Query Complexity of Search Problems

Arkadev Chattopadhyay
Tata Institute of Fundamental Research, Mumbai, India

Yogesh Dahiya
The Institute of Mathematical Sciences (a CI of Homi Bhabha National Institute), Chennai, India

Meena Mahajan
The Institute of Mathematical Sciences (a CI of Homi Bhabha National Institute), Chennai, India

Abstract

We relate various complexity measures like sensitivity, block sensitivity, certificate complexity for multi-output functions to the query complexities of such functions. Using these relations, we provide the following improvements upon the known relationship between pseudo-deterministic and deterministic query complexity for total search problems:

- We show that deterministic query complexity is at most the third power of its pseudo-deterministic query complexity. Previously, a fourth-power relation was shown by Goldreich, Goldwasser and Ron (ITCS'13).
- We improve the known separation between pseudo-deterministic and randomized decision tree size for total search problems in two ways: (1) we exhibit an $\exp(\widetilde{\Omega}(n^{1/4}))$ separation for the SearchCNF relation for random k-CNFs. This seems to be the first exponential lower bound on the pseudo-deterministic size complexity of SearchCNF associated with random k-CNFs. (2) we exhibit an $\exp(\Omega(n))$ separation for the ApproxHamWt relation. The previous best known separation for any relation was $\exp(\Omega(n^{1/2}))$.

We also separate pseudo-determinism from randomness in And and (And, Or) decision trees, and determinism from pseudo-determinism in Parity decision trees. For a hypercube colouring problem, that was introduced by Goldwasswer, Impagliazzo, Pitassi and Santhanam (CCC'21) to analyze the pseudo-deterministic complexity of a complete problem in TFNP^{dt}, we prove that either the *monotone* block-sensitivity or the *anti-monotone* block sensitivity is $\Omega(n^{1/3})$; Goldwasser et al. showed an $\Omega(n^{1/2})$ bound for *general* block-sensitivity.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Models of computation

Keywords and phrases Decision trees, Search problems, Pseudo-determinism, Randomness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.34

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/039/` [3]

1 Introduction

The question of whether randomness adds computational power over determinism, and if so, how much, has been a question of great interest that is still not completely understood. Naturally, the answer depends on the computational model under consideration, but it also depends on the type of problems one hopes to solve. One may wish to compute some function of the input, a special case being decision problems where the function has just two possible values. There are also search problems, where for some fixed relation $R \subseteq X \times Y$ and an input $x \in X$, one wishes to find a $y \in Y$ that is related to x; i.e. $(x, y) \in R$. If every $x \in X$ has at least one such y, we have a total search problem defined by R, the R-search problem. In the context of (total) search problems, a nuanced usage of randomness led to the beautiful notion of pseudo-determinism; see [7]. A function f solves the R-search problem if for every x, $(x, f(x)) \in R$. A randomized algorithm which computes such an f

with high probability is said to be a pseudo-deterministic algorithm solving the R-search problem. Thus a pseudo-deterministic algorithm uses randomness to solve a search problem and almost always provides a canonical solution per input.

The original papers introducing pseudo-determinism examined both polynomial-time algorithms and sublinear-time algorithms; in the latter case, the computational resource measure is query complexity. In [8], a maximal separation was established between pseudo-deterministic and randomized query algorithms; $\Omega(n)$ vs $O(1)$. Very recently, in [9], this separation was revisited. The separating problems in [8] do not lie in the query-complexity analogue of NP (nondeterministic polylog query complexity, or polylog query complexity to deterministically verify a solution, TFNP^{dt}). This is a very natural class of search problems, and in [9], an almost-maximal separation between randomized and pseudo-deterministic search is established for a problem in this class. The problem in question is SearchCNF: given an assignment to the variables of a highly unsatisfiable k-CNF formula, to search for a falsified clause; this problem is very easy for randomized search ($O(1)$ queries), and solutions are easily verifiable. Theorem 7 of [9] establishes that for unsatisfiable k-CNF formulas on n variables with sufficiently strong expansion in the clause-variable incidence graph (in particular, for most random k-CNF formulas), the corresponding search problem has pseudo-deterministic complexity $\Omega(\sqrt{n})$, even in the quantum query setting. In [9], the size measure of decision trees in the pseudo-deterministic setting was also studied. Lifting the query separation using a small gadget, a strong separation between randomized size and pseudo-deterministic size was obtained: SearchCNF problem on random k-CNFs lifted with 2-bit Xor has randomized size $O(1)$ but require pseudo-deterministic size $\exp(\Omega(\sqrt{n}))$.

Taking this study further, Theorem 3 of [9] shows that the promise problem PromiseFIND1, of finding a 1 in an n-bit string with Hamming weight at least $n/2$, is in a sense complete for the class of search problems that are in TFNP^{dt} and have efficient randomized query algorithms. By relating this search problem to a certain combinatorial problem concerning colourings of the hypercube, and by using the lower bound for SearchCNF, a lower bound of $\Omega(\sqrt{n})$ on the pseudo-deterministic complexity of PromiseFIND1 is obtained (Theorem 14 and subsequent remark in [9]). The colouring problem on hypercubes states that any proper coloring of the hypercube contains a point with many 1s and with high block sensitivity. In [9], a point with block sensitivity $\Omega(\sqrt{n})$ is proven to exist (Theorem 14), and a point with block sensitivity $\Omega(n)$ is conjectured to exist (Conjecture 16).

Our contributions

(1) We improve upon the known relationship between pseudo-deterministic query complexity and deterministic query complexity for total search problems: We show that deterministic query complexity is at most the third power of its pseudo-deterministic query complexity. (Previously a fourth-power relation was shown in [8].)

(2) We improve the known separations between pseudo-deterministic and randomized decision tree size in two ways: (1) an $\exp(\widetilde{\Omega}(n^{1/4}))$ separation for the SearchCNF relation for random k-CNFs (the $\exp(\Omega(n^{1/2}))$ separation in [9] is only for the lifted formulas k-CNF composed with Xor), and (2) an $\exp(\Omega(n))$ separation for the ApproxHamWt relation (the previous best separation for any relation was $\exp(\Omega(n^{1/2}))$).

(3) We separate pseudo-deterministic and randomized query complexity in And and (And, Or) decision trees, and show that deterministic and pseudo-deterministic complexity are polynomially related in these models, upto polylog$n$ factors. In the Parity decision tree model, we observe that deterministic and pseudo-deterministic query complexities are well separated.

(4) For the hypercube colouring problem posed in [9], we prove that either the monotone block-sensitivity or the anti-monotone block sensitivity is $\Omega(n^{1/3})$; previously an $\Omega(n^{1/2})$ bound was known but only for general block-sensitivity.

Significance, context, and techniques

We now describe each of our contributions, with surrounding context, in more detail.

For Boolean functions, randomized and deterministic query complexities are known to be polynomially related by the classic result of Nisan [14]. Since deterministic query lower bounds are often easy to obtain using some kind of adversary argument, this provides a route to randomized query lower bounds for Boolean functions. For search problems, however, there is no such polynomial relation. Note that separating pseudo-determinism from randomness requires a lower bound against randomized query algorithms that provide canonical solutions. Such algorithms compute multi-output functions (following nomenclature from [9]) as opposed to Boolean functions. Thus what is required is randomized query lower bounds for multi-output functions. For such functions, too, lower bounds for deterministic querying are often relatively easy to obtain. And again, as for Boolean functions, deterministic and randomized query complexity for multi-output functions are known to be polynomially related; in [8] (Theorem 4.1(3)), the authors show that the deterministic query complexity is bounded above by the fourth power (as opposed to cubic power for Boolean functions) of the randomized complexity. They also show that it is bounded above by the cubic power times a factor that depends on the size of the search problem's range. We revisit these relations, and further tighten them to a cubic power relation. Thus for search problems, deterministic query complexity is bounded above by the cubic power of its pseudo-deterministic query complexity; Theorem 3.2. We show this by relating various complexity measures like sensitivity, block sensitivity, certificate complexity for multi-output functions to their query complexities; Theorem 3.1.

