion, which says the size of R can be at most 1 after ($\lceil \log n \rceil + 1$) repetitions of the second step. So, the above procedure does terminate and at the end of the process we get g, a subfunction of f on $\{0,1\}^V$, and $R \subseteq \{0,1\}^V$ with following properties

- $g(x) = 1$ for all $x \in R$.
- (Spreadness). Each $x \in R$ has Hamming weight ℓ and no subcube of 1-codimension $\lceil \ell/2 \rceil$ contains more than 2^{-k} fraction of inputs of R.
- (Sensitivity). Each $x \in R$ has $\mathrm{bs}_1(g,x) = \ell$.

Let μ_1 be the distribution that randomly samples an input from R. From the properties of R stated above, we get μ_1 is a distribution over 1-inputs of g which is $(\lceil \ell/2 \rceil, 2^{-k}, 1)$-spread and $(\ell, 1, 1)$-sensitive w.r.t. g. Applying Theorem A.7, we get $\log \mathrm{RSize}^{\mathrm{dt}}_{1/10}(f) \geq \log \mathrm{RSize}^{\mathrm{dt}}_{1/10}(g) = \Omega(k) = \Omega(\log \mathrm{N}_1(f)/\log n)$, which yields a lower bound of $\Omega(\log \mathrm{N}(f)/\log n)$. □

B NECESSITY OF LOG FACTORS

We give an example which shows that a factor of $\log n$ can not be avoided in Theorem 4.1 and Theorem 4.5. Let $\textsc{Thr}^n_k : \{0,1\}^n \to \{0,1\}$ be the function which outputs 1 if and only if $|x| \geq k$. $\textsc{Thr}^n_{n-1}$ and $\textsc{Thr}^n_2$ appeared in the work of [30] and [4] to demonstrate that deterministic And (Or) query complexity and randomized And (Or) query complexity can be arbitrarily apart. Below we observe that even deterministic (And, Or) query complexity can be arbitrarily separated from randomized (And, Or) query complexity.

Claim B.1. $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\textsc{Thr}^n_{n-1}) = \Theta(\log n)$.

Proof. First, we show that $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\textsc{Thr}^n_{n-1}) = O(\log n)$. In fact $\mathrm{D}^{\wedge\text{-dt}}(\textsc{Thr}^n_{n-1})$ is $O(\log n)$. For input $x \in \{0,1\}^n$, f outputs 0 if and only if it contains 2 or more than 2 bits set to 0 in x. Consider the query algorithm which uses binary search for the first-bit position set to 0 in x. This can be done using $\lceil \log n \rceil$ And queries. If the binary search fails to find a 0 bit output 1, the input must have been all 1 input. Otherwise, let i be the bit position the binary search returns. If $i = n$ output 1, else output the answer to the query $\bigwedge_{j=i+1}^{n} x_j$. So we need at most $\lceil \log n \rceil + 1$ And queries to compute $\textsc{Thr}^n_{n-1}$.

For the lower bound, we give an adversary strategy to respond to queries such that it forces any correct (And, Or)-decision tree for $\textsc{Thr}^n_{n-1}$ to make $\Omega(\log n)$ queries. The adversary, $\mathcal{A}$, maintains a partial assignment ρ. Let S be a set of unset bits in ρ. Bits set by ρ will all be set to 1 by the adversary. $\mathcal{A}$ starts with all bits unset i.e., $S = [n]$ to begin with. For a query Q, if Q is already determined by ρ, $\mathcal{A}$ answers accordingly. Otherwise, let Q' be the reduced query containing variables from set $P \subseteq S$. $\mathcal{A}$ answers Q' as follows:

(1) If Q' is an Or query, $\mathcal{A}$ answers 1 and set x_i to 1 in ρ where x_i is an arbitrary variable in Q'.
(2) If Q' is an And query and $|P| < |S|/2$, $\mathcal{A}$ answers 1 and set all variables indexed by P to 1 in ρ.
(3) If Q' is an And query and $|P| \geq |S|/2$ then $\mathcal{A}$ answers 0 and set all variables indexed by $S \setminus P$ to 1 in ρ.

