Electronic Colloquium on Computational Complexity, Report No. 114 (2015)

#SAT Algorithms from Shrinkage

Avishay Tal *

Abstract

We present a deterministic algorithm that counts the number of satisfying assignments for any de Morgan formula F of size at most $n^{3-16\varepsilon}$ in time $2^{n-n^{\varepsilon}} \cdot \mathrm{poly}(n)$, for any small constant $\varepsilon > 0$. We do this by derandomizing the randomized algorithm mentioned by Komargodski et al. (FOCS, 2013) and Chen et al. (CCC, 2014). Our result uses the tight "shrinkage in expectation" result of de Morgan formulas by Håstad (SICOMP, 1998) as a black-box, and improves upon the result of Chen et al. (MFCS, 2014) that gave deterministic counting algorithms for de Morgan formulas of size at most $n^{2.63}$.

Our algorithm generalizes to other bases of Boolean gates giving a $2^{n-n^{\varepsilon}} \cdot \mathrm{poly}(n)$ time counting algorithm for formulas of size at most $n^{\Gamma+1-O(\varepsilon)}$, where Γ is the shrinkage exponent for formulas using gates from the basis.

*Weizmann Institute of Science, Rehovot, Israel. avishay.tal@weizmann.ac.il. Supported by an Adams Fellowship of the Israel Academy of Sciences and Humanities, by an ISF grant and by the I-CORE Program of the Planning and Budgeting Committee.

ISSN 1433-8092

1 Introduction

A de Morgan formula is a binary tree in which each leaf is labeled with a literal from $\{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$ and each internal node is labeled with either a Boolean AND or OR gate. Such a tree naturally describes a Boolean function on n variables by propagating values from leaves to root, and returning the root's value. The formula size is the number of leaves in the tree; for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ we denote by $L(f)$ the minimal size formula which computes f.

The shrinkage of de Morgan formulas under random restriction was first used by Subbotovskaya [Sub61] to prove an $\Omega(n^{3/2})$ lower bound for the formula size of the n-way parity function. A long line of works [Sub61, PZ93, IN93, Hås98, Tal14] showed that any de Morgan formula shrinks by a factor of $O(p^2)$ under random restrictions, keeping each variable "alive" with probability p, and fixing it to a random bit otherwise. This yields an $\Omega(n^2)$ lower bound on the size of any de Morgan formula computing the n-way parity. In addition, using the aforementioned shrinkage result, the explicit function $A : \{0,1\}^n \to \{0,1\}$ constructed by Andreev [And87] has formula size $\Omega(n^3/\log^{2+o(1)} n)$. This is the best known formula size lower bound for an explicit function.

In recent years, shrinkage results yielded many other applications concerning de Morgan formulas: pseudorandom generators [IMZ12], average case lower bounds [KR13, KRT13, CKK+14], #SAT algorithms [San10, CKK+14, CKS14], compression algorithms [CKK+14], and Fourier concentration results [IK14]. In some of these applications, the results are meaningful for functions computed by de Morgan formulas of size $O(n^{2.99})$, indicating that the shrinkage results were exploited to the fullest.

Designing #SAT (and SAT) algorithms for restricted families of circuits/formulas is a fundamental challenge in computational complexity. In this work, we consider #SAT algorithms for de Morgan formulas, i.e. algorithms that count the number of satisfying assignments to a given formula. Santhanam [San10] gave a #SAT algorithm for de Morgan formulas of linear size, that runs in time $2^{n-\Omega(n)}$. Seto and Tamaki [ST13] gave an algorithm with a similar running time for linear size formulas over the full binary basis. Chen et al. [CKK+14] gave a #SAT algorithm for formulas of size $n^{2.49}$, which runs in time $2^{n-n^{\Omega(1)}}$. They used the suboptimal $O(p^{3/2})$ shrinkage result of Subbotovskaya [Sub61], since it guarantees that the formula shrinks with respect to a greedy process that picks at each step the *heaviest* variable (i.e., the variable that appears in the maximum number of leaves).

Let us briefly sketch their strategy. At first, they pre-compute for all formulas of size at most $n^{1-\varepsilon}$, the number of satisfying assignments in $\exp(O(n^{1-\varepsilon}\log n))$ time, and store these values in a look-up table. Given a larger formula, their algorithm deterministically picks at each step the heaviest variable, branches according to its two possible values, simplifies both formulas, and applies recursion. The recursion stops whenever we reach a formula of size smaller than $n^{1-\varepsilon}$, in which case we return the pre-computed value.