Using the recent result from [4] that derandomized the size measures for total Boolean functions, we establish a polynomial relationship between the log of pseudo-deterministic size and the log of deterministic size, ignoring polylog factors in the input dimension; Theorem 4.3. This gives us another way to separate randomized size from pseudo-deterministic size: any total search problem which is easy with randomization but difficult for deterministic search will lead to a separation between pseudo-deterministic size and randomized size; one such problem is SearchCNF on suitably expanding k-CNF formulas. In [9], it was shown that SearchCNF for such formulas *lifted* by small gadgets like XOR, has large pseudo-deterministic size complexity. There are known situations where the complexity of a formula and its lift by small gadgets widely vary in search problems. For instance, it was known that proving the unsatisfiability of formulas corresponding to Tseitin lifted by a small gadget should be hard in cutting planes proof system [6]. The popular belief was that such hardness extends to even unlifted Tseitin formulas. In a breakthrough work [5], this belief was proven false! However, we are able to obtain an $\exp(\widetilde{\Omega}(n^{1/4}))$ lower bound on the pseudo-deterministic size complexity for SearchCNF with unlifted random k-CNF formulas, in contrast to the bounds from [9]. As far as we know, this is the first exponential lower bound on the pseudo-deterministic size complexity of SearchCNF for random k-CNF formulas. Like Tseitin formulas, determining the complexity of random k-CNF formulas in various models remains an important theme of current research.

We also show, see Theorem 4.5, that any completion of the promise-problem ApproxMaj by a total Boolean function, requires large randomized decision tree size. Observing that this promise-problem is "embedded" in the ApproxHamWt search problem, we obtain an $\exp(\Omega(n))$ separation between the pseudo-deterministic and randomized size complexity of ApproxHamWt, in Theorem 4.6.

The more general query models we consider are those of AND (OR) decision trees, abbreviated as ADT's (ODT's), where each query is a conjunction of variables, (AND, OR) decision trees, where each query is either a conjunction or a disjunction of variables, and PARITY decision trees, where each query reports the parity of some subset of variables. These models obviously generalize decision trees, and are more powerful in the deterministic setting, appear naturally in contexts like combinatorial group testing and other contexts. More recently, they have been advocated by [11] as a meaningful intermediate model between query and communication complexity. For AND and (AND, OR) decision trees, we show that pseudo-determinism is still separated from randomness; Theorems 5.3 and 5.6. To show the former, we relate randomized query complexity for multi-output functions in this model to monotone block sensitivity. To show the latter, we note that a recently proved result from [4] relating depth in (AND, OR) trees and size in ordinary trees for Boolean functions, also holds for multi-output functions. Furthermore, using other results from [4] that derandomized the AND and (AND, OR) decision trees for total Boolean functions, we observe that pseudo-determinism and determinism are polynomially related in these settings, ignoring polylog$n$ factors; Theorems 5.4 and 5.7. For PARITY decision trees, in contrast, we observe that determinism is separated from pseudo-determinism; Theorem 5.8. There is no polynomial relation between these two complexity measures. In this setting, we do not know whether pseudo-determinism is separated from randomness.

Finally, we revisit the hypercube coloring problem from [9]. There, the existence of a point with large Hamming weight and block-sensitivity $\Omega(\sqrt{n})$ is established, using the previously established lower bound for SEARCHCNF. We give a completely combinatorial and constructive argument to show that a point with large Hamming weight and block-sensitivity $\Omega(n^{1/3})$ exists, Theorem 6.3. While we seemingly sacrifice stronger bounds in the quest for simplicity, our algorithm actually proves something that is stronger in a different way, and hence our result is perhaps incomparable with that of [9]. The difference is that we identify many sensitive blocks that are all 1's, or many sensitive blocks that are all 0's. In other words, we show that the monotone (or anti-monotone) block sensitivity is $\Omega(n^{1/3})$. Monotone block sensitivity was used recently, first by [12] and then by [4], to prove query complexity lower bounds for ADT's. In particular, our result implies that every function that solves PROMISEFIND1, requires large depth to be implemented by *either* randomized ADT's *or* by randomized ODT's. We believe that this could be strengthened to show that such solutions are always hard for randomized ADTs [1]. Proving such a result is an interesting open problem.

Related work

For Boolean functions, the relations between many complexity measures and query complexity has been studied extensively in the literature. A consolidation of many known results appears in [2] as well as in [10].

Organisation of the paper

In Section 3 we establish the relationships between various measures for multi-output functions, and establish the polynomial relation between pseudo-deterministic and deterministic query complexity for search problems. In Section 4 we establish relations between pseudo-deterministic size and deterministic size. Section 5 discusses the complexity of search problems

[1] Note that there are solutions that are easy for ODTs, even deterministically. For instance, a binary search can be implemented to find efficiently the first occurrence of a 1.

in And, (And, Or), and Parity decision trees. Section 6 discusses the hypercube coloring problem. We refer the reader to the full version of our paper [3] for proofs omitted due to space constraints.

2 Preliminaries

We use standard notation and terminology in the paper, which we briefly explain here. For detailed definitions, we refer the reader to the full version of our paper [3]. For $x \in \{0,1\}^*$, and $b \in \{0,1\}$, $|x|$ is the length of x and $|x|_b$ is the number of occurrences of b in x. We also write $\mathrm{wt}(x)$ for $|x|_1$. To sample uniformly from a set S, we write $\sim_u S$. For $B \subseteq [n]$ and $b \in \{0,1\}$, b_B is the n-bit string that equals b at positions in B and $1-b$ elsewhere.

For a multi-output function $f : \{0,1\}^n \to [m]$ and input $x \in \{0,1\}^n$, we use the following measures, which are natural extensions from the Boolean case, to capture different aspects of its complexity: $\mathrm{s}(f,x)$ counts the number of input bits at x that can be flipped to change the output of $f(x)$; $\mathrm{bs}(f,x)$ is the maximum integer r for which there exist r disjoint sensitive blocks of f at x (where f is sensitive to block B on input x if $f(x) \neq f(x \oplus 1_B)$); $\mathrm{C}(f,x)$ is the minimum number of input bits required to uniquely identify the output of $f(x)$. Maximizing over all inputs gives $\mathrm{s}(f)$, $\mathrm{bs}(f)$, and $\mathrm{C}(f)$, the sensitivity, block sensitivity, and certificate complexity of f, respectively. We also use sensitivity and block sensitivity measures that only allow flipping either 0s or 1s. A set $B \subseteq [n]$ is a *sensitive b-block of f at input x* if $x_i = b$ for each $i \in B$, and $f(x) \neq f(x \oplus 1_B)$. The *$b$-block sensitivity of f at x*, denoted $\mathrm{bs}_b(f,x)$, is the maximum integer r for which there exist r disjoint sensitive b-blocks of f at x. The *b-sensitivity of f at x*, $\mathrm{s}_b(f,x)$, is the number of sensitive b-bits of x. Maximizing over all inputs gives $\mathrm{s}_b(f)$, and $\mathrm{bs}_b(f)$, the b-sensitivity and b-block sensitivity of f, respectively. Note that $\mathrm{s}_0(f)$ and $\mathrm{bs}_0(f)$ are the same as the monotone sensitivity and monotone block sensitivity used in [12] to study And-decision trees.