It is clear that in cases 1 and 2, the partial assignment ρ maintained by the adversary makes him consistent with his answers. For case 3, the adversary can stay consistent with his answers if the set of unset bits, S, is non-empty. Moreover, at any stage of answering, if $|S| \geq 2$ then there exist two inputs consistent with the adversary's answers with Hamming weight $n-2$ and $n-1$, respectively. This tells us that as long as more than 1 bit is unset in ρ, no (And, Or)-decision tree for $\textsc{Thr}^n_{n-1}$ can output an answer and be error-free. Since with each Or query size of S reduces by 1 and with each And query it reduces by at most a multiplicative factor of 1/2, we conclude that a correct (And, Or)-decision tree for $\textsc{Thr}^n_{n-1}$ must make $\Omega(\log n)$ queries. □

Note that by symmetry we also get $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\textsc{Thr}^n_{2}) = \Theta(\log n)$.

Observation B.2 ([4],[30, Example 6.3]). $\mathrm{R}^{\wedge\text{-dt}}(\textsc{Thr}^n_{n-1}) = O(1)$. *Indeed, consider the randomized* And *query algorithm which samples a subset $S \subseteq [n]$ uniformly at random and outputs 0 if both $\bigwedge_{i\in S} x_i$ and $\bigwedge_{i\notin S} x_i$ equal to 0 and output 1 otherwise. Note that if $|x| \geq n-1$, the algorithm outputs the correct answer with probability 1. If $|x| \leq n-2$, then with probability 1/2 the algorithm is correct. Repeating the algorithm once more will give an error probability of 1/4.*

Combining Claim B.1 and Observation B.2, we get that $\mathrm{R}^{\wedge\text{-dt}}(\textsc{Thr}^n_{n-1}) = O(1)$ and $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\textsc{Thr}^n_{n-1}) = \Omega(\log n)$, telling us that a dependence on input dimension can not be avoided in Theorem 4.1 and Theorem 4.5.

C BEST POSSIBLE SEPARATIONS

We next observe that the best possible separation between the log of deterministic size and the log of randomized size (and also the best possible separation between deterministic and randomized And (Or) query complexity) must be as large as the best possible separation between deterministic and randomized query complexity. We thank Weiqiang Yuan for pointing out this observation to us. We need the following deterministic size lower bound for composed functions which follows from [16, Theorem 6.6] and Proposition 2.11.

Corollary C.1 (Proposition 2.11, [16, Theorem 6.6]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $\log \mathrm{DSize}^{\mathrm{dt}}(f \circ \textsc{Xor}) \geq \mathrm{Depth}(f) + 1$.*

Observation C.2. *For a Boolean function f, $\log \mathrm{RSize}^{\mathrm{dt}}(f \circ \textsc{Xor}) = O(\mathrm{R}^{\mathrm{dt}}(f))$. Indeed, one way to construct a randomized decision tree for $f \circ$ Xor is to start with an optimal randomized decision tree T for f, inflate each internal node u of T into a copy of* Xor *on the appropriate input variables and attach the left and the right subtree of u as appropriate at the leaves of this copy of* Xor*. From construction, it is clear that such a decision tree will compute $f \circ$ Xor with the same error probability as that of T on f. Furthermore, the size of the constructed tree is bounded by $4^{\mathrm{R}^{\mathrm{dt}}(f)}$.*

Combining Corollary C.1 and Observation C.2, we have that for any Boolean function f, $\log \mathrm{RSize}^{\mathrm{dt}}(f \circ \textsc{Xor}) = O(\mathrm{R}^{\mathrm{dt}}(f))$ and $\log \mathrm{DSize}^{\mathrm{dt}}(f \circ \textsc{Xor}) = \Omega(\mathrm{D}^{\mathrm{dt}}(f))$, giving us that the separation between $\log \mathrm{RSize}^{\mathrm{dt}}$ and $\log \mathrm{DSize}^{\mathrm{dt}}$ is at least as large as the separation between R^{dt} and D^{dt}. In particular, using the modified GPW function from [1] as f we get that $\log \mathrm{RSize}^{\mathrm{dt}}$ and $\log \mathrm{DSize}^{\mathrm{dt}}$ are quadratically separated. Furthermore, using Lemma 4.2, we get that $\mathrm{D}^{(\wedge,\vee)\text{-dt}}$ and $\mathrm{R}^{(\wedge,\vee)\text{-dt}}$ must also be at least quadratically separated. Next, we observe that the separation between $\mathrm{D}^{\wedge\text{-dt}}$ and $\mathrm{R}^{\wedge\text{-dt}}$ must also be at least as large as the separation between D^{dt} and R^{dt}.

