Deterministic search for CNF satisfying assignments in almost polynomial time

Rocco A. Servedio[*]
Columbia University

Li-Yang Tan[†]
Toyota Technological Institute

September 15, 2017

Abstract

We consider the fundamental derandomization problem of deterministically finding a satisfying assignment to a CNF formula that has many satisfying assignments. We give a deterministic algorithm which, given an n-variable poly(n)-clause CNF formula F that has at least $\varepsilon 2^n$ satisfying assignments, runs in time

$$n^{\tilde{O}(\log\log n)^2}$$

for $\varepsilon \geq 1/\text{polylog}(n)$ and outputs a satisfying assignment of F. Prior to our work the fastest known algorithm for this problem was simply to enumerate over all seeds of a pseudorandom generator for CNFs; using the best known PRGs for CNFs [DETT10], this takes time $n^{\tilde{\Omega}(\log n)}$ even for constant ε. Our approach is based on a new general framework relating deterministic search and deterministic approximate counting, which we believe may find further applications.

[*]Supported by NSF grants CCF-1420349 and CCF-1563155. Email: `rocco@cs.columbia.edu`

[†]Supported by NSF grant CCF-1563122. Part of this research was done during a visit to Columbia University. Email: `liyang@cs.columbia.edu`

1 Introduction

Understanding the role of randomness in efficient computation has been a major focus of complexity theory over the past several decades. In particular, much effort has been dedicated to developing general techniques for *unconditional derandomization*, i.e. methods of constructing efficient deterministic algorithms (that do not rely on any unproven hardness assumptions) for computational problems that are known to have efficient randomized algorithms. Notable successes have been achieved in this line of work: pseudorandom generators with highly non-trivial seed length, and much-faster-than-brute-force deterministic approximate counting algorithms, are now known for many function classes such as those defined by logarithmic space, small-depth circuits, sparse and low-degree $\mathbb{F}_2$ polynomials, various classes of branching programs, functions of a few halfspaces, low-degree polynomial threshold functions, and more (see e.g. [AW85, Nis91, LVW93, NW94, LV96, SZ99, Tre04, Bra10, RS10, GOWZ10, DGJ+10, DKN10, GKM+11, GMR+12, IMZ12, Kan12, MZ13, TX13, DS14, BRRY14, HS16] and many other works).

While striking progress has thus been made, there remain fundamental gaps in our understanding of the overarching question in unconditional derandomization: can every randomized algorithm be made deterministic with only a polynomial slowdown? In particular, while highly non-trivial results have been achieved for the classes mentioned above, a "full derandomization"—i.e. a deterministic algorithm running in *polynomial time*, as opposed to, say, quasipolynomial time—remains elusive even for some of the simplest classes of functions. (Even for the class of linear threshold functions, a full derandomization was only achieved in relatively recent work [RS10, GKM+11].)

The question we consider. Perhaps the most basic full derandomization problem that remains open is the *CNF search problem*:

> *Input:* An n-variable M-clause CNF formula F that is promised to have many, say at least $\varepsilon 2^n$, satisfying assignments.
>
> *Goal:* Output any satisfying assignment of F.

Using randomness it is easy to find a satisfying assignment with high probability simply by sampling $O(1/\varepsilon)$ many assignments and evaluting F on each one. Is there a polynomial-time *deterministic* algorithm? This problem was first considered by Ajtai and Wigderson in their pioneering work [AW85] on unconditional derandomization, in which they gave the first non-trivial (subexponential-time) deterministic algorithm for the problem.

1.1 Prior results and related work

We briefly recall the prior state of the art for this and related problems.

Pseudorandom generators and hitting sets for CNFs. Prior to our work the fastest known algorithm was simply to enumerate over all seeds of a pseudorandom generator G that ε-fools the class of M-clause n-variable CNF formulas; the definition of a pseudorandom generator immediately implies that some seed string y will have $F(G(y)) = 1$. Using the best known construction of ε-PRGs for M-clause n-variable CNFs [DETT10], this gives an algorithm running in time $\mathrm{poly}(n) \cdot (M/\varepsilon)^{\tilde{O}(\log(M/\varepsilon))}$. We observe that this PRG-based approach is oblivious to the input formula F, and can be used even if F is only provided as a black-box oracle instead of an explicit CNF formula. While this may be viewed as an advantage, it also suggests that non-oblivious approaches which exploit the structure of the input formula F may be able to achieve faster runtimes. We further observe that only an ε-hitting set for CNFs rather than an ε-PRG is required for this oblivious approach, but the best known explicit construction of hitting sets for general CNFs is simply the [DETT10] PRG. We recall that a seemingly-modest improvement of the [DETT10] PRG's seed length from $\tilde{O}(\log^2(M/\varepsilon))$ to $O(\log^{1.99}(M/\varepsilon))$, even for ε-hitting sets, would improve state-of-the-art lower bounds against depth-three circuits, breaking a longstanding barrier in circuit complexity. (For the special case of *read-once* CNF formulas, Síma and Zák [SZ10] have given an ε-hitting set of poly(n) size for $\varepsilon > 5/6$, and Gopalan et al. [GMR+12] have given an ε-PRG with seed length $\tilde{O}(\log(n/\varepsilon))$.)

The work of Goldreich and Wigderson. Recently, Goldreich and Widgderson [GW14] initiated the study of deterministic search in the regime where ε is extremely close to 1, a relaxation of the standard regime where we typically think of $\varepsilon = 1/2$ or $\varepsilon = o(1)$. As one of their main results, they give a polynomial-time deterministic search algorithm for AC^0 circuits when $\varepsilon \geq 1 - 2^{n^{0.99}}/2^n$. For the special case of M-clause n-variable CNF formulas (the subject of this work), they observe that if $\varepsilon \geq 1 - 1/(4M)$ then any $\delta = 1/(4M)$-biased sample space over $\{0,1\}^n$ must contain a satisfying assignment of F. Since well-known deterministic algorithms [NN93, AGHP92] can enumerate all poly(n/δ) elements of such a sample space in poly(n/δ) time, this gives a poly(n, M) time algorithm in this special case. (As they note in their paper, this observation is already implicit in the work of [GMR+12].)

Deterministic approximate counting and answering Trevisan's question. While the PRG-based approach described above is the most efficient algorithm known for deterministic CNF search, a more efficient algorithm is known for deterministic *approximate counting* of CNF satisfying assignments. Building on early work of Luby and Veličković [LV96], Gopalan, Meka, and Reingold [GMR13] gave a deterministic algorithm which, given as input an M-clause n-variable CNF F and a parameter $\varepsilon > 0$, runs in time $(Mn/\varepsilon)^{\tilde{O}(\log\log n + \log\log M + \log(1/\varepsilon))}$ and outputs an (additive) ε-accurate estimate of the fraction of assignments that satisfy F.

Trevisan [Tre10] has remarked that it is curious that this deterministic approximate counting algorithm—which in particular yields a certificate that F has at least $\Omega(\varepsilon 2^n)$ satisfying assignments—does not yield a comparably efficient algorithm to *find* a satisfying assignment. In [Tre10] he posed the problem of developing a deterministic search algorithm running in time comparable to that of deterministic approximate counting algorithms. Our work gives a positive solution to this problem (though it should be noted that our search algorithm's exponent is roughly quadratic in the exponent of the [GMR13] counting algorithm).

1.2 Our main result and approach

We give a deterministic CNF search algorithm that runs in almost polynomial time:

Theorem 1. *There is a deterministic algorithm which, when given as input an M-clause CNF formula F over $\{0,1\}^n$ that has $|F^{-1}(1)| \geq \varepsilon 2^n$, runs in time*

$$\left(\frac{Mn}{\varepsilon}\right)^{\tilde{O}(\log\log(Mn) + \log(1/\varepsilon))^2}$$

and outputs a satisfying assignment of F.

For the case when $M = \mathrm{poly}(n)$ and $\varepsilon \geq 1/\mathrm{polylog}(n)$, the running time of our algorithm is $n^{\tilde{O}(\log\log n)^2}$. As discussed above, the previous fastest algorithm takes time $n^{\tilde{\Omega}(\log n)}$ when $M = \mathrm{poly}(n)$, even for constant ε.