For a search problem $\mathcal{S}$, a (deterministic) decision tree T computing $\mathcal{S}$ is a binary tree with internal nodes labelled by the variables and the leaves labelled by $y \in \mathcal{Y}$. It evaluates an unknown input x by traversing the tree based on variable queries. The label of the leaf reached must belong to $\mathcal{S}(x)$. The depth of a decision tree T is the length of the longest root-to-leaf path, and its size is the number of leaves. The deterministic query/size complexity of $\mathcal{S}$, denoted by $\mathrm{D}^{\mathrm{dt}}(\mathcal{S})/\mathrm{DSize}^{\mathrm{dt}}(\mathcal{S})$, is defined to be the minimum depth/size of a decision tree that computes $\mathcal{S}$.

A randomized query algorithm/decision tree $\mathcal{A}$ is defined by a distribution $\mathcal{D}_{\mathcal{A}}$ over deterministic decision trees. It evaluates an input x by randomly selecting a tree T from $\mathcal{D}_{\mathcal{A}}$ and outputting the label of the leaf reached by T on x. The algorithm is considered to compute $\mathcal{S}$ with error at most ϵ if, for every x, the probability that $A(x)$ belongs to $\mathcal{S}(x)$ is at least $1-\epsilon$. The depth/size complexity of the randomized decision tree is determined by the maximum depth/size among all decision trees in the distribution's support. The randomized query/size complexity of $\mathcal{S}$ with error ϵ, denoted by $\mathrm{R}^{\mathrm{dt}}_{\epsilon}(\mathcal{S})/\mathrm{RSize}^{\mathrm{dt}}_{\epsilon}(\mathcal{S})$, is the minimum depth/size of a randomized decision tree that computes $\mathcal{S}$ with error ϵ. For a probability distribution $\mathcal{D}$ over the domain of $\mathcal{S}$, the $(\mathcal{D},\epsilon)$-distributional query/size complexity of $\mathcal{S}$, denoted by $\mathrm{D}^{\mathrm{dt}}_{\mathcal{D},\epsilon}(\mathcal{S})/\mathrm{DSize}^{\mathrm{dt}}_{\mathcal{D},\epsilon}(\mathcal{S})$, is the minimum depth/size of a deterministic decision tree that gives a correct answer on $1-\epsilon$ fraction of inputs weighted by $\mathcal{D}$. When we drop ϵ from the subscript of a randomized/distributional query measures, we assume $\epsilon = 1/3$.

A multi-output function $f : \{0,1\}^n \to [m]$ solves $\mathcal{S}$, denoted by $f \in_s \mathcal{S}$, if for every $x \in \{0,1\}^n$, $(x, f(x)) \in S$. A pseudo-deterministic query algorithm for a search problem $\mathcal{S}$, with error 1/3, is a randomized decision tree that computes some multi-output function

f that solves $\mathcal{S}$ with error at most $1/3$. The pseudo-deterministic query complexity of $\mathcal{S}$, denoted by $\mathrm{psD}^{\mathrm{dt}}(\mathcal{S})$, is equal to $\min_{f \in_s \mathcal{S}} \mathrm{R}^{\mathrm{dt}}(f)$, and pseudo-deterministic size complexity of $\mathcal{S}$, denoted by $\mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S})$, is equal to $\min_{f \in_s \mathcal{S}} \mathrm{RSize}^{\mathrm{dt}}(f)$. Note that randomized query algorithms for $\mathcal{S}$ do not necessarily output a canonical value with high probability, only a value y such that $(x, y) \in \mathcal{S}$ with high probability.

Known results that we use

▶ **Proposition 2.1** ([15]). *For a search relation $\mathcal{S}$,*
$R^{\mathrm{dt}}_{\epsilon}(\mathcal{S}) = \max_{\mathcal{D}} D^{\mathrm{dt}}_{\mathcal{D},\epsilon}(\mathcal{S})$ *and* $\mathrm{RSize}^{\mathrm{dt}}_{\epsilon}(\mathcal{S}) = \max_{\mathcal{D}} \mathrm{DSize}^{\mathrm{dt}}_{\mathcal{D},\epsilon}(\mathcal{S})$.

▶ **Proposition 2.2** ([14, 10, 2]). *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$,*
1. $s(f) \leq bs(f) \leq C(f) \leq s(f)bs(f)$.
2. $s(f) \leq bs(f) \leq 3R^{\mathrm{dt}}_{1/3}$.
3. $C(f) \leq D^{\mathrm{dt}}(f) \leq C(f)^2$.
4. $D^{\mathrm{dt}}(f) \leq C(f)bs(f)$.
5. $D^{\mathrm{dt}}(f) = O((R^{\mathrm{dt}}(f))^3)$.

▶ **Proposition 2.3** (restated from [8]). *For a search relation $\mathcal{S}$,*
1. $D^{\mathrm{dt}}(\mathcal{S}) \leq \left(psD^{\mathrm{dt}}(\mathcal{S})\right)^4$. *[Restated from Theorem 4.1(3) in [8]]*
2. $D^{\mathrm{dt}}(\mathcal{S}) \leq \left(psD^{\mathrm{dt}}(\mathcal{S})\right)^3 \ell_{\mathcal{S}}(n)$, *where $\ell_{\mathcal{S}}(n)$ is the number of bits required to represent the range of $\mathcal{S}$. [Restated from Theorem 4.1(3) in [8]]*

▶ **Proposition 2.4.**
1. (Corollary 4.2 in [8]) *For the relation*
$\textsc{ApproxHamWt} = \{(x, v) : |wt(x) - v| \leq n/10\}$,
$psD^{\mathrm{dt}}(\textsc{ApproxHamWt}) \in \Omega(n)$ *and* $R^{\mathrm{dt}}(\textsc{ApproxHamWt}) = O(1)$.
2. (Theorem 4 in [9]) *For the relation* $\textsc{PromiseFIND1} = \{(x, i) : wt(x) \geq |x|/2 \wedge x_i = 1\}$,
$psD^{\mathrm{dt}}(\textsc{PromiseFIND1}) \in \Omega(\sqrt{n})$ *and* $R^{\mathrm{dt}}(\textsc{PromiseFIND1}) = O(1)$.

▶ **Proposition 2.5.** *For F a random 3-CNF formula on n variables with $m = \Theta(n)$ clauses sampled from $\mathcal{F}^{3,n}_m$, with probability $1 - o(1)$, F is unsatisfiable and furthermore,*
1. $R^{\mathrm{dt}}(\textsc{SearchCNF}(F)) = O(1)$.
2. $D^{\mathrm{dt}}(\textsc{SearchCNF}(F)) = \Omega(n)$. *(From [13, 1])*
3. $\mathrm{DSize}^{\mathrm{dt}}(\textsc{SearchCNF}(F)) = \exp(\Omega(n))$. *(From [1])*

3 Relating measures for multi-output functions

We show the analogs of Proposition 2.2 for multi-output functions. The idea is to do the necessary modifications to the analogous results in the Boolean function case. For the proof, we refer the reader to the full version of our paper [3].

▶ **Theorem 3.1.** *For a function $f : \{0,1\}^n \to [m]$, the following relations hold.*
1. $C(f) \leq s(f)bs(f)$.
2. $s(f) \leq bs(f) \leq 3R^{\mathrm{dt}}_{1/3}(f)$
3. $C(f) \leq D^{\mathrm{dt}}(f) \leq C(f)^2$.
4. $D^{\mathrm{dt}}(f) \leq 2C(f)bs(f)$.
5. $D^{\mathrm{dt}}(f) = O((R^{\mathrm{dt}}(f))^3)$.

Using the above, we can now improve the bounds from Proposition 2.3 for search problems. One $\mathrm{psD}^{\mathrm{dt}}(\mathcal{S})$ factor from item 1 there can be removed, as also the $\ell_{\mathcal{S}}(n)$ factor in item 2.