It seems crucial for their analysis that the shrinkage result: (1) holds with respect to some deterministic choice of the variables to fix and (2) holds under constructive simplification rules, which can be computed in polynomial time. While the latter is true for Håstad's nearly tight result [Hås98], the former is not known to hold. In [CKS14] a deterministic #SAT algorithm for formulas of size $n^{2.63}$ was given using the shrinkage result of Paterson and Zwick [PZ93], which runs in time $2^{n-n^{\Omega(1)}}$ as well. The authors used the fact that Paterson and Zwick's shrinkage result holds with respect to a greedy choice of variables as well. However, they needed to show that there are constructive simplification rules under which the shrinkage result holds - which required some

additional work. Komargodski et al. [KRT13] obtained a "shrinkage with high probability" result with respect to a combination of greedy and random selection of variables. They observed that their result, combined with the algorithmic ideas of [CKK+14], implies a zero-error *randomized* algorithm for formulas of size $n^{2.99}$, which runs in time $2^{n-n^{\Omega(1)}}$.

Recently and independently to us, Chen [Che15] gave a $2^{n-n^{\Omega(1)}}$ time counting algorithm for formulas of size $n^{1+\Gamma_k-\varepsilon}$ over the basis of all unate gates of arity k, where $\Gamma_k = 1 + 1/(3k-4)$.

1.1 Our Results

We present a *deterministic* #SAT algorithm for formulas of size $n^{2.99}$ which runs in time $2^{n-n^{\Omega(1)}}$.

Theorem 1.1. *Let $\varepsilon > 0$ be a small enough constant. Then, there exists a #SAT algorithm for formulas of size at most $n^{3-16\varepsilon}$ which runs in time $2^{n-n^{\varepsilon}} \cdot \mathrm{poly}(n)$.*

Despite the two aforementioned thumb-rules, our algorithm does not assume shrinkage with respect to some deterministic choice of variables, nor does it rely on a set of constructive simplification rules. It is solely based on: (1) the promise that for any Boolean function f

$$\mathop{\mathbf{E}}_{\rho\sim\mathcal{R}_p}[L(f|_\rho)] \leq c_0 \cdot (L(f)p^2 + 1)$$

for some universal constant c_0,[1] and (2) the decomposition lemma proved by [Tal14] (based on [IMZ12]). Our result generalizes to formulas over other bases of Boolean gates, as long as the basis contains the OR and AND gates, and in particular to monotone de Morgan formulas.

Theorem 1.2. *Let $\mathcal{B}$ be a set of gates which include the* OR *and* AND *gates, and denote by Γ the shrinkage exponent of this basis.*[2] *Let $\varepsilon > 0$ be a small enough constant, then there exists a #SAT algorithm for formulas over $\mathcal{B}$ of size $n^{\Gamma+1-O(\varepsilon)}$ which runs in time $2^{n-n^{\varepsilon}} \cdot \mathrm{poly}(n)$.*

We remark it is known that for the $\{\mathrm{AND}, \mathrm{OR}\}$ basis $2 \leq \Gamma \leq 1/\log_2(\sqrt{5}-1) \approx 3.27$, and Paterson and Zwick [PZ93] conjectured the upper bound is tight. In addition, Chockler and Zwick [CZ01] showed that a finite basis of gates admits a non-trivial shrinkage exponent $\Gamma > 1$ if and only if it consists of unate functions. They derived explicit lower bounds on the shrinkage exponent of the basis of all unate gates of arity k (denoted U_k), namely $\Gamma \geq 1 + 1/(3k-4)$.

We first describe the randomized algorithm mentioned by [KRT13], based on the algorithm of [CKK+14]. As mentioned before, one first computes in a table A, the fraction of satisfying assignments for any formula of size at most $n^{1-\varepsilon}$. Given a larger formula of size $O(n^{3-16\varepsilon})$, the randomized algorithm works as follows: we shall apply a recursion of depth $1/\varepsilon - 7$ (assuming for convenience that $1/\varepsilon - 7$ is integral). At each step, given a formula F and a set of variables S on which the formula is supported, we apply the decomposition lemma from [IMZ12, Tal14]. Let $\ell = n^{2\varepsilon}$, the decomposition lemma yields an equivalent formula to F which is of the form $H(F_1, \ldots, F_m)$ where H is read-once, each F_i is of size $\leq \ell$, and $m\ell = O(L(F))$. Next, we pick a subset S_{FREE} of S such that each variable is selected independently with probability $p = n^{-\varepsilon}$. We go over all restrictions ρ which keep S_{FREE} alive and fix $S \setminus S_{\mathrm{FREE}}$. To simplify $F|_\rho$, we simplify

[1] Where $\mathcal{R}_p$ is the distribution of random restrictions which keeps each variable alive with probability p and fixes it to a random constant with probability $1-p$.