Our approach is based on a new general framework for obtaining deterministic search algorithms from deterministic approximate counting algorithms (given a few additional ingredients). Roughly speaking, this approach is an extension of the generic naive reduction described in the next subsection; while the naive reduction constructs a satisfying assignment one coordinate at a time, our approach assigns a whole block of coordinates at each iteration as described in Section 1.2.2 below. We are optimistic that this framework may find further applications for other deterministic search problems.

1.2.1 Warm up: a simple and naive search algorithm based on approximate counting

To motivate our approach, we begin by considering a very simple and naive way of obtaining a deterministic search algorithm from a deterministic approximate counting algorithm. (We will specialize our discussion to the class of CNF formulas, but the generic reduction we describe here relates these two derandomization tasks for all function classes.) Suppose we have a deterministic approximate counting algorithm A_{count} for the class of CNF formulas: given as input an M-clause CNF formula F over $\{0,1\}^n$, this algorithm A_{count} runs in time $T(n, M, \delta)$ and outputs an (additive) δ-accurate estimate of $\mathbf{Pr}[F(\boldsymbol{x}) = 1]$. Then this immediately yields, in a black-box manner, a deterministic search algorithm A_{search} with the following performance guarantee: given as input an M-clause CNF formula F over $\{0,1\}^n$ that has $|F^{-1}(1)| \geq \varepsilon 2^n$, the algorithm A_{search} runs in time

$$T(n, M, \varepsilon/(4n)) \cdot 2n \tag{1}$$

and outputs a satisfying assignment of F. The argument follows the standard n-stage decision-to-search reduction; in the $(i+1)$-st stage, after the first i bits $(z_1, \dots, z_i) \in \{0,1\}^i$ have been obtained, the algorithm runs A_{count} with accuracy parameter $\delta := \varepsilon/(4n)$ both on $F(z_1, \dots, z_i, 0, x_{i+1}, \dots, x_n)$ and on $F(z_1, \dots, z_i, 1, x_{i+1}, \dots, x_n)$, and takes as the next coordinate z_{i+1} the bit corresponding to the higher output value from A_{count}. A straightforward induction shows that for all $i \in [n]$ we have

$$\mathbf{E}\left[F(z_1, \dots, z_i, \boldsymbol{x}_{i+1}, \dots, \boldsymbol{x}_n)\right] \geq \mathbf{E}\left[F(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)\right] - 2i \cdot \delta,$$

so the final string $(z_1, \dots, z_n)$ satisfies $\mathbf{E}[F(z_1, \dots, z_n)] \geq \varepsilon - \frac{\varepsilon}{2} > 0$ and hence $F(z_1, \dots, z_n) = 1$.

However, instantiating this approach with the best known deterministic approximate counting algorithm due to Gopalan, Meka, and Reingold [GMR13], which runs in time

$$T(m, M, \delta) = (Mn/\delta)^{\tilde{O}(\log\log n + \log\log M + \log(1/\delta))},$$

we see that the running time (1) evaluates to

$$(Mn/\varepsilon)^{\tilde{O}(\log(n/\varepsilon) + \log\log M)}.$$

This is $n^{\tilde{\Omega}(\log n)}$ when $M = \mathrm{poly}(n)$ (even for constant ε), which is no improvement over the trivial PRG-based algorithm. The crux of the problem with this naive approach is that we cannot afford to run the [GMR13] approximate counting algorithm to such high accuracy, $\delta = O(\varepsilon/n)$.

1.2.2 Our approach: a more efficient reduction

At the highest level, our search algorithm shares the same overall structure as the naive bit-by-bit approach sketched above. Our algorithm is recursive in nature and uncovers a satisfying assignment of F in a stage-wise manner: in each stage we run a deterministic approximate counting algorithm on subfunctions of F, and we recurse on the one for which our estimate of its fraction of satisfying assignments is the largest. However, instead of uncovering a single coordinate of a satisfying assignment per stage, our algorithm uncovers a *p fraction of the remaining coordinates* per stage where $p \gg 1/n$. (In our analysis $p = \exp(-\Theta(\log\log(Mn/\varepsilon))^2)$, though its precise value is unimportant for the rest of this high-level discussion.) Roughly speaking, this allows us to circumvent the problem highlighted above since there will be at most $p^{-1}\ln n$ many stages in total (rather than n), and so in each stage we can run the [GMR13] approximate counting algorithm with a much larger error parameter $\delta = \Omega(\varepsilon/(p^{-1}\ln n))$ instead of $\delta = O(\varepsilon/n)$.

Three main ingredients of our approach. We will describe our approach in general terms, since the overall framework is fairly versatile and could be instantiated in other contexts.

- Let $\mathcal{C}$ be the function class of interest, the class for which we would like to design a deterministic algorithm for the "$\mathcal{C}$ search problem": given as input an n-variable function $F \in \mathcal{C}$ that is promised to have at least $\varepsilon 2^n$ satisfying assignments, find a satisfying assignment. (Our analysis will assume that $\mathcal{C}$ is closed under restrictions, which holds for natural function classes including the class of M-clause CNF formulas.)
- Let $\mathcal{C}_{\text{simple}} \subseteq \mathcal{C}$ be a subclass of "simple" functions within $\mathcal{C}$.

As alluded to above, the plan is to do search for $\mathcal{C}$ recursively in stages, uncovering a satisfying assignment of $F \in \mathcal{C}$ "chunk-by-chunk". In each stage we employ three pseudorandom constructs, the first two of which are:

1. A PRG for $\mathcal{C}_{\text{simple}}$, and
2. A deterministic approximate counting algorithm A_{count} for $\mathcal{C}$.

The win of our approach over the trivial PRG-based search algorithm will rely on both (1) the simplicity of the functions in $\mathcal{C}_{\text{simple}}$ enabling PRGs of significantly shorter seed length than those known for $\mathcal{C}$, and in similar spirit, (2) the existence of an approximate counting algorithm for $\mathcal{C}$ with runtime significantly better than that of the trivial PRG-based algorithm for $\mathcal{C}$.

The third and final ingredient is a "pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\text{simple}}$ simplification lemma":

3. Pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\text{simple}}$ simplification lemma.

   Roughly speaking, such a simplification lemma says the following: there is a pseudorandom distribution $\mathcal{R}$ over restrictions such that for all $F \in \mathcal{C}$, with high probability over $\boldsymbol{\rho} \leftarrow \mathcal{R}$ the randomly restricted function $F \upharpoonright \boldsymbol{\rho}$ belongs to $\mathcal{C}_{\text{simple}}$. In more detail, this pseudorandom distribution $\mathcal{R}$ over the space of restrictions $\{0,1,*\}^n$ should have the following structure:

   (a) The set of "live" positions $\boldsymbol{L} \subseteq [n]$ (i.e. the set of $*$'s) can be sampled efficiently with seed length r_{SL}. We write $\boldsymbol{L} \leftarrow \mathcal{R}_{\text{stars}}$ to denote a draw from this pseudorandom distribution over subsets of $[n]$.

(b) Non-live positions $[n] \setminus \boldsymbol{L}$ are filled in independently and uniformly with $\{0,1\}$, and do not count against the seed length r_{SL}. We write $\boldsymbol{\rho} \leftarrow \{0,1\}^{[n]\setminus \boldsymbol{L}}$ to denote a draw of such a restriction.

We will require each subset $L \in \mathrm{supp}(\mathcal{R}_{\mathrm{stars}})$ to have size at least pn for some not-too-small $p \in (0,1)$ (equivalently, we will require $\mathcal{R}$ to be supported on restrictions that leave at least a p fraction of coordinates unfixed). As we will soon see, this is "the same p" as the p in the high-level description of our approach in the first paragraph of this subsection; the size of L corresponds exactly to the number of coordinates of a satisfying assignment that we uncover per stage.

The guarantee that we will require of this pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma is roughly as follows: for every $F \in \mathcal{C}$,

$$\mathop{\mathbf{E}}_{\boldsymbol{L}\leftarrow\mathcal{R}_{\mathrm{stars}}}\left[\mathop{\mathbf{Pr}}_{\boldsymbol{\rho}\leftarrow\{0,1\}^{[n]\setminus\boldsymbol{L}}}\left[(F \restriction \boldsymbol{\rho}) \notin \mathcal{C}_{\mathrm{simple}}\right]\right] \le \delta_{\mathrm{SL}}, \tag{2}$$

where the failure probability δ_{SL} is as small as possible. In fact, our approach does not actually require that $F \restriction \boldsymbol{\rho}$ belong to $\mathcal{C}_{\mathrm{simple}}$; it suffices for $F \restriction \boldsymbol{\rho}$ to be well-approximated by some $F' \in \mathcal{C}_{\mathrm{simple}}$ for a suitable notion of approximation ($F \restriction \boldsymbol{\rho}$ has a "δ-lower-approximator" in $\mathcal{C}_{\mathrm{simple}}$). The analysis of our CNF search algorithm will crucially exploit this relaxation of (2), but for clarity of exposition we will assume the stronger guarantee of (2) for the description of our general framework.