▶ **Theorem 3.2.** *For any total search problem $\mathcal{S}$, $D^{\mathrm{dt}}(\mathcal{S}) = O((psD^{\mathrm{dt}}(\mathcal{S}))^3)$.*

Proof. For total search problem $\mathcal{S}$, let $\tilde{f}$ be a function solving $\mathcal{S}$, with $\mathrm{psD}^{\mathrm{dt}}(\mathcal{S}) = \mathrm{R}^{\mathrm{dt}}(\tilde{f})$. Then, using Theorem 3.1(5), we obtain
$\mathrm{D}^{\mathrm{dt}}(\mathcal{S}) = \min_{f \in_s \mathcal{S}} \mathrm{D}^{\mathrm{dt}}(f) \leq \mathrm{D}^{\mathrm{dt}}(\tilde{f}) \leq O((\mathrm{R}^{\mathrm{dt}}_{1/3}(\tilde{f})^3) = O(\mathrm{psD}^{\mathrm{dt}}(\mathcal{S})^3)$. ◀

4 Pseudo-deterministic size vs deterministic size

In this section, we establish a polynomial relationship, up to polylog n factors, between the logarithm of pseudo-deterministic size and the logarithm of deterministic size for total search problems, Theorem 4.3. We also improve the separation between pseudo-deterministic and randomized size, Theorem 4.6.

Before showing these results, we first examine an argument for extending results on Boolean functions to multi-output functions. We note that a relationship between randomized and deterministic complexity in a query model for Boolean functions yields an almost identical relationship between pseudo-deterministic complexity and deterministic complexity for search problems. The result follows from a straightforward application of a binary search argument and also appears in the work of [8] for making a similar claim for the ordinary query model. We give a proof sketch, for details refer to the full version of our paper [3].

▶ **Proposition 4.1.** *In a query model M, let D^{M} ($\mathrm{DSize}^{\mathrm{M}}$), R^{M} ($\mathrm{RSize}^{\mathrm{M}}$) and psD^{M} ($\mathrm{psDSize}^{\mathrm{M}}$) denote the query complexity (size complexity, respectively) in the deterministic, randomized and pseudo-deterministic settings. Then,*

1. *If for some monotonic function $q : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and every total Boolean function $f : \{0,1\}^n \to \{0,1\}$, $D^{\mathrm{M}}(f) \leq q(R^{\mathrm{M}}(f), n)$, then for any total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, $D^{\mathrm{M}}(\mathcal{S}) = O(q(psD^{\mathrm{M}}(\mathcal{S}), n) \cdot \min(\log m, psD^{\mathrm{M}}(\mathcal{S})))$.*
2. *If for some monotonic function $q : \mathbb{R} \times \mathbb{N} \to \mathbb{N}$ and every total Boolean function $f : \{0,1\}^n \to \{0,1\}$, $\log \mathrm{DSize}^{\mathrm{M}}(f) \leq q(\log \mathrm{RSize}^{\mathrm{M}}(f), n)$, then for any total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, $\log \mathrm{DSize}^{\mathrm{M}}(\mathcal{S}) = O(q(\log \mathrm{psDSize}^{\mathrm{M}}(\mathcal{S}), n) \cdot \min(\log m, \log \mathrm{psDSize}^{\mathrm{M}}(\mathcal{S})))$.*

Proof Sketch. We give a proof sketch for the second statement. For search problem $\mathcal{S}$ with $\mathrm{psDSize}^{\mathrm{M}}(\mathcal{S}) = s$, let function $\tilde{f}$ solve $\mathcal{S}$ with $\mathrm{RSize}^{\mathrm{M}}(\tilde{f}) = s$ witnessed by a randomized tree T. For $k = \lceil \log m \rceil$, define the Boolean functions $f_1, f_2, \ldots, f_k$ where $f_i(x)$ extracts the ith bit in the k-bit representation of $\tilde{f}(x)$. Then for each $i \in [k]$, $\mathrm{RSize}^{\mathrm{M}}(f_i) \leq \mathrm{RSize}^{\mathrm{M}}(\tilde{f}) = s$; simply relabel the leaves of T appropriately. By the hypothesised relation for Boolean functions, for each $i \in [\lceil \log m \rceil]$, $\log \mathrm{DSize}^{\mathrm{M}}(f_i) \leq q(\log \mathrm{RSize}^{\mathrm{M}}(f_i), n) \leq q(\log s, n)$. Let T_i be a deterministic tree achieving this size bound. By composing the trees $T_1, T_2, \ldots, T_k$ and suitably relabelling the leaves with elements from $[m]$, we obtain a deterministic tree for $\tilde{f}$ of size $(2^{q(\log s, n)})^k$. With careful analysis, we can get $\min(\log m, \log \mathrm{psDSize}^{\mathrm{M}}(\mathcal{S})$ in place of k in the exponent. ◀

Recently it was shown in [4] that for total Boolean functions, the logarithms of deterministic and randomized size are polynomially related, ignoring a polylog factor in input size.

▶ **Proposition 4.2** ([4, Theorem 3.1(b)]). *For a total Boolean function $f : \{0,1\}^n \to \{0,1\}$,*

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

Using the relation from Proposition 4.2 as the "hypothesised function" in Proposition 4.1(2), we obtain the following result, relating the log of deterministic size and the log of pseudo-deterministic size for search problems.

▶ **Theorem 4.3.** *For a total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, we have*

$$\log \mathrm{DSize}^{\mathrm{dt}}(\mathcal{S}) = O(\log^4 \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S}) \cdot \log^3(n) \cdot \min(\log m, \log \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S}))).$$

In [9] (Theorem 22), a separation was established between pseudo-deterministic and randomized size for a SearchCNF problem, defined on suitably expanding kCNF formulas *lifted* with 2-bit XOR gadgets. It was shown that the randomized size complexity of this problem is $O(1)$, while the pseudo-deterministic size complexity is $\exp(\Omega(\sqrt{n}))$. We obtain a similar (but weaker) separation for the SearchCNF problem *without any lifting*, by putting together Proposition 2.5(Item 3) and Theorem 4.3.

▶ **Corollary 4.4.** *For F a random 3-CNF formula on n variables with $m = cn$ clauses sampled from $\mathcal{F}^{3,n}_m$, with probability $1 - o(1)$, F is unsatisfiable and*

$$\mathrm{psDSize}^{\mathrm{dt}}(\textsc{SearchCNF}(F)) = \exp(\Omega(n^{1/4}/\log n)).$$

Since $\mathrm{RSize}^{\mathrm{dt}}$ of SEARCHCNF on random 3-CNF formulas is $O(1)$ w.h.p (see Proposition 2.5(Item 1)), we get a separation between $\mathrm{RSize}^{\mathrm{dt}}$ and $\mathrm{psDSize}^{\mathrm{dt}}$, albeit not as strong as [9]. However, Theorem 4.3 allows us to conclude that any total search problem separating randomized and deterministic size will yield a separation between $\mathrm{RSize}^{\mathrm{dt}}$ and $\mathrm{psDSize}^{\mathrm{dt}}$.

We now improve the separation between randomized and pseudo-deterministic sizes, from $O(1)$ vs $\exp(\Omega\sqrt{n})$ as shown in [9], to $O(1)$ vs $\exp(\Omega(n))$. To achieve this, we focus on the APPROXHAMWT problem. For this problem, a linear depth separation between randomized and pseudo-deterministic algorithms is already known from [8] (see Proposition 2.4). By using a 1-bit indexing gadget, we can lift the depth separation in APPROXHAMWT to an exponential size separation, as was done in [9, Theorem 22]. (The 1-bit indexing gadget replaces each variable x by the function $\textsc{Sel}(x^a, x^b, x^c) =$ if $x^a = 1$ then x^b else x^c.) In the rest of this section, we show that the exponential size separation between randomized and pseudo-deterministic algorithms can also be achieved using APPROXHAMWT itself *without the lift*. It is easy to see that the randomized size of APPROXHAMWT is $O(1)$. We show that its pseudo-deterministic size is $\exp(\Omega(n))$. To this end, we establish that every solution to APPROXHAMWT embeds a hard boolean function whose randomized decision tree size is exponential in the input size. This hard function is a completion of the promise problem Approximate Majority, APPROXMAJ.