Observation C.3. *For a Boolean function f, $\mathrm{D}^{\wedge\text{-dt}}(f \circ \textsc{Xor}) \geq \mathrm{D}^{\mathrm{dt}}(f)/\log n$*

$$\begin{aligned}
\mathrm{D}^{\wedge\text{-dt}}(f \circ \textsc{Xor}) &\geq \mathrm{D}^{0\text{-dt}}(f \circ \textsc{Xor}) && \textit{(by Claim 2.16)}\\
&\geq \log \mathrm{DSize}^{\mathrm{dt}}(f \circ \textsc{Xor})/\log n && \textit{(by Equation (4))}\\
&\geq \mathrm{D}^{\mathrm{dt}}(f)/\log n && \textit{(by Corollary C.1)}.
\end{aligned}$$

However, $\mathrm{R}^{\wedge\text{-dt}}(f \circ \textsc{Xor}) \leq 2\mathrm{R}^{\mathrm{dt}}(f)$, since we can simulate the optimal randomized decision tree for f, querying both the variables corresponding to i-th copy of Xor *when the decision tree queries the i-th input bit. Using the modified GPW function from [1] as f we get that $\mathrm{D}^{\wedge\text{-dt}}$ and $\mathrm{R}^{\wedge\text{-dt}}$ are at least quadratically separated.*

Using the observation above, our result in Theorem 4.5 can be considered a generalization of Nisan's derandomization of ordinary decision tree depth. Using Observation C.3 and Theorem 4.5, we can recover Nisan's cubic relationship between D^{dt} and R^{dt} upto polylog n factors.

$$\begin{aligned}\mathrm{D}^{\mathrm{dt}}(f) =& O(\mathrm{D}^{\wedge\text{-dt}}(f \circ \textsc{Xor}) \log n) = O(\mathrm{R}^{\wedge\text{-dt}}(f \circ \textsc{Xor})^3 \cdot \log^5 n) \\ =& O(\mathrm{R}^{\mathrm{dt}}(f)^3 \cdot \log^5 n).\end{aligned}$$

REFERENCES

[1] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. 2017. Separations in Query Complexity Based on Pointer Functions. *J. ACM* 64, 5 (2017), 32:1–32:24. https://doi.org/10.1145/3106234 Earlier version in STOC 2016.

[2] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. 2001. Quantum lower bounds by polynomials. *J. ACM* 48, 4 (2001), 778–797. https://doi.org/10.1145/502090.502097 Earlier version in FOCS 1998.

[3] Aleksandrs Belovs. 2015. Quantum Algorithms for Learning Symmetric Juntas via the Adversary Bound. *Comput. Complex.* 24, 2 (2015), 255–293. https://doi.org/10.1007/s00037-015-0099-2 Earlier version in CCC 2014.

[4] Yosi Ben-Asher and Ilan Newman. 1995. Decision trees with Boolean threshold queries. *J. Comput. System Sci.* 51, 3 (1995), 495–502. https://doi.org/10.1006/jcss.1995.1085

[5] Shalev Ben-David, Adam Bouland, Ankit Garg, and Robin Kothari. 2018. Classical Lower Bounds from Quantum Upper Bounds. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 339–349. https://doi.org/10.1109/FOCS.2018.00040

[6] Gal Beniamini and Noam Nisan. 2021. Bipartite perfect matching as a real polynomial. In *53rd Annual ACM SIGACT Symposium on Theory of Computing STOC*. ACM, 1118–1131. https://doi.org/10.1145/3406325.3451002

[7] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. 2021. Properly learning decision trees in almost polynomial time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE, 920–929. https://doi.org/10.1145/3561047