For $\mathcal{C}$ being the class of CNF formulas, we remark that "pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemmas" have been the subject of much research [AW85, AAI+01, IMP12, GMR13, TX13, GW14]. These simplification lemmas, more commonly referred to as *pseudorandom switching lemmas* in this context, are achieved for various notions of "simplicity", with $\mathcal{C}_{\mathrm{simple}}$ being juntas [AW85, AAI+01, IMP12, GW14], decision trees [TX13], or small-width CNF formulas [GMR13]. We remark that for all these notions of "simple" CNF formulas, there are indeed PRGs with significantly shorter seed length than the best known PRG for general CNF formulas [DETT10]. (In our analysis $\mathcal{C}_{\mathrm{simple}}$ will be the class of $(\log((\log Mn)/\varepsilon))$-width CNF formulas, as this leads to the best overall parameters in our final result.)

Going back to the general framework, we now explain how these three pseudorandom constructs—(1) PRG for $\mathcal{C}_{\mathrm{simple}}$, (2) deterministic approximate counting algorithm A_{count} for $\mathcal{C}$, and (3) pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma—fit together to give a deterministic search algorithm for $\mathcal{C}$.

A simple but crucial fact from [AW85]. At the heart of our analysis is an elementary fact about pseudorandom simplification lemmas. This fact was first stated and utilized in the influential work of Ajtai and Wigderson [AW85] giving the first non-trivial PRG for AC^0 circuits; variants of it also play a role in the more recent PRG constructions of [GMR+12, IMZ12, RSV13, TX13].

Suppose that we have a pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma satisfying (2). Fix an $L \in \mathrm{supp}(\mathcal{R}_{\mathrm{stars}})$ such that the inner probability of (2) is at most δ_{SL}. Let $\mathcal{D}$ be a distribution that δ_{PRG}-fools $\mathcal{C}_{\mathrm{simple}}$, and suppose $\mathcal{D}$ can be sampled with r_{PRG} many random bits. The simple but crucial fact from [AW85] is the following: the distribution over $\{0,1\}^n$ where

1. The coordinates in $[n] \setminus L$ are filled in with uniform random bits;

2. The coordinates in L are filled in according to the pseudorandom distribution $\mathcal{D}$, $(\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}})$-fools $\mathcal{C}$. That is, for all $F \in \mathcal{C}$,

$$\mathop{\mathbf{E}}_{\substack{\boldsymbol{x}\leftarrow\mathcal{U}\\ \boldsymbol{y}\leftarrow\mathcal{D}}}\big[F(\boldsymbol{x}_{[n]\setminus L}, \boldsymbol{y}_L)\big] = \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[F(\boldsymbol{x})\big] \pm (\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}}).$$

Given this observation of [AW85], it follows that there must exist at least one $y \in \mathrm{supp}(\mathcal{D})$ such that

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[F(\boldsymbol{x}_{[n]\setminus L}, y_L)\big] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[F(\boldsymbol{x})\big] - (\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}}).$$

Equivalently, the restriction π^* that fixes the coordinates in L according to y preserves (from below) F's fraction of satisfying assignments up to an error of $(\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}})$, by which we mean:

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[(F \upharpoonright \pi^*)(\boldsymbol{x})\big] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[F(\boldsymbol{x})\big] - (\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}}). \tag{3}$$

Note that the number of coordinates that π^* fixes is precisely the size of L, which explains why, as alluded to above, we require the pseudorandom simplification lemma to be such that every $L \in \mathrm{supp}(\mathcal{R}_{\mathrm{stars}})$ has size at least pn for some not-too-small $p \in (0, 1)$.

Our search algorithm and its analysis. Our goal in a single stage of the recursive algorithm is to find a restriction that (approximately) satisfies (3): such a restriction reduces our search space $\{0,1\}^n$ by $|\pi^{*-1}(\{0,1\})| = |L| \geq pn$ many dimensions, while ensuring that the restricted function $F \upharpoonright \pi^*$ still has "many" satisfying assignments.

To accomplish this, our search algorithm cycles through all $2^{r_{\mathrm{SL}}+r_{\mathrm{PRG}}}$ candidates π—that is, all possible restrictions fixing L according to y where $L \in \mathrm{supp}(\mathcal{R}_{\mathrm{stars}})$ and $y \in \mathrm{supp}(\mathcal{D})$—and for each candidate π, it runs the deterministic approximate counting algorithm A_{count} to estimate $\mathbf{E}[(F \upharpoonright \pi)(\boldsymbol{x})]$ to accuracy δ_{count}. It is straightforward to see that the restriction $\tilde{\pi}$ for which A_{count}'s estimate is the largest will satisfy

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[(F \upharpoonright \tilde{\pi})(\boldsymbol{x})\big] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}\big[F(\boldsymbol{x})\big] - (\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}}) - 2\delta_{\mathrm{count}}.$$

Up to an additive factor of $2\delta_{\mathrm{count}}$, this restriction $\tilde{\pi}$ is "as good as" the restriction π^* from (3). Our algorithm recurses on $F \upharpoonright \tilde{\pi}$, a function over $\{0,1\}^{\tilde{\pi}^{-1}(*)}$ where $|\tilde{\pi}^{-1}(*)| \leq (1-p)n$. The runtime of this single stage of our recursive algorithm is at most

$$2^{r_{\mathrm{SL}}+r_{\mathrm{PRG}}} \cdot T(n, \delta_{\mathrm{count}}),$$

where $T(n, \delta)$ denotes the running time of the deterministic approximate counting algorithm A_{count}, when given as input an n-variable function $F \in \mathcal{C}$ and accuracy parameter δ.

By fixing at least a p fraction of the remaining coordinates in each stage, we ensure that there are at most $p^{-1} \ln n$ many stages in total, after which all n coordinates will have been fixed to a certain assignment $x \in \{0,1\}^n$ and the algorithm terminates with x as its output. Hence, by choosing parameters so that

$$\delta_{\mathrm{SL}} + \delta_{\mathrm{PRG}} + 2\delta_{\mathrm{count}} \leq \frac{1}{2} \cdot \frac{\varepsilon}{p^{-1} \ln n},$$

we ensure that the algorithm always recurses on a subfunction that is satisfied by at least an $(\varepsilon/2)$-fraction of its assignments. In particular, this guarantees that the n-bit assignment $x \in \{0,1\}^n$ which the algorithm outputs is indeed a satisfying assignment of F. The overall runtime of the entire algorithm is

$$2^{r_{\mathrm{SL}}+r_{\mathrm{PRG}}} \cdot T(n, \delta_{\mathrm{count}}) \cdot p^{-1} \ln n.$$

1.3 Organization of this paper

In the rest of this paper we instantiate the general framework described above with $\mathcal{C}$ being the class of M-clause n-variable CNF formulas, thus establishing our main result (Theorem 1).

In Section 2 we recall the relevant definitions and state a few simplifying assumptions. In Section 3 we state the pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\text{simple}}$ simplification lemma that we will use in our context (the pseudorandom switching lemma of [GMR13], with $\mathcal{C}_{\text{simple}}$ being the class of small-width CNF formulas) and establish some of its basic properties. In Sections 4 and 5 we use an extension of the [AW85] fact, together with this pseudorandom switching lemma and a PRG for $\mathcal{C}_{\text{simple}}$, to construct a small set of restrictions that is guaranteed to contain a "good" restriction π^*, one that fixes a significant fraction of coordinates while preserving the bias of a CNF formula from below. In Section 6 we show how to use a deterministic approximate counting algorithm to search through this set and find a restriction $\tilde{\pi}$ that is "almost as good as" π^*, thus completing the description of one stage of our recursive search algorithm. Finally, in Section 7 we put the pieces together and give our overall recursive search algorithm.