APPROXMAJ is a promise problem (i.e. partial Boolean functions; certain bit strings are promised to never appear as inputs) where the task is to compute the majority of the given bit string. The promise is that the fraction of bits set to 1 in the input is either at least 3/4 or at most 1/2. A completion of APPROXMAJ is a total Boolean function that extends APPROXMAJ arbitrarily on the non-promised inputs. We show that every solution to APPROXHAMWT embeds some completion of APPROXMAJ, and that the randomized decision tree size of every completion of APPROXMAJ is exponential in the input size.

▶ **Theorem 4.5.** *For the promise problem (partial boolean function)* APPROXMAJ*,*

$$\textsc{ApproxMaj}(x) = \begin{cases} 0 & \text{if } |x| \leq n/2 \\ 1 & \text{if } |x| \geq 3n/4 \end{cases},$$

every completion f of APPROXMAJ *has* $\mathrm{RSize}^{\mathrm{dt}}(f) = \exp(\Omega(n))$.

The proof of Theorem 4.5 is based on a corruption argument and follows the template for proving randomized decision tree size lower bounds in [4, Theorem A.7]. This argument is essentially due to Swagato Sanyal, and we thank him for allowing us to include it here. Before we see its proof, let us use it to establish an exponential separation between randomized and pseudo-deterministic size forAPPROXHAMWT.

▶ **Theorem 4.6.** *Let $\mathcal{S}$ be the search problem* ApproxHamWt $= \{(x, v) \in \{0,1\}^n \times \{0, 1, ..., n\} : |wt(x) - v| \leq n/10\}$*, where $wt(x)$ is the Hamming weight of x. Then* $\text{RSize}^{\text{dt}}(\mathcal{S}) = O(1)$*, while* $\text{psDSize}^{\text{dt}}(\mathcal{S}) = \exp(\Omega(n))$*.*

Proof. The Hamming weight of a string can be estimated within a small ($\theta(n)$) additive error by querying a constant number of variables uniformly at random and outputting the scaled-up fraction of 1's seen in the queried bits. Thus $\text{RSize}^{\text{dt}}(\mathcal{S}) = O(1)$.

To show a pseudo-deterministic size lower bound, we need to show that any function that solves the ApproxHamWt problem must have randomized decision tree of size $\exp(\Omega(n))$. Let f be any function solving the ApproxHamWt problem. The total Boolean function $\bar{f} : \{0,1\}^n \to \{0,1\}$ defined as $\bar{f}(x) = 1$ iff $f(x) > 6n/10$ is a completion of ApproxMaj. Given a randomized decision tree that computes f, we can relabel the leaves appropriately to obtain a randomized decision tree that computes $\bar{f}$. Using Theorem 4.5, we conclude that $\text{RSize}^{\text{dt}}(f) \geq \text{RSize}^{\text{dt}}(\bar{f}) = \exp(\Omega(n))$. ◀

Proof of Theorem 4.5. Let f be any completion of ApproxMaj. Our strategy is to construct a hard distribution $\mathcal{D}$ on the inputs $\{0,1\}^n$ such that $\text{DSize}^{\text{dt}}_{\mathcal{D},1/3}(f) = \exp(\Omega(n))$, and then use Yao's minmax principle (see Proposition 2.1) to conclude that $\text{RSize}^{\text{dt}}(f) = \exp(\Omega(n))$. To define the hard distribution, we start by introducing some terminology. For an input $x \in \{0,1\}^n$, let $S^1_x = \{i : x_i = 1\}$ and $S^0_x = [n] \setminus S^1_x$. We say that x is 0-sensitive if all the 0s in x are sensitive with respect to f. For $x \in \{0,1\}^n$, we define the set of extreme upward neighbors of x as $\text{EUN}(x) = \{y : S^1_x \subseteq S^1_y, f(x) = f(y)$ and y is 0-sensitive$\}$. With this terminology in place, we can define the hard distribution as follows:

1. Let $rep : \{0,1\}^n \to \{0,1\}^n$ be a function which maps $x \in \{0,1\}^n$ to an arbitrary input from $\text{EUN}(x)$. Define μ_0 , a distribution over $f^{-1}(0)$ as follows: Sample an x of Hamming weight $n/2$ uniformly at random, and output $rep(x)$.
2. Define μ_1, a distribution over $f^{-1}(1)$, as follows: Sample an x according to μ_0, an index i uniformly at random from S^0_x, and return $x \oplus 1_{\{i\}}$.
3. Our hard distribution $\mathcal{D}$ is $(\mu_0 + \mu_1)/2$ i.e. with probability 1/2 return a sample from μ_0, and with probability 1/2 return a sample from μ_1.

We show below that $\text{DSize}^{\text{dt}}_{\mathcal{D},1/10}(f) = \exp(\Omega(n))$. Let T be a deterministic decision tree computing f correctly on at least 9/10-probability mass when the input is sampled according to $\mathcal{D}$. Since $\mathcal{D}$ samples with probability 1/2 from μ_0 and with probability 1/2 from μ_1, T must be correct on at least 4/5-th mass of μ_0 as well as at least 4/5 mass of μ_1. Let L_0 be set of all 0-labelled leaves(0-leaves) in T. Let ρ_0 and ρ_1 be the μ_0 and μ_1 mass captured by 0-leaves respectively; i.e.,

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

As discussed above above, $\rho_0 \geq 4/5$ and $\rho_1 \leq 1/5$.

For a leaf v, let Z_v denote the set of indices of variables fixed to zero on the path leading to v and $E_v(x)$ denote the event that the input x reaches leaf v. We will show that 0-paths with small $|Z_v|$ together capture at most 2/5 of the μ_0 mass, and 0-paths with large $|Z_v|$ individually capture exponentially small μ_0 mass. Thus to ensure that ρ_0 is large enough, there must be many 0-leaves.

1. (0-paths with few 0's). Firstly, we show that 0-paths which see less than $\lceil n/8 \rceil$ 0's must capture no more than 2/5-th mass of μ_0, i.e.,

$$\rho^0_0 = \sum_{\substack{v \in L_0 \\ |Z_v| < \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[x \text{ reaches } v] \leq 2/5.$$

This follows from the sensitivity property of μ_0. Specifically, each y in the support of μ_0 has a Hamming weight less than $3n/4$, and since all the 0s in y are sensitive, each y in the support of μ_0 has 0-sensitivity of at least $n/4$. Therefore, if a 0-path has not observed many 0s, the corresponding leaf will also capture a significant amount of μ_1 mass. Formally, consider a subcube Q corresponding to a 0-leaf with less than $\lceil n/8 \rceil$ variables fixed to 0. Due to the sensitivity property of μ_0, each x supported by μ_0 has at least $n/4$ sensitive 0s. Hence, any x supported by μ_0 that lies in Q has at least half of its total 0s unfixed. By flipping any of these 0s, we obtain an input supported by μ_1 that still lies in Q. Therefore,

$$1/5 \geq \rho_1 \geq \sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{x \sim \mu_1}[E_v(x)] = \sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{\substack{x \sim \mu_0 \\ i \sim_u S_x^0}}[E_v(x \oplus 1_{\{i\}})]$$

$$\geq \sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{\substack{x \sim \mu_0 \\ i \sim_u S_x^0}}[E_v(x) \text{ and } i \notin Z_v] = \sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[E_v(x)] \Pr_{\substack{x \sim \mu_0 \\ i \sim_u S_x^0}}[i \notin Z_v | E_v(x)]$$

$$\geq \sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[E_v(x)] \cdot \left(\frac{|S_x^0| - n/8}{|S_x^0|} \right) \geq \frac{1}{2} \cdot \left(\sum_{\substack{v \in L_0 \\ |v|_0 < \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[E_v(x)] \right) = \frac{1}{2}\rho_0^0.$$

(In the last inequality, we use the fact that $|S_x^0| > n/4$.) Hence $\rho_0^0 \leq 2/5$.