[8] Guy Blanc, Jane Lange, and Li-Yang Tan. 2020. Top-Down Induction of Decision Trees: Rigorous Guarantees and Inherent Limitations. In *11th Innovations in Theoretical Computer Science Conference, ITCS (LIPIcs, Vol. 151)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 44:1–44:44. https://doi.org/10.4230/LIPIcs.ITCS.2020.44

[9] Joakim Blikstad, Jan Van Den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. 2022. Nearly optimal communication and query complexity of bipartite matching. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1174–1185. https://doi.org/10.1109/FOCS54457.2022.00113

[10] Manuel Blum and Russell Impagliazzo. 1987. Generic oracles and oracle classes. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. IEEE, 118–126. https://doi.org/10.1109/SFCS.1987.30

[11] Arkadev Chattopadhyay, Yogesh Dahiya, Nikhil S. Mande, Jaikumar Radhakrishnan, and Swagato Sanyal. 2022. Randomized versus Deterministic Decision Tree Size. *Electron. Colloquium Comput. Complex.* TR22-185 (2022). ECCC:TR22-185 https://eccc.weizmann.ac.il/report/2022/185

[12] Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. 2021. Query-to-Communication Lifting Using Low-Discrepancy Gadgets. *SIAM J. Comput.* 50, 1 (2021), 171–210. https://doi.org/10.1137/19M1310153 Preliminary version in ICALP, 2019.

[13] Arkadev Chattopadhyay, Ankit Garg, and Suhail Sherif. 2021. Towards Stronger Counterexamples to the Log-Approximate-Rank Conjecture. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS (LIPIcs, Vol. 213)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 13:1–13:16. https://doi.org/10.4230/LIPIcs.FSTTCS.2021.13

[14] Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. 2019. Simulation Theorems via Pseudo-random Properties. *Comput. Complex.* 28, 4 (2019), 617–659. https://doi.org/10.1007/s00037-019-00190-7

[15] Arjan Cornelissen, Nikhil S. Mande, and Subhasree Patro. 2022. Improved Quantum Query Upper Bounds Based on Classical Decision Trees. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS (LIPIcs, Vol. 250)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:22. https://doi.org/10.4230/LIPIcs.FSTTCS.2022.15

[16] Yogesh Dahiya and Meena Mahajan. 2021. On (Simple) Decision Tree Rank. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS (LIPIcs, Vol. 213)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:16. https://doi.org/10.4230/LIPIcs.FSTTCS.2021.15

[17] Andrzej Ehrenfeucht and David Haussler. 1989. Learning decision trees from random examples. *Information and Computation* 82, 3 (1989), 231 – 246. https://doi.org/10.1016/0890-5401(89)90001-1 Earlier version in COLT'88.

[18] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. 2020. Monotone Circuit Lower Bounds from Resolution. *Theory Comput.* 16 (2020), 1–30. https://doi.org/10.1145/3188745.3188838 Earlier version in STOC'18.

[19] Uma Girish, Avishay Tal, and Kewen Wu. 2021. Fourier Growth of Parity Decision Trees. In *36th Computational Complexity Conference, CCC (LIPIcs, Vol. 200)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 39:1–39:36. https://doi.org/10.4230/LIPIcs.CCC.2021.39

[20] Mika Göös and T. S. Jayram. 2016. A Composition Theorem for Conical Juntas. In *31st Conference on Computational Complexity, CCC (LIPIcs, Vol. 50)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 5:1–5:16. https://doi.org/10.4230/LIPIcs.CCC.2016.5

[21] Mika Göös, Toniann Pitassi, and Thomas Watson. 2018. Deterministic Communication vs. Partition Number. *SIAM J. Comput.* 47, 6 (2018), 2435–2450. https://doi.org/10.1109/FOCS.2015.70 Earlier version in FOCS 2015..

[22] Mika Göös, Toniann Pitassi, and Thomas Watson. 2020. Query-to-Communication Lifting for BPP. *SIAM J. Comput.* 49, 4 (2020). https://doi.org/10.1109/FOCS.2017.21 Preliminary version in FOCS, 2017.