2 Background and setup

For $r < n$, we say that a distribution $\mathcal{D}$ over $\{0,1\}^n$ can be *sampled efficiently with r random bits* if (i) $\mathcal{D}$ is the uniform distribution over a multiset of size exactly 2^r of strings from $\{0,1\}^n$, and (ii) there is a deterministic algorithm $\text{Gen}_{\mathcal{D}}$ which, given as input a uniform random r-bit string $\boldsymbol{x} \leftarrow \{0,1\}^r$, runs in time $\text{poly}(n)$ and outputs a string drawn from $\mathcal{D}$.

For $\delta > 0$ and a class $\mathcal{C}$ of functions from $\{0,1\}^n$ to $\{0,1\}$, we say that a distribution $\mathcal{D}$ over $\{0,1\}^n$ *δ-fools $\mathcal{C}$ with seed length r* if (a) $\mathcal{D}$ can be sampled efficiently with r random bits via algorithm $\text{Gen}_{\mathcal{D}}$, and (b) for every function $f \in \mathcal{C}$, we have

$$\left| \mathop{\mathbf{E}}_{\boldsymbol{s}\leftarrow\{0,1\}^r}[f(\text{Gen}_{\mathcal{D}}(\boldsymbol{s}))] - \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\{0,1\}^n}[f(\boldsymbol{x})] \right| \leq \delta.$$

Equivalently, we say that $\text{Gen}_{\mathcal{D}}$ is a *δ-PRG for $\mathcal{C}$ with seed length r*.

Given a function $f : \{0,1\}^n \to \{0,1\}$ and a class of functions $\mathcal{C}$ from $\{0,1\}^n$ to $\{0,1\}$, we say that *f is δ-sandwiched by $\mathcal{C}$* if there exist functions $g_\ell, g_u \in \mathcal{C}$ such that (i) $g_\ell(x) \leq f(x) \leq g_u(x)$ for all $x \in \{0,1\}^n$, and (ii) $\mathbf{E}_{\boldsymbol{x}\leftarrow\{0,1\}^n}[g_u(\boldsymbol{x}) - g_\ell(\boldsymbol{x})] \leq \delta$. The function g_ℓ (g_u, respectively) is said to be a *lower δ-approximator* (*upper δ-approximator*, respectively) for f.

Some simplifying assumptions. We first observe that we may assume without loss of generality that our algorithm is given the value of ε. This is because the algorithm can try values $\varepsilon = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \cdots$, halting when it finds a satisfying assignment, without changing the claimed asymptotic running time. We next observe that we may assume without loss of generality that the input CNF formula has $M \geq n$ many clauses. This is because if $M < n$ then we can pad F with $n - M$ clauses $(x_1 \vee \overline{x}_1), \cdots, (x_{n-M} \vee \overline{x}_{n-M})$ to obtain an equivalent formula F' with n clauses and run the algorithm on F'.

The following simple observation allows us to assume without loss of generality that the M-clause input CNF formula has width bounded by $O(\log(M/\varepsilon))$:

Observation 2 (Trimming F). *Let F be an M-clause CNF over $\{0,1\}^n$, and let F' be the CNF obtained from F by trimming each clause of width $w' > w := \log(2M/\varepsilon)$ to width exactly w (by removing an arbitrary $w' - w$ literals from the clause). Then*

1. $F'^{-1}(1) \subseteq F^{-1}(1)$.

2. $\mathbf{E}[F'(\boldsymbol{x})] \geq \mathbf{E}[F(\boldsymbol{x})] - \varepsilon/2$.

We observe that F' can be constructed deterministically from F in time $\mathrm{poly}(M, n)$.

Proof. The observation about efficiently constructing F' from F is immediate, as is part (1) since if an assignment satisfies a given clause of F' then clearly it satisfies the corresponding clause of F. Part (2) holds because each time a clause is replaced by its trimmed version, the total number of satisfying assignments is reduced by at most $2^{-w} \cdot 2^n = \frac{\varepsilon}{2M} \cdot 2^n$. □

Our algorithm will begin by trimming all wide clauses of F (of width greater than $\log(2M/\varepsilon)$) to have width exactly $\log(2M/\varepsilon)$. By Observation 2, if F is ε-satisfiable then the resulting F' remains $(\varepsilon/2)$-satisfiable, and furthermore any satisfying assignment of F' is a satisfying assignment of the original CNF F.

Combining all of the simple observations in this section, in order to prove Theorem 1 it suffices to prove the following:

Theorem 3. *There is a deterministic algorithm with the following properties: It is given as input a value $\varepsilon > 0$ and a CNF formula F over $\{0,1\}^n$ with $M \geq n$ clauses, each of width at most $O(\log(M/\varepsilon))$, such that $|F^{-1}(1)| \geq \varepsilon 2^n$. The algorithm runs in time*

$$\left(\frac{M}{\varepsilon}\right)^{\tilde{O}(\log\log M + \log(1/\varepsilon))^2}$$

and outputs a satisfying assignment of F.

In the rest of the paper we prove Theorem 3 (so the number of clauses M is assumed to be at least n throughout the rest of the paper).

3 The [GMR13] pseudorandom switching lemma

As outlined in Section 1.2.2, one of the main ingredients of our deterministic search framework is a "pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma". For $\mathcal{C}$ being the class of CNF formulas, these are more commonly known as *pseudorandom switching lemmas*—randomness efficient versions of the seminal switching lemmas [FSS84, Ajt83, Yao85, Hås86] from circuit complexity—and they have been the subject of much research [AW85, AAI+01, IMP12, GMR13, TX13, GW14].

We will use a recent pseudorandom switching lemma of Gopalan et al. [GMR13] as it leads to the best overall running time. In this pseudorandom switching lemma $\mathcal{C}_{\mathrm{simple}}$ is the class of "narrow" (width-w') CNFs. As alluded to in Section 1.2.2, this is not quite a pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma in the sense of (2): rather than showing that $F \upharpoonright \boldsymbol{\rho}$ belongs to $\mathcal{C}_{\mathrm{simple}}$ with high probability, the [GMR13] pseudorandom switching lemma only guarantees that $F \upharpoonright \boldsymbol{\rho}$ is *sandwiched* by $F_{\mathrm{upper}}, F_{\mathrm{lower}} \in \mathcal{C}_{\mathrm{simple}}$ with high probability. But as we show in the next section, the analysis we sketched in Section 1.2.2 extends to accommodate this; in fact, for our purposes it suffices for $F \upharpoonright \boldsymbol{\rho}$ just to have a lower sandwiching approximator in $\mathcal{C}_{\mathrm{simple}}$.

We recall a standard definition from pseudorandomness:

Definition 1 (p-regular distributions)**.** *A distribution $\mathcal{R}_{\mathrm{stars}}$ over subsets of $[n]$ is said to be p-*regular *if for each $i \in [n]$ we have $\mathbf{Pr}_{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}}}[i \in \boldsymbol{L}] = p$.*

Our deterministic search framework requires that the pseudorandom $\mathcal{C}$-to-$\mathcal{C}_{\mathrm{simple}}$ simplification lemma holds with respect to a distribution over restrictions with the following structure: first a draw from a pseudorandom distribution $\mathcal{R}_{\mathrm{stars}}$ selects a subset $\boldsymbol{L} \subseteq [n]$ of coordinates which will "receive $*$'s" (the $\boldsymbol{L}$ive coordinates), and then the non-$*$ coordinates $[n] \setminus \boldsymbol{L}$ are filled in uniformly at random with bits. The [GMR13] pseudorandom switching lemma satisfies this prescribed structure:

Theorem 4 (Theorem 5.3 of [GMR13], pseudorandom switching lemma)**.** *There is a universal constant $C > 0$ such that for all $w, w', \delta_{\mathrm{sand}}, \eta > 0$ and all p satisfying*

$$p \leq \frac{\eta}{(w \log(1/\delta_{\mathrm{sand}}))^{C \log w}}, \tag{4}$$

there is a p-regular distribution $\mathcal{R}_{\mathrm{stars}}$ over subsets of $[n]$ that can be sampled efficiently using r_{SL} random bits where

$$r_{\mathrm{SL}} = O((\log w)(\log n + w' \log((\log w)/\eta)) + w \log(w \log(1/\delta_{\mathrm{sand}}))) \tag{5}$$

and the following holds: for any width-w CNF F,

$$\Pr_{\substack{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}} \\ \boldsymbol{\rho} \leftarrow \{0,1\}^{[n] \setminus \boldsymbol{L}}}} [F \upharpoonright \boldsymbol{\rho} \text{ is not } \delta_{\mathrm{sand}}\text{-sandwiched by width-}w' \text{ CNFs}] \leq \delta_{\mathrm{sand}} + \eta^{w'/4}.$$