2. (0-paths with lots of 0's). Secondly, we show that a 0-path which sees more than $\lceil n/8 \rceil$ 0's can capture at most $\kappa = \exp(-\Omega(n))$ of μ_0 mass. Consider a leaf v labelled 0 such that the path leading to v fixes $t \geq \lceil n/8 \rceil$ variables to 0; $|Z_v| = t \geq n/8$. Let $S_{n/2}$ be all strings of Hamming weight $n/2$. We have

$$\kappa = \Pr_{y \sim \mu_0}[y \text{ reaches } v] = \Pr_{x \sim_u S_{n/2}}[rep(x) \text{ reaches } v]$$

$$\leq \Pr_{x \sim_u S_{n/2}}[S^1_{rep(x)} \cap Z_v = \emptyset] \leq \Pr_{x \sim_u S_{n/2}}[S_x^1 \cap Z_v = \emptyset] \quad (\text{because } S_x^1 \subseteq S^1_{rep(x)})$$

$$\leq \frac{\binom{n-t}{n/2}}{\binom{n}{n/2}} \leq \frac{\binom{7n/8}{n/2}}{\binom{n}{n/2}} = \prod_{i=0}^{n/2-1} \frac{7n/8 - i}{n - i} \leq \left(\frac{7}{8}\right)^{n/2} = 2^{-\Omega(n)}.$$

With these two observations, we can now obtain the desired lower bound.

$$4/5 \leq \rho_0 = \sum_{v \in L_0} \Pr_{x \sim \mu_0}[E_v(x)] = \sum_{\substack{v \in L_0 \\ |Z_v| < \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[E_v(x)] + \sum_{\substack{v \in L_0 \\ |Z_v| \geq \lceil n/8 \rceil}} \Pr_{x \sim \mu_0}[E_v(x)]$$
$$\leq 2/5 + \kappa \times (\text{number of 0-leaves}).$$

Hence the number of 0-leaves is at least $2/(5\kappa) = \exp(\Omega(n))$. ◀

5 More general decision trees

A variable is queried at each node of a decision tree. Generalising the class of permitted queries gives rise to many variants of decision trees that have been considered in different contexts. In this section, we consider three such classes.

- AND-*decision trees (ADT)*: queries are restricted to AND of non-negated variables.
- (AND, OR)-*decision trees*: queries are restricted to AND or OR of variables.
- PARITY-*decision trees (PDT)*: queries are restricted to PARITY of variables.

We denote the deterministic, pseudo-deterministic and randomized query complexity in each of these types of trees as ($\mathrm{D}^{\wedge\text{-dt}}$, $\mathrm{psD}^{\wedge\text{-dt}}$, $\mathrm{R}^{\wedge\text{-dt}}$), ($\mathrm{D}^{(\wedge,\vee)\text{-dt}}$, $\mathrm{psD}^{(\wedge,\vee)\text{-dt}}$, $\mathrm{R}^{(\wedge,\vee)\text{-dt}}$), and ($\mathrm{D}^{\oplus\text{-dt}}$, $\mathrm{psD}^{\oplus\text{-dt}}$, $\mathrm{R}^{\oplus\text{-dt}}$), respectively.

AND, OR and PARITY are the most basic Boolean functions. It is thus natural to study the relationship between determinism, pseudo-determinism and randomization in these settings. We will use the following recently-proved results from [4].

▶ **Proposition 5.1** ([4]). *For every total Boolean function $f : \{0,1\}^n \to \{0,1\}$,*
1. $D^{\wedge\text{-dt}}(f) = O(R^{\wedge\text{-dt}}(f)^3 \log^4(n))$. *([4, Theorem 4.5])*
2. $\log \mathrm{DSize}^{\mathrm{dt}}(f)/(2\log n) \leq \mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) \leq 4 \log \mathrm{DSize}^{\mathrm{dt}}(f)$. *([4, Lemma 4.2])*
3. $\log \mathrm{RSize}^{\mathrm{dt}}(f)/(2\log n) \leq \mathrm{R}^{(\wedge,\vee)\text{-dt}}(f) \leq 4 \log \mathrm{RSize}^{\mathrm{dt}}(f)$. *([4, Lemma 4.2])*
4. $\mathrm{D}^{(\wedge,\vee)\text{-dt}}(f) = O(\mathrm{R}^{(\wedge,\vee)\text{-dt}}(f)^4 \log^7(n))$. *([4, Theorem 4.1])*

AND-decision trees

Pseudo-determinism can be separated from randomness in AND decision trees. To establish the separation, we first give a technique to prove a pseudo-deterministic lower bound using 0-block sensitivity. The following theorem generalizes Theorem 3.1(2) to AND decision trees. The same relation is proved for Boolean functions in [12], by reduction to a hard communication problem; We give a more direct proof in the full version of our paper [3] for multi-output functions by constructing a hard distribution and using Yao's minimax principle.

▶ **Theorem 5.2.** *For a multi-output function f, $R_{1/3}^{\wedge\text{-dt}}(f) \geq bs_0(f)/3$.*
For a total search problem $\mathcal{S}$, $psD_{1/3}^{\wedge\text{-dt}}(\mathcal{S}) \geq \min_{f \in_s \mathcal{S}} bs_0(f)/3$.

Using this result, we can now separate randomized and pseudo-deterministic complexity.

▶ **Theorem 5.3.** *Let $\mathcal{S}$ be the search problem* APPROXHAMWT $= \{(x, v) : |wt(x) - v| \leq n/10\}$. *Then $R^{\wedge\text{-dt}}(\mathcal{S}) = R^{\mathrm{dt}}(\mathcal{S}) = O(1)$, while $psD^{\wedge\text{-dt}}(\mathcal{S}) = \Omega(n)$.*

Proof. It is easy to see, and already noted in Corollary 4.2 of [8], that $\mathrm{R}^{\mathrm{dt}}(\mathcal{S}) = O(1)$. To show $\mathrm{psD}^{\wedge\text{-dt}}(\mathcal{S}) = \Omega(n)$, we will show that any f solving $\mathcal{S}$ must have 0-sensitivity of at least $4n/5$. This too follows the proof outline from Corollary 4.2 of [8], where a lower bound on $\mathrm{psD}^{\mathrm{dt}}$ was obtained. But using Theorem 5.2, we draw the stronger conclusion that $\mathrm{psD}^{\wedge\text{-dt}}(\mathcal{S}) \geq 4n/5$. Suppose that for some f solving $\mathcal{S}$, $\mathrm{s}_0(f) < 4n/5$. We start with $x^0 = 0^n$ and create a sequence of inputs $\langle x^i \rangle$ such that $wt(x^i) = i$ and $f(x^i) = f(0^n)$. Because f solves APPROXHAMWT, $n/10 \geq f(0^n) = f(x^1) = f(x^2) = \ldots = f(x^l) \geq l - n/10$. Thus if we are able to create such a sequence of length at least $l = n/5 + 1$, then we already have a contradiction. The only thing left is to create the sequence x^i. For $0 \leq i \leq n/5$, given x^i with $f(x^i) = f(0^n)$, we need to find a suitable x^{i+1}. Note that x^i has exactly $n - i$ 0-bit positions, of which at most $\mathrm{s}_0(f)$ are sensitive, so at least $s = n - i - \mathrm{s}_0(f)$ 0-bit positions are not sensitive. Since $\mathrm{s}_0(f) < 4n/5$ and $i \leq n/5$, $s > 0$, so x^i has at least one non-sensitive 0-bit position. Pick any such position, say j, and define $x^{i+1} = x^i \oplus 1_{\{j\}}$. Note that x^{i+1} satisfies the desired properties i.e. $f(x^{i+1}) = f(x^i) = f(0^n)$ and $wt(x^{i+1}) = i + 1$. ◀

On the other hand, using Proposition 5.1(1) with Proposition 4.1, we get a polynomial relationship between $\mathrm{psD}^{\wedge\text{-dt}}$ and $\mathrm{D}^{\wedge\text{-dt}}$.