[23] Juris Hartmanis and Lane A Hemachandra. 1986. *One-way functions, robustness, and the non-isomorphism of NP-complete sets.* Technical Report. Cornell University. https://doi.org/1813/6636

[24] Hamed Hatami, Kaave Hosseini, and Shachar Lovett. 2018. Structure of Protocols for XOR Functions. *SIAM J. Comput.* 47, 1 (2018), 208–217. https://doi.org/10.1109/FOCS.2016.38 Earlier version in FOCS 2016.

[25] Russell Impagliazzo, Toniann Pitassi, and Alasdair Urquhart. 1994. Upper and Lower Bounds for Tree-Like Cutting Planes Proofs. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science, LICS*. IEEE Computer Society, 220–228. https://doi.org/10.1109/LICS.1994.316069

[26] Dmitry Itsykson and Dmitry Sokolov. 2020. Resolution over linear equations modulo two. *Ann. Pure Appl. Log.* 171, 1 (2020). https://doi.org/10.1016/j.apal.2019.102722

[27] Stasys Jukna. 2012. *Boolean Function Complexity - Advances and Frontiers*. Algorithms and Combinatorics, Vol. 27. Springer. https://doi.org/10.1007/978-3-642-24508-4

[28] Stasys Jukna, Alexander A. Razborov, Petr Savický, and Ingo Wegener. 1999. On P versus NP ∩ co-NP for decision trees and read-once branching programs. *Comput. Complex.* 8, 4 (1999), 357–370. https://doi.org/10.1007/s000370050005

[29] Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. 2021. Guest Column: Models of computation between decision trees and communication. *SIGACT News* 52, 2 (2021), 46–70. https://doi.org/10.1145/3471469.3471479

[30] Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. 2021. Log-rank and lifting for AND-functions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 197–208. https://doi.org/10.1145/3406325.3450999

[31] Raghav Kulkarni and Avishay Tal. 2016. On Fractional Block Sensitivity. *Chic. J. Theor. Comput. Sci.* 2016 (2016). http://cjtcs.cs.uchicago.edu/articles/2016/8/contents.html

[32] Bruno Loff and Sagnik Mukhopadhyay. 2019. Lifting Theorems for Equality. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS (LIPIcs, Vol. 126)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 50:1–50:19. https://doi.org/10.4230/LIPIcs.STACS.2019.50

[33] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. 2022. Lifting with Sunflowers. In *13th Innovations in Theoretical Computer Science Conference, ITCS (LIPIcs, Vol. 215)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 104:1–104:24. https://doi.org/10.4230/LIPIcs.ITCS.2022.104

[34] Sagnik Mukhopadhyay, Jaikumar Radhakrishnan, and Swagato Sanyal. 2018. Separation Between Deterministic and Randomized Query Complexity. *SIAM J. Comput.* 47, 4 (2018), 1644–1666. https://doi.org/10.1137/17M1124115

[35] Noam Nisan. 1991. CREW PRAMs and Decision Trees. *SIAM J. Comput.* 20, 6 (1991), 999–1007. https://doi.org/10.1145/73007.73038 Earlier version in STOC'89.

[36] Ran Raz and Iddo Tzameret. 2008. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Log.* 155, 3 (2008), 194–224. https://doi.org/10.1016/j.apal.2008.04.001

[37] Michael E. Saks and Avi Wigderson. 1986. Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees. In *27th Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 29–38. https://doi.org/10.1109/SFCS.1986.44

[38] Marc Snir. 1985. Lower Bounds on Probabilistic Linear Decision Trees. *Theor. Comput. Sci.* 38 (1985), 69–82. https://doi.org/10.1016/0304-3975(85)90210-5

[39] Gábor Tardos. 1989. Query complexity, or why is it difficult to separate NP^A cap coNP^A from P^A by random oracles A? *Comb.* 9, 4 (1989), 385–392. https://doi.org/10.1007/BF02125350

[40] Hing Yin Tsang, Chung Hoi Wong, Ning Xie, and Shengyu Zhang. 2013. Fourier Sparsity, Spectral Norm, and the Log-Rank Conjecture. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 658–667. https://doi.org/10.1109/FOCS.2013.76

Received 2022-11-07; accepted 2023-02-06