We require pseudorandom restrictions that do not put down too few $*$'s. This motivates the following corollary:

Corollary 3.1 (Condition on having sufficiently many stars)**.** *For the distribution $\mathcal{R}_{\mathrm{stars}}$ defined in Theorem 4, let $\mathcal{R}'_{\mathrm{stars}}$ denote the distribution of $\boldsymbol{L} \leftarrow \mathrm{supp}(\mathcal{R}_{\mathrm{stars}})$ conditioned on $\boldsymbol{L}$ satisfying $|\boldsymbol{L}| \geq pn/2$. Then for any width-$w$ CNF F,*

$$\Pr_{\substack{\boldsymbol{L} \leftarrow \mathcal{R}'_{\mathrm{stars}} \\ \boldsymbol{\rho} \leftarrow \{0,1\}^{[n] \setminus \boldsymbol{L}}}} [F \upharpoonright \boldsymbol{\rho} \text{ is not } \delta_{\mathrm{sand}}\text{-sandwiched by width-}w' \text{ CNFs}] \leq \frac{2(\delta_{\mathrm{sand}} + \eta^{w'/4})}{p}.$$

Proof. Since $\mathcal{R}_{\mathrm{stars}}$ is p-regular we have that $\mathbf{E}_{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}}}[|\boldsymbol{L}|] = pn$, and so

$$\Pr_{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}}}[\boldsymbol{L} \in \mathrm{supp}(\mathcal{R}'_{\mathrm{stars}})] = \Pr_{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}}}\left[|\boldsymbol{L}| \geq \frac{pn}{2}\right] \geq \frac{p}{2}.$$

Hence

$$\begin{aligned}
\Pr_{\substack{\boldsymbol{L} \leftarrow \mathcal{R}'_{\mathrm{stars}} \\ \boldsymbol{\rho} \leftarrow \{0,1\}^{[n] \setminus \boldsymbol{L}}}} [F \upharpoonright \boldsymbol{\rho} \text{ is not } \delta_{\mathrm{sand}} \cdots] &= \Pr_{\substack{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}} \\ \boldsymbol{\rho} \leftarrow \{0,1\}^{[n] \setminus \boldsymbol{L}}}} [F \upharpoonright \boldsymbol{\rho} \text{ is not } \delta_{\mathrm{sand}} \cdots \mid \boldsymbol{L} \in \mathrm{supp}(\mathcal{R}'_{\mathrm{stars}})] \\
&\leq \Pr_{\substack{\boldsymbol{L} \leftarrow \mathcal{R}_{\mathrm{stars}} \\ \boldsymbol{\rho} \leftarrow \{0,1\}^{[n] \setminus \boldsymbol{L}}}} [F \upharpoonright \boldsymbol{\rho} \text{ is not } \delta_{\mathrm{sand}} \cdots] \cdot \frac{1}{\mathbf{Pr}[\boldsymbol{L} \in \mathrm{supp}(\mathcal{R}'_{\mathrm{stars}})]} \\
&\leq \frac{2(\delta_{\mathrm{sand}} + \eta^{w'/4})}{p}.
\end{aligned}$$

□

4 Bias preservation via pseudorandom switching lemmas

An important ingredient in our approach is a simple but ingenious observation due to Ajtai and Wigderson [AW85] which we state and prove as Lemma 4.1 below. Informally, it says the following: Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function and suppose there is a partition of $[n]$ into L and $[n] \setminus L$ with the following property: with high probability over a uniform random restriction $\boldsymbol{\rho}$ fixing the coordinates in $[n] \setminus L$ and leaving the coordinates in L free, the function $F \upharpoonright \boldsymbol{\rho}$ falls into a class $\mathcal{C}_{\text{simple}}$ that is fooled by a distribution $\mathcal{D}$ over $\{0,1\}^n$. Then the pseudorandom distribution over restrictions that fixes the coordinates in L according to $\mathcal{D}$ and leaves coordinates in $[n] \setminus L$ free approximately preserves the bias of F.

In fact, in our analysis we will only require that $F \upharpoonright \boldsymbol{\rho}$ has a lower approximator in $\mathcal{C}_{\text{simple}}$. This is because for our purposes (deterministic search) it suffices to approximately preserve the bias of F only in one direction: we have to ensure that the bias of F does not decrease by too much (so that we do not lose too many or all of the satisfying assignments), but we are fine if the bias increases.

Lemma 4.1 (Implicit in [AW85])**.** *Let $F : \{0,1\}^n \to \{0,1\}$ and $L \subseteq [n]$. Fix a class $\mathcal{C}_{\text{simple}}$ of functions over $\{0,1\}^n$ and let $\mathcal{D}$ be a distribution over $\{0,1\}^n$ that δ_{PRG}-fools $\mathcal{C}_{\text{simple}}$. Suppose that*

$$\Pr_{\boldsymbol{\rho} \leftarrow \{0,1\}^{[n]\setminus L}}\left[\, F \upharpoonright \boldsymbol{\rho} \textit{ does not have a lower } \delta_{\text{sand}}\textit{-approximator in } \mathcal{C}_{\text{simple}} \,\right] \leq \delta_{\text{SL}}. \tag{6}$$

Then

$$\mathop{\mathbf{E}}_{\substack{\boldsymbol{x} \leftarrow \mathcal{U} \\ \boldsymbol{y} \leftarrow \mathcal{D}}}[F(\boldsymbol{x}_{[n]\setminus L}, \boldsymbol{y}_L)] \geq \mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\text{PRG}} + \delta_{\text{sand}} + \delta_{\text{SL}}).$$

Proof. If $F \upharpoonright \rho$ has a lower δ_{sand}-approximator $F' \in \mathcal{C}_{\text{simple}}$ then

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[(F \upharpoonright \rho)(\boldsymbol{x})] &\leq \mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[F'(\boldsymbol{x})] + \delta_{\text{sand}} && (F' \text{ is a } \delta_{\text{sand}}\text{-approximator for } F \upharpoonright \rho) \\
&\leq \left(\mathop{\mathbf{E}}_{\boldsymbol{y} \leftarrow \mathcal{D}}[F'(\boldsymbol{y})] + \delta_{\text{PRG}} \right) + \delta_{\text{sand}} && (\mathcal{D}\ \delta_{\text{PRG}}\text{-fools } F') \\
&\leq \left(\mathop{\mathbf{E}}_{\boldsymbol{y} \leftarrow \mathcal{D}}[(F \upharpoonright \rho)(\boldsymbol{y})] + \delta_{\text{PRG}} \right) + \delta_{\text{sand}}, && (F' \leq (F \upharpoonright \rho) \text{ pointwise})
\end{aligned}$$

and so

$$\begin{aligned}
\mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[F(\boldsymbol{x})] &= \mathop{\mathbf{E}}_{\boldsymbol{\rho} \leftarrow \{0,1\}^{[n]\setminus L}}\left[\mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[(F \upharpoonright \boldsymbol{\rho})(\boldsymbol{x})] \right] \\
&\leq \left(\mathop{\mathbf{E}}_{\boldsymbol{\rho} \leftarrow \{0,1\}^{[n]\setminus L}}\left[\mathop{\mathbf{E}}_{\boldsymbol{y} \leftarrow \mathcal{D}}[(F \upharpoonright \boldsymbol{\rho})(\boldsymbol{y})] \right] + \delta_{\text{PRG}} + \delta_{\text{sand}} \right) + \delta_{\text{SL}} && ((6) \text{ and above}) \\
&= \mathop{\mathbf{E}}_{\substack{\boldsymbol{x} \leftarrow \mathcal{U} \\ \boldsymbol{y} \leftarrow \mathcal{D}}}[F(\boldsymbol{x}_{[n]\setminus L}, \boldsymbol{y}_L)] + (\delta_{\text{PRG}} + \delta_{\text{sand}} + \delta_{\text{SL}}).
\end{aligned}$$

This completes the proof. □

We will apply Lemma 4.1 with $\mathcal{C}_{\text{simple}}$ being the class of width-w' CNFs (we will keep w' a free parameter for now, but looking ahead we will ultimately set $w' = \Theta(\log w + \log((\log M)/\varepsilon))$), and $\mathcal{D}$ being the distribution given by [GMR13]'s pseudorandom generator:

Theorem 5 (Theorem 3.1 of [GMR13], PRG for width-w' CNFs)**.** *The class of width-w' CNFs over $\{0,1\}^n$ can be δ_{PRG}-fooled by a distribution $\mathcal{D}_{\mathrm{PRG}}$ which can be sampled with*

$$r_{\mathrm{PRG}} = O((w')^2(\log(w'\log(1/\delta_{\mathrm{PRG}})))^2 + w'\log(w')\log(1/\delta_{\mathrm{PRG}}) + \log\log n). \tag{7}$$

random bits.