▶ **Theorem 5.4.** *For a total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, we have*

$$D^{\wedge\text{-dt}}(\mathcal{S}) = O(psD^{\wedge\text{-dt}}(\mathcal{S})^3 \cdot \log^4(n) \cdot \min(\log m, psD^{\wedge\text{-dt}}(\mathcal{S}))).$$

(And, Or)-decision trees

The results of Proposition 5.1(2),(3) are proved in [4] only for total Boolean functions. However, the proof there is based on a syntactic argument, where the upper bound relies on a tree-balancing argument and the lower bound is obtained by opening up And and Or queries. Since the proof is syntactic, it naturally extends to multi-output functions and search problems as well. Using this exension to multi-output functions, we obtain the following relationship between $\mathrm{psDSize}^{\mathrm{dt}}$ and $\mathrm{psD}^{(\wedge,\vee)\text{-dt}}$.

▶ **Lemma 5.5.** *For a total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, we have*

$$\log \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S})/(2\log n) \le \mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}) \le 4 \log \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S}).$$

Proof. For $\mathcal{S}$, let f and g be multi-output function solving $\mathcal{S}$, with $\mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S}) = \mathrm{RSize}^{\mathrm{dt}}(f)$ and $\mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}) = \mathrm{R}^{(\wedge,\vee)\text{-dt}}(g)$ respectively. Then

$$4 \log \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S}) = 4 \log \mathrm{RSize}^{\mathrm{dt}}(f) \overset{(*)}{\ge} \mathrm{R}^{(\wedge,\vee)\text{-dt}}(f) \ge \mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S})$$
$$= \mathrm{R}^{(\wedge,\vee)\text{-dt}}(g) \overset{(*)}{\ge} \log \mathrm{RSize}^{\mathrm{dt}}(g)/(2\log n) \ge \log \mathrm{psDSize}^{\mathrm{dt}}(\mathcal{S})/(2\log n).$$

The inequalities marked (*) holds because of Proposition 5.1(3). ◀

This, along with the size separation from Theorem 4.6, gives us a separation between randomized and pseudo-deterministic query complexity in (And, Or)-decision trees.

▶ **Theorem 5.6.** *Let $\mathcal{S}$ be the search problem* ApproxHamWt*. Then $\mathrm{R}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}) = O(1)$ and $\mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}) = \Omega(n/\log n)$.*

On the other hand, using Proposition 5.1(4) along with Proposition 4.1 gives a polynomial relation between pseudo-determinism and determinism, upto polylog$n$ factors.

▶ **Theorem 5.7.** *For a total search problem $\mathcal{S} \subseteq \{0,1\}^n \times [m]$, we have*

$$\mathrm{D}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}) = O(\mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S})^4 \cdot \log^7(n) \cdot \min(\log m, \mathrm{psD}^{(\wedge,\vee)\text{-dt}}(\mathcal{S}))).$$

Parity-decision trees

For Parity decision trees, for total Boolean functions, the randomized and deterministic Parity query complexities are linearly separated: for the And and Or functions, the deterministic PDT complexity is $\Omega(n)$, whereas the randomized PDT complexity is $O(1)$. The search analogue of the Or function gives an almost linear separation between determinism and pseudo-determinism in the PDT model. See the full version of our paper [3] for details.

▶ **Theorem 5.8.** *Let $\mathcal{S}$ be the search problem* SearchOR *$= \{(x,v) : (x_v = 1)$ or $(x = 0^n \wedge v = n+1)\}$. Then $D^{\oplus\text{-dt}}(\mathcal{S}) = n$ whereas $psD^{\oplus\text{-dt}}(\mathcal{S}) = O(\log n \log\log n)$.*

Establishing a super-polynomial separation between randomness and pseudo-determinism remains open for Parity decision trees.

6 A combinatorial proof of a combinatorial problem

In [9], the authors studied the pseudo-deterministic query complexity of a promise problem (PromiseFIND1). Here the input bit string has 1s in at least half the positions, and the task is to find a 1. They observed that PromiseFIND1 is a complete problem for easily-verifiable search problems with randomized query algorithms (see Theorem 3 in [9]), and proved a $\Omega(\sqrt{n})$ lower bound on its pseudo-deterministic query complexity. They conjectured that the pseudo-deterministic query lower bound for PromiseFIND1 can be improved to $\Omega(n)$. Towards understanding the PromiseFIND1 problem better, they introduced a natural colouring problem on hypercubes which states that any proper coloring of the hypercube contains a point with many 1s and with high block sensitivity.

▶ **Definition 6.1.** *A proper coloring of the n-dimensional hypercube H_n is any function $\phi : \{0,1\}^n - \{0^n\} \longrightarrow [n]$ such that for all $\beta \in \{0,1\}^n - \{0^n\}$, $\beta_{\phi(\beta)} = 1$.*

We say a proper coloring ϕ is d-sensitive if there exists a $\beta \in \{0,1\}^n$ such that $|\beta|_1 \geq n/2$ and β has block sensitivity at least d with respect to ϕ. That is, there are d disjoint blocks of inputs, $B_1, ..., B_d$ such that for all $i \in [d]$, $\phi(\beta) \neq \phi(\beta \oplus 1_{B_i})$. The hypercube coloring problem is about proving lower bound on the (block) sensitivity of every proper coloring. In [9] it was shown that every proper coloring is $\Omega(\sqrt{n})$-sensitive.

▶ **Proposition 6.2** (Theorem 14 [9]). *Every proper coloring of H_n is $\Omega(\sqrt{n})$-sensitive.*

The hypercube coloring problem is closely related to the pseudodeterministic query complexity of PromiseFIND1. It is a straightforward observation that showing every proper coloring is d-sensitive implies a lower bound of d on the pseudo-deterministic query complexity of PromiseFIND1. To prove Proposition 6.2 in [9], the sensitivity lower bound for the search problem associated with a random unsat k-XOR formula was converted into a block sensitivity lower bound for the hypercube coloring problem.

We give a self-contained combinatorial solution to the coloring problem. Our solution shows that every proper coloring of hypercube has a $\beta \in \{0,1\}^n$ with Hamming weight $\geq n/2$ and with block sensitivity $\Omega(n^{1/3})$. In fact, either the 1-block sensitivity or the 0-block sensitivity (or both) is $\Omega(n^{1/3})$. Thus this appears incomparable with the bound from [9]. Our solution is constructive: Algorithm 1 finds the required high-weight, high-block-sensitivity point, by querying ϕ at various points.