(We remark that the [DETT10] PRG for width-w' M-clause CNFs can be used in place of Theorem 5 in our analysis, and will result the same overall running time.)

5 Existence of a bias-preserving restriction π^*

We are ready to combine the results from the previous sections to prove the key structural fact underlying our search algorithm. Roughly speaking, the next lemma says that there is a small set of restrictions, all of which fix a significant fraction of coordinates, such that for every width-w CNF F there is at least one restriction in this set that approximately preserves the bias of F from below.

Lemma 5.1 (Existence of a bias-preserving restriction)**.** *For all $w, w', \delta_{\mathrm{sand}}, \delta_{\mathrm{PRG}}, \eta > 0$ and all p satisfying assumption (4) of Theorem 4, there is a distribution $\mathcal{R}_{\mathrm{gentle}}$ over restrictions in $\{0,1,*\}^n$ such that the following hold:*

1. *$\mathcal{R}_{\mathrm{gentle}}$ is uniform over a multiset of at most $2^{r_{\mathrm{SL}}+r_{\mathrm{PRG}}}$ many outcomes, where*

$$\begin{aligned} r_{\mathrm{SL}} &= O((\log w)(\log n + w'\log((\log w)/\eta)) + w\log(w\log(1/\delta_{\mathrm{sand}}))) \\ r_{\mathrm{PRG}} &= O((w')^2(\log(w'\log(1/\delta_{\mathrm{PRG}})))^2 + w'\log(w')\log(1/\delta_{\mathrm{PRG}}) + \log\log n). \end{aligned}$$

2. *$|\pi^{-1}(\{0,1\})| \ge pn/2$ for all $\pi \in \mathrm{supp}(\mathcal{R}_{\mathrm{gentle}})$.*

3. *For any width-w CNF F over $\{0,1\}^n$, there is at least one $\pi^* \in \mathrm{supp}(\mathcal{R}_{\mathrm{gentle}})$ such that*

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[(F\upharpoonright\pi^*)(\boldsymbol{x})] \ge \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}), \tag{8}$$

where

$$\delta_{\mathrm{SL}} = \frac{2(\delta_{\mathrm{sand}} + \eta^{w'/4})}{p}.$$

Proof. The distribution $\mathcal{R}_{\mathrm{gentle}}$ is defined as follows: to make a draw $\boldsymbol{\pi} \leftarrow \mathcal{R}_{\mathrm{gentle}}$,

1. Draw $\boldsymbol{L} \leftarrow \mathcal{R}'_{\mathrm{stars}}$, the distribution over subsets $L \subseteq [n]$ defined in Corollary 3.1.

2. Draw $\boldsymbol{y} \leftarrow \mathcal{D}_{\mathrm{PRG}}$, the distribution over $\{0,1\}^n$ from Theorem 5 that δ_{PRG}-fools w'-CNFs.

3. Output the restriction $\boldsymbol{\pi} \in \{0,1,*\}^n$ where

$$\boldsymbol{\pi}_i = \begin{cases} \boldsymbol{y}_i & \text{if } i \in \boldsymbol{L} \\ * & \text{otherwise.} \end{cases}$$

By Corollary 3.1 and Theorem 5, we have that $\mathcal{R}'_{\text{stars}}$ is uniform over a multiset of at most $2^{r_{\text{SL}}}$ outcomes and $\mathcal{D}_{\text{PRG}}$ is uniform over a multiset of $2^{r_{\text{PRG}}}$ many outcomes, and hence $\mathcal{R}_{\text{gentle}}$ is uniform over a multiset of at most $2^{r_{\text{SL}}+r_{\text{PRG}}}$ many outcomes. By its definition, the distribution $\mathcal{R}'_{\text{stars}}$ satisfies $|L| \geq pn/2$ for all $L \in \text{supp}(\mathcal{R}'_{\text{stars}})$, and hence $|\pi^{-1}(\{0,1\})| \geq pn/2$ for all $\pi \in \text{supp}(\mathcal{R}_{\text{gentle}})$.

It remains to justify the third claim above. For any width-w CNF F, by Corollary 3.1 there must be at least one $L \in \text{supp}(\mathcal{R}'_{\text{stars}})$ that satisfies the assumption (6) of Lemma 4.1 with $\mathcal{C}_{\text{simple}}$ being the class of width-w' CNFs and $\delta_{\text{SL}} = 2(\delta_{\text{sand}} + \eta^{w'/4})/p$. For such an L, it follows from Lemma 4.1 and the fact that $\mathcal{D}_{\text{PRG}}$ δ_{PRG}-fools $\mathcal{C}_{\text{simple}}$ that

$$\mathop{\mathbf{E}}_{\substack{\boldsymbol{x}\leftarrow\mathcal{U}\\ \boldsymbol{y}\leftarrow\mathcal{D}_{\text{PRG}}}}[F(\boldsymbol{x}_{[n]\setminus L}, \boldsymbol{y}_L)] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\text{PRG}} + \delta_{\text{sand}} + \delta_{\text{SL}}),$$

and hence

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x}_{[n]\setminus L}, y_L)] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\text{PRG}} + \delta_{\text{sand}} + \delta_{\text{SL}})$$

for at least one $y \in \text{supp}(\mathcal{D}_{\text{PRG}})$. This pair (y, L) therefore defines a restriction $\pi^* \in \text{supp}(\mathcal{R}_{\text{gentle}})$—the restriction that fixes the coordinates in L according to y—that satisfies (8), and the proof is complete. □

6 Finding π^*, or a restriction $\tilde{\pi}$ that is almost as good

To find a restriction that (approximately) satisfies (8) we will approximate the bias of $F \upharpoonright \pi$ for all candidates $\pi \in \text{supp}(\mathcal{R}_{\text{gentle}})$ using a deterministic approximate counting algorithm for CNF formulas:

Theorem 6 (Theorem 4.6 of [GMR13] (second equation before end of proof), approximate counting algorithm)**.** *There is a deterministic algorithm that runs in time*

$$T_{\text{count}} = Mn^{O(\log(w/\delta_{\text{count}}))}(\log n)^{O(w)}2^{O(w\log(w/\delta_{\text{count}})(\log\log(w/\delta_{\text{count}}))^2)}$$

and δ_{count}-approximates the bias of any M-clause width-w CNF F over $\{0,1\}^n$, i.e. it outputs a value $v \in [0,1]$ such that $|v - \mathbf{E}_{\boldsymbol{x}\leftarrow\{0,1\}^n}[F(\boldsymbol{x})]| \leq \delta_{\text{count}}$.

Combining Lemma 5.1 and Theorem 6, we get:

Corollary 6.1 (One stage of our recursive algorithm)**.** *There is a deterministic algorithm A with the following guarantee. Given as input an M-clause width-w CNF F over $\{0,1\}^n$ and parameters $w', \delta_{\text{sand}}, \delta_{\text{PRG}}, \delta_{\text{count}}, \eta > 0$ and p satisfying assumption (4) of Theorem 4,*

1. *A runs in time*

$$\exp(r_{\text{SL}}(n, w, w', \eta, \delta_{\text{sand}}) + r_{\text{PRG}}(n, w', \delta_{\text{PRG}})) \cdot T_{\text{count}}(n, M, w, \delta_{\text{count}}),$$

where r_{SL} and r_{PRG} are as defined in Lemma 5.1, and T_{count} is as defined in Theorem 6.

2. *A outputs a restriction $\tilde{\pi} \in \{0,1,*\}^n$ such that*

   (a) $|\tilde{\pi}^{-1}(\{0,1\})| \geq pn/2$,

(b) $\tilde{\pi}$ approximately preserves the bias of F from below:

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[(F\restriction\tilde{\pi})(\boldsymbol{x})] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}) - 2\delta_{\mathrm{count}},$$

where

$$\delta_{\mathrm{SL}} = \frac{2(\delta_{\mathrm{sand}} + \eta^{w'/4})}{p}.$$

Proof. The algorithm A cycles through all (at most) $2^{r_{\mathrm{SL}}+r_{\mathrm{PRG}}}$ many restrictions π in the support of the distribution $\mathcal{R}_{\mathrm{gentle}}$ defined in Lemma 5.1, and for each one uses [GMR13]'s approximate counting algorithm in Theorem 6 to approximate the bias of $F\restriction\pi$ to accuracy δ_{count}. A outputs the restriction $\tilde{\pi}$ for which its estimate of the bias of $F\restriction\tilde{\pi}$ is the largest.

The bound on the running time of A is an immediate consequence of Lemma 5.1 and Theorem 6, as is item $2(a)$ in the claim. It remains to verify that $\tilde{\pi}$ satisfies $2(b)$. By Lemma 5.1, there is at least one $\pi^* \in \mathrm{supp}(\mathcal{R}_{\mathrm{gentle}})$ satisfying

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[(F\restriction\pi^*)(\boldsymbol{x})] \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}).$$

By the correctness of [GMR13]'s approximate counting algorithm, A's estimate of the bias $F\restriction\pi^*$ is at least

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[(F\restriction\pi^*)(\boldsymbol{x})] - \delta_{\mathrm{count}} \geq \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}) - \delta_{\mathrm{count}},$$

and hence so is its estimate of the bias of $F\restriction\tilde{\pi}$. Finally, again by the correctness of [GMR13]'s approximate counting algorithm, we conclude that the true bias of $F\restriction\tilde{\pi}$ is within δ_{count} of A's estimate, and hence

$$\begin{aligned}\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[(F\restriction\pi^*)(\boldsymbol{x})] &\geq \left(\mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}) - \delta_{\mathrm{count}}\right) - \delta_{\mathrm{count}} \\ &= \mathop{\mathbf{E}}_{\boldsymbol{x}\leftarrow\mathcal{U}}[F(\boldsymbol{x})] - (\delta_{\mathrm{PRG}} + \delta_{\mathrm{sand}} + \delta_{\mathrm{SL}}) - 2\delta_{\mathrm{count}}.\end{aligned}$$

This completes the proof. ☐

6.1 Applying Corollary 6.1: setting of parameters

We first introduce two more parameters $T \in \mathbb{N}$ and $\tau \in (0,1)$ to denote

$$T := \frac{2\ln n}{p} \quad \text{and} \quad \tau := \frac{\varepsilon}{2T}.$$

Looking ahead, the semantics of T and τ are as follows: each stage of our recursive search algorithm—a call to the subroutine in Corollary 6.1—fixes at least a $p/2$ fraction of the remaining coordinates (recall item $2(a)$ of Corollary 6.1), so T is chosen so that after T stages the number of unfixed coordinates is at most

$$n \cdot (1-p/2)^T = n \cdot (1-p/2)^{(2\ln n)/p} < 1,$$

i.e. we will have arrived at an actual assignment to the CNF F. Since T is an upper bound on the number of calls to the subroutine in Corollary 6.1, we will set parameters so that the bias of F is preserved to within an additive $\tau = \varepsilon/2T$ in each call. This ensures that the bias of F remains at least

$$\varepsilon - \tau \cdot T = \frac{\varepsilon}{2} > 0$$

throughout, and hence the final assignment we arrive at is in fact a satisfying assignment of F.

With these definitions of T and τ in hand, we will invoke the algorithm in Corollary 6.1 with the following choice of parameters:

$$\begin{aligned}
p &= \left(\frac{1}{w\log((\log M)/\varepsilon)}\right)^{2C\log w}, \\
\eta &= \frac{1}{w\log((\log M)/\varepsilon)}, \\
w' &= 16C\log w + 4\log\left(\frac{192\ln M}{\varepsilon}\right),
\end{aligned}$$

where $C > 0$ is the universal constant from Theorem 4, and

$$\delta_{\text{count}} = \frac{\tau}{3}, \qquad \delta_{\text{PRG}} = \frac{\tau}{6}, \qquad \delta_{\text{sand}} = \frac{p\tau}{48}.$$

The next proposition justifies our choice of parameters:

Proposition 6.2. *For this choice of parameters, we have that*

1. *p, η, and δ_{sand} satisfy assumption (4) of Theorem 4:*

$$p \le \frac{\eta}{(w\log(1/\delta_{\text{sand}}))^{C\log w}}.$$

2. *For $\delta_{\text{SL}} = 2(\delta_{\text{sand}} + \eta^{w'/4})/p$,*

$$\delta_{\text{PRG}} + \delta_{\text{sand}} + \delta_{\text{SL}} + 2\delta_{\text{count}} \le \tau.$$

Proof. For the first claim, we note that

$$\begin{aligned}
\log\left(\frac{1}{\delta_{\text{sand}}}\right) &= \log\left(\frac{192\ln n}{\varepsilon p^2}\right) \\
&= \log((\log n)/\varepsilon) + 2\log(1/p) + O(1) \\
&= O(\log^2 w)(\log((\log M)/\varepsilon)),
\end{aligned}$$

and so indeed for w larger than a suitable absolute constant, we have

$$\begin{aligned}
\frac{\eta}{(w\log(1/\delta_{\text{sand}}))^{C\log w}} &> \frac{\eta}{(w\log((\log M)/\varepsilon))^{1.01C\log w}} \\
&= \left(\frac{1}{w\log((\log M)/\varepsilon)}\right)^{1.01C\log w+1} \\
&> \left(\frac{1}{w\log((\log M)/\varepsilon)}\right)^{2C\log w} \quad = \quad p.
\end{aligned}$$

As for the second claim, by our choice of $\delta_{\mathrm{PRG}} = \tau/6$ and $\delta_{\mathrm{count}} = \tau/3$ the claimed bound is equivalent to

$$\delta_{\mathrm{sand}} + \delta_{\mathrm{SL}} \leq \frac{\tau}{6}.$$

Since $\delta_{\mathrm{sand}} < \delta_{\mathrm{SL}}$, it suffices to ensure that

$$\delta_{\mathrm{SL}} \leq \frac{\tau}{12}, \qquad \text{or equivalently,} \qquad \delta_{\mathrm{sand}} + \eta^{w'/4} \leq \frac{p\tau}{24}.$$

Recalling our choice of $\delta_{\mathrm{sand}} = p\tau/48$, it remains to check that

$$\eta^{w'/4} \leq \frac{p\tau}{48} = \frac{\varepsilon p^2}{192 \ln n}.$$

Indeed,

$$\begin{aligned}\eta^{w'/4} &= \left(\frac{1}{w \log((\log M)/\varepsilon)}\right)^{4C \log w + \log((192 \ln M)/\varepsilon)} \\ &< \left(\frac{1}{w \log((\log M)/\varepsilon)}\right)^{4C \log w} \cdot 2^{-\log((192 \ln M)/\varepsilon)} \\ &= \frac{\varepsilon p^2}{192 \ln M} \leq \frac{\varepsilon p^2}{192 \ln n} \qquad \text{(using } M \geq n\text{)}.\end{aligned}$$

This completes the proof of the second claim. □

We note the following estimates for our choice of parameters when $w = O(\log(M/\varepsilon))$ (recall Theorem 3 and in particular that $M \geq n$):

$$\begin{aligned}\frac{1}{p} &= (\log(M/\varepsilon))^{O(\log\log(M/\varepsilon))} &(9)\\ \log(1/\eta) &= O(\log\log(M/\varepsilon)) &(10)\\ w' &= O(\log((\log M)/\varepsilon)) &(11)\\ \log(1/\delta) &= O(\log((\log M)/\varepsilon)) + O(\log\log(M/\varepsilon))^2. \quad \text{(for } \delta \in \{\delta_{\mathrm{count}}, \delta_{\mathrm{PRG}}, \delta_{\mathrm{sand}}\}\text{)} &(12)\end{aligned}$$

Proposition 6.2 yields the following special case of Corollary 6.1:

Corollary 6.3 (Corollary 6.1 for our choice of parameters)**.** *There is a deterministic algorithm A with the following guarantee. Given as input an M-clause width-w CNF F over $\{0,1\}^n$,*

1. *A runs in time*

$$\begin{aligned}&\exp(r_{\mathrm{SL}}(n, w, w', \eta, \delta_{\mathrm{sand}}) + r_{\mathrm{PRG}}(n, w', \delta_{\mathrm{PRG}})) \cdot T_{\mathrm{count}}(n, M, w, \delta_{\mathrm{count}}) \\ &= \left(\frac{M}{\varepsilon}\right)^{\tilde{O}(\log((\log M)/\varepsilon))^2}\end{aligned}$$

2. *A outputs a restriction $\tilde{\pi} \in \{0,1,*\}^n$ such that*
   (a) $|\tilde{\pi}^{-1}(\{0,1\})| \geq pn/2$,
   (b) $\mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[(F \upharpoonright \tilde{\pi})(\boldsymbol{x})] \geq \mathop{\mathbf{E}}_{\boldsymbol{x} \leftarrow \mathcal{U}}[F(\boldsymbol{x})] - \tau.$

7 Putting the pieces together: the overall search algorithm

Using the results of the previous subsections we now prove Theorem 3. The claimed algorithm is given as input a pair (F, ε); recall that from the theorem statement and as shown in Section 2, we may assume that the CNF F has $M \geq n$ clauses each of width at most $w = O(\log(M/\varepsilon))$.