▶ **Theorem 6.3.** *Every proper coloring ϕ of the Boolean hypercube has a $\beta \in \{0,1\}^n$ with $|\beta| \geq n/2$ satisfying $bs_0(\phi,\beta) = \Omega(n^{1/3})$ or $bs_1(\phi,\beta) = \Omega(n^{1/3})$.*

Proof. In Algorithm 1, we describe a procedure to find the required point β. To prove that the algorithm is correct, we need to prove that if it returns $\beta \in \{0,1\}^n$ and blocks $D_1, D_2, \ldots, D_r$, then

1. $\beta \in \mathcal{X}$ (i.e. β has Hamming weight at least $n/2$),
2. $D_1, D_2, \ldots, D_r$ are disjoint sensitive blocks of ϕ at β, and
3. either all these blocks are 1-blocks of β or all these blocks are 0-blocks.
4. $r = \Omega(n^{1/3})$,

Observe that by construction, for each $i \in [t+1]$ where β^i is constructed by the algorithm, β^i has 0s in B_j for $j < i$ and 1s in B_i (in fact, 1s elsewhere); hence the blocks $B_1, \ldots, B_{i-1}$ are disjoint.

Further, by construction, each complete iteration of the for loop adds fewer than t^2 positions to C: there are fewer than t blocks (otherwise the algorithm would terminate at line 9) and each block has size less than t (otherwise the algorithm would terminate at line 13). Thus, since $|C_0| = 0$, if the algorithm reaches line 15 in iteration i, then C_i has size less than $i \cdot t^2$. Hence β^{i+1} has hamming weight $n - |C_i| > n - it^2 \geq n - t^3 > n - n/2 \geq n/2$ and is in $\mathcal{X}$.

Algorithm 1 Algorithm to find the sensitive point.

```
Require: A proper coloring $\phi$. i.e. For $\mathcal{X} = \{x \in \{0,1\}^n \mid \sum_i x_i \geq n/2\}$, $\phi : \mathcal{X} \to [n]$
    satisfying $\forall x \in \mathcal{X}$, $x_{\phi(x)} = 1$.
 1: $t \leftarrow \lfloor (n/2)^{1/3} \rfloor$; $C_0 \leftarrow \emptyset$
 2: for $i$ from 1 to $t$ do
 3:     $\beta^i \leftarrow 0_{C_{i-1}}$                          ▷ Reference input to find $t$ sensitive 1-blocks.
 4:     $\ell \leftarrow \phi(\beta^i)$; $s \leftarrow \mathrm{bs}_1(\phi, \beta^i)$    ▷ $\{\ell\}$ is a 1-sensitive block of $\beta^i$, so $s \geq 1$
 5:     $B_{i,1}, B_{i,2}, ..., B_{i,s}$: disjoint, minimally-sensitive
 6:     1-blocks achieving the 1-block sensitivity $s$.
 7:     $B_i \leftarrow \cup_{j=1}^{s} B_{i,j}$                   ▷ $\ell \in B_i$
 8:     if $s \geq t$ then
 9:         return $\beta^i$ and $\{B_{i,1}, B_{i,2}, ..., B_{i,s}\}$     ▷ $\mathrm{bs}_1(\phi, \beta^i) \geq t$
10:     end if
11:     if $\max_{j \in [s]} |B_{i,j}| \geq t$ then
12:         Pick any such $j \in [s]$ with $|B_{i,j}| \geq t$.
13:         return $\beta^i \oplus 1_{B_{i,j}}$ and $\{\{k\} \mid k \in B_{i,j}\}$    ▷ $\mathrm{s}_0(\phi, \beta^i \oplus 1_{B_{i,j}}) \geq t$
14:     end if
15:     $C_i \leftarrow C_{i-1} \cup B_i$                       ▷ We show: $C_i$ forms a $\phi$-certificate for $\beta^i$
16: end for
17: $\beta^{t+1} \leftarrow 0_{C_t}$
18: return $\beta^{t+1}$ and $\{B_1, B_2, ..., B_t\}$              ▷ $\mathrm{bs}_0(\phi, \beta^{t+1}) \geq t$
```

If the algorithm terminates at line 9 in the ith iteration of the for loop, then by the choice in line 6 the returned blocks are disjoint 1-sensitive blocks of $\beta = \beta^i$, and there are at least t of them. Similarly, if the algorithm terminates at line 13 in the ith iteration of the for loop, then by minimality of the sensitive block $B_{i,j}$ chosen in line 12, each position in $B_{i,j}$ is a 0-sensitive location in $\beta = \beta^i \oplus 1_{B_{i,j}}$, and there are at least t of them.

If the algorithm terminates at line 18, then each B_i is a 0-block of $\beta = \beta^{t+1}$ and there are t such blocks. It remains to prove that each B_i is sensitive for $\beta = \beta^{t+1}$. To show this, we will first show that each C_i is a certificate for β^i, and then show that this implies each B_i is sensitive for β.

For the first part, suppose for some $i \in [t]$, C_i is not a certificate for β^i. Then there exists an $\alpha \in \mathcal{X}$ such that $\forall j \in C_i, \alpha_j = \beta^i_j$, but $\phi(\alpha) \neq \phi(\beta^i)$. Let B be the set of positions where α and β^i differ i.e. $\alpha = \beta^i \oplus 1_B$. Since α and β^i agree on C_i, B must be disjoint from C_i. Since $\phi(\beta^i) \neq \phi(\alpha) = \phi(\beta^i \oplus 1_B)$, B is a 1-sensitive block of ϕ at β^i. By the choice in line 6 at the ith iteration, β^i has no 1-sensitive blocks disjoint from the blocks $B_{i,1}, \ldots, B_{i,s}$. But B_i is precisely the union of the these blocks, and is contained in C_i, so B is disjoint from B_i, a contradiction. Hence C_i is indeed a ϕ-certificate for β^i.

For the second part, note that for each $i \in [t]$, β and β^i agree on C_{i-1} and $\beta \oplus B_i$ and β^i agree on C_i. Since C_i is a certificate for β^i, $\phi(\beta \oplus B_i) = \phi(\beta^i) = \ell$, say. By the definition of proper coloring, $\{\ell\}$ is a 1-sensitive block of β^i, and since the blocks chosen in line 6 are the maximum possible 1-sensitive blocks, $\ell \in B_i$. But $\phi(\beta) \neq \ell$ because $\beta = 0_{C_t}$ and has only 0s in B_i. Thus $\phi(\beta) \neq \phi(\beta \oplus B_i)$, and hence B_i is a 0-sensitive block for β.

Finally, by choice of t, we see that $r = \Omega(n^{1/3})$. This concludes the correctness proof. ◀

References

1 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – Resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

2 Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

3 Arkadev Chattopadhyay, Yogesh Dahiya, and Meena Mahajan. Query complexity of search problems. *Electron. Colloquium Comput. Complex.*, TR23-039, 2023. URL: https://eccc.weizmann.ac.il/report/2023/039/.

4 Arkadev Chattopadhyay, Yogesh Dahiya, Nikhil S Mande, Jaikumar Radhakrishnan, and Swagato Sanyal. Randomized versus deterministic decision tree size. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 867–880, 2023.

5 Daniel Dadush and Samarth Tiwari. On the Complexity of Branching Proofs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:35, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2020.34.

6 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 902–911, 2018.

7 Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, page 136, 2011.

8 Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science ITCS*, pages 127–138. ACM, 2013. See also ECCC Vol. 19, T.R. 12-101, 2012.

9 Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudo-deterministic query complexity of NP search problems. In Valentine Kabanets, editor, *36th Computational Complexity Conference CCC*, volume 200 of *LIPIcs*, pages 36:1–36:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

10 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

11 Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. Guest column: Models of computation between decision trees and communication. *ACM SIGACT News*, 52(2):46–70, 2021.

12 Alexander Knop, Shachar Lovett, Sam McGuire, and Weiqiang Yuan. Log-rank and lifting for and-functions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 197–208, 2021.

13 László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, 1995.

14 Noam Nisan. Crew prams and decision trees. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 327–335, 1989.

15 Andrew Chi-Chih Yao. Lower bounds by probabilistic arguments (extended abstract). In *24th Annual Symposium on Foundations of Computer Science FOCS*, pages 420–428. IEEE Computer Society, 1983.