The algorithm proceeds for at most $T = (2 \ln n)/p$ iterative stages (where p is as defined in Section 6.1) as follows. In the t-th stage it operates on a CNF formula $F \upharpoonright (\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^{t-1})$; the first stage is the $(t = 1)$-th stage and we take $\tilde{\pi}^0$ to be the trivial restriction which assigns $*$ to each of the n input variables, so $F \upharpoonright \tilde{\pi}^0$ is simply the input CNF F. Before starting the first stage, the algorithm records the values of parameters $w, w', \eta, \delta_{\text{sand}}$, and δ_{PRG}. (Observe that all of these values $w, w', \eta, \delta_{\text{sand}}, \delta_{\text{PRG}}$ are defined solely in terms of M and ε, see Equations (10), (11) and (12); these values will never change during the execution of the algorithm.)

Stage 1 is carried out as follows:

- Let n_1 denote the number of variables that are alive under restriction $\tilde{\pi}^0$, which in stage 1 is $n_1 = n$. The algorithm compute the seed lengths $r_{\text{SL},1} := r_{\text{SL}}(n_1, w, w', \eta, \delta_{\text{sand}})$ and $r_{\text{PRG},1} := r_{\text{PRG}}(n_1, w', \delta_{\text{PRG}})$.
- Then the algorithm executes the deterministic algorithm A from Corollary 6.3 on the n_1-variable function $F \upharpoonright \tilde{\pi}^0$. The algorithm produces a restriction $\tilde{\pi}^1 \in \{0, 1, *\}^{n_1}$ with the properties described in 2(a) and 2(b) of Corollary 6.3.

The general t-th stage of the algorithm is carried out in a similar way:

- Let n_t denote the number of variables that are alive under the restriction $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^{t-1} \in \{0, 1, *\}^n$. The algorithm computes the seed lengths $r_{\text{SL},t} := r_{\text{SL}}(n_t, w, w', \eta, \delta_{\text{sand}})$ and $r_{\text{PRG},t} := r_{\text{PRG}}(n_t, w', \delta_{\text{PRG}})$ which are appropriate for the pseudorandom switching lemma and pseudorandom generators for n_t-variable functions.
- Then the algorithm executes the deterministic algorithm A from Corollary 6.3 *on the n_t-variable CNF $F \upharpoonright (\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^{t-1})$*. The algorithm produces a restriction $\tilde{\pi}^t \in \{0, 1, *\}^{n_t}$ with the properties described in 2(a) and 2(b) of Corollary 6.3.

We may view the restriction $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t$ as belonging to $\{0, 1, *\}^n$. If $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t$ belongs to $\{0, 1\}^n$ (leaves no variables free) then the algorithm halts and outputs $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t$, otherwise it increments t and proceeds to the next stage.

It remains to establish correctness; this is easy given Corollary 6.3. A crucial aspect of the algorithm is that in the t-th stage it works on the n_t-variable CNF $F \upharpoonright (\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^{t-1})$. Thanks to part 2(a) of Corollary 6.3, this implies that each value of n_t is at most $n(1-p/2)^{t-1}$, so consequently after at most T stages the algorithm will indeed obtain a restriction $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t \in \{0, 1\}^n$ and halt as desired. For the running time of the algorithm, it follows from part (1) of Corollary 6.3 that the running time of each of the (at most) $T = (2 \ln n)/p$ stages is upper bounded by $(M/\varepsilon)^{\tilde{O}(\log((\log M)/\varepsilon))^2}$ and hence this is also an upper bound on the running time of the entire algorithm (recalling the bound on p from (9)). Finally, from the discussion at the start of Section 6.1, we have that the bias of $F(\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t)$ is greater than zero, and hence $\tilde{\pi}^0 \circ \cdots \circ \tilde{\pi}^t$ is a satisfying assignment as desired. This concludes the proof of Theorem 3.

References

[AAI$^+$01] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Comput. Complexity*, 10(2):117–138, 2001. 1.2.2, 3

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992. 1.1

[Ajt83] Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. 3

[AW85] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 11–19, 1985. 1, 1, 1.2.2, 1.2.2, 1.2.2, 1.3, 3, 4, 4.1

[Bra10] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *Journal of the ACM*, 57(5):28, 2010. 1

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. 1

[DETT10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Proceedings of the 13th International Workshop on Randomization and Computation (RANDOM)*, pages 504–517, 2010. (document), 1.1, 1.2.2, 4

[DGJ$^+$10] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco Servedio, and Emanuele Viola. Bounded independence fools halfspaces. *SIAM Journal on Computing*, 39(8):3441–3462, 2010. 1

[DKN10] Ilias Diakonikolas, Daniel Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20, 2010. 1

[DS14] Anindya De and Rocco Servedio. Efficient deterministic approximate counting for low-degree polynomial threshold functions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 832–841, 2014. 1

[FSS84] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. 3

[GKM$^+$11] Parikshit Gopalan, Adam Klivans, Raghu Meka, Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. An FPTAS for #Knapsack and related counting problems. In *52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 817–826, 2011. 1

[GMR$^+$12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings* *of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–129, 2012. 1, 1.1, 1.1, 1.2.2

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Comput. Complexity*, 22(2):275–310, 2013. 1.1, 1.2.1, 1.2.2, 1.2.2, 1.3, 3, 3, 4, 4, 5, 6, 6

[GOWZ10] Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC)*, pages 223–234, 2010. 1

[GW14] Oded Goldreich and Avi Widgerson. On derandomizing algorithms that err extremely rarely. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118. ACM, 2014. 1.1, 1.2.2, 3

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986. 3

[HS16] Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to AC^0. In *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, pages 32:1–32:14, 2016. 1

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 961–972, 2012. 1.2.2, 3

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 111–119, 2012. 1, 1.2.2

[Kan12] Daniel Kane. A structure theorem for poorly anticoncentrated Gaussian chaoses and applications to the study of polynomial threshold functions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 91–100, 2012. 1

[LV96] Michael Luby and Boban Veličković. On deterministic approximation of DNF. *Algorithmica*, 16(4-5):415–433, 1996. 1, 1.1

[LVW93] Michael Luby, Boban Veličković, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the 2nd ISTCS*, pages 18–24, 1993. 1

[MZ13] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013. 1

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991. 1

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. 1.1

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. 1

[RS10] Yuval Rabani and Amir Shpilka. Explicit construction of a small epsilon-net for linear threshold functions. *SIAM J. on Comput.*, 39(8):3501–3520, 2010. 1

[RSV13] Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM)*, pages 655–670, 2013. 1.2.2

[SZ99] Michael Saks and Shiyu Zhou. $\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *J. Comput. System Sci.*, 58(2):376–403, 1999. 1

[SZ10] Jirí Síma and Stanislav Zák. A polynomial time construction of a hitting set for read-once branching programs of width 3. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:88, 2010. 1.1

[Tre04] Luca Trevisan. A note on approximate counting for k-DNF. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM)*, pages 417–426, 2004. 1

[Tre10] Luca Trevisan. Open Problems in Unconditional Derandomization. Presentation at China Theory Week 2010, slides available at `http://conference.itcs.tsinghua.edu.cn/CTW2010/content/Slides/2.pdf`, 2010. 1.1

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC^0 . In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC)*, pages 242–247, 2013. 1, 1.2.2, 1.2.2, 3

[Yao85] Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 1–10, 1985. 3