[2] The shrinkage exponent of a given basis $\mathcal{B}$ is the supremum over all $\alpha \in \mathbb{R}^+$ such that all formulas using gates in $\mathcal{B}$ shrink by a factor of $O(p^\alpha)$ under random restrictions sampled from $\mathcal{R}_p$.

each subformula $F_i|_\rho$, and get $F|_\rho \equiv H(F_1|_\rho, \ldots, F_m|_\rho)$. We then apply the counting algorithm recursively on $F' \equiv H(F_1|_\rho, \ldots, F_m|_\rho)$ and S_{FREE}. At depth $1/\varepsilon - 7$, we return the number of satisfying assignments either by querying the table A, in case the restricted formula is of size at most $n^{1-\varepsilon}$, or otherwise by performing an exhaustive search.

The algorithm works since at any step of the recursion $O(p^2) = O(n^{-2\varepsilon})$ shrinkage happens with probability $1 - \exp(-n^\varepsilon)$. Hence, for almost all recursion paths, we get a formula of size $O(n^{3-16\varepsilon}/n^{2-14\varepsilon}) < n^{1-\varepsilon}$. The algorithm takes $2^{n-n^\varepsilon} \cdot \text{poly}(n)$ time, since we enumerate on $n - n^{7\varepsilon}$ variables for $1 - \exp(-n^\varepsilon)$ fraction of the paths, and we do a full enumeration for the rest.

One new ingredient we introduce in this paper is an additional pre-computing step that allows us to be independent of the simplification rules. We pre-compute in a table B for any formula of size at most $n^{2\varepsilon}$, the smallest formula which is equivalent to it. This can be done in $\exp(O(n^{2\varepsilon} \log n))$ time. To simplify $F|_\rho$, it is enough to simplify each subformula $F_i|_\rho$ using the table B, and get $F|_\rho \equiv H(F_1|_\rho, \ldots, F_m|_\rho)$. The modified randomized algorithm is presented next.

Algorithm 1 RandomizedCount(F, S)

Input: a de Morgan formula F on the set of variables S.
Output: The number of satisfying assignment among all $2^{|S|}$ possible assignments.

1: **if** $L(F) \leq n^{1-\varepsilon}$ **then return** $A[F] \cdot 2^{|S|}$.
2: Decompose F into $H(F_1, \ldots, F_m)$ using the decomposition algorithm on F and $\ell = n^{2\varepsilon}$.
3: Pick S_{FREE} to be a random subset of S, where each $i \in S$ is picked ind. with probability p.
4: Let $S_{\text{FIX}} = S \setminus S_{\text{FREE}}$.
5: $res \leftarrow 0$.
6: **for all** assignments ρ to S_{FIX} **do**
7: Simplify each $F_i|_\rho$ to F'_i using table B.
8: Let $F' = H(F'_1, \ldots, F'_m)$.
9: **if** $L(F') < 8c_0 \cdot p^2 L(F)$ **then**
10: $res \leftarrow res + RandomizedCount(F', S_{\text{FREE}})$.
11: **else**
12: Count via exhaustive search on S_{FREE} the number of satisfying assignments to F' and
13: add this number to res.
14: **return** res.

One thing we swept under the rug is that Algorithm 1 actually does not perform well on all formulas, and we need to modify it a bit, by making the set S_{FIX} contain all heavy variables that appear in at least $10 \cdot \sum_{i=1}^m L(F_i)/|S|$ leaves in $F_1, \ldots, F_m$. We defer this issue to Section 4, where we present the deterministic algorithm in full detail.

In order to present a deterministic algorithm, we derandomize Step 3. First, we observe that the analysis of [KRT13] also works when S_{FREE} is sampled using a $n^{5\varepsilon}$-wise independent distribution, and such distributions may be sampled using a seed of $O(n^{5\varepsilon} \cdot \log n)$ random bits. While this implies a $\exp(O(n^{5\varepsilon} \log n))$ time procedure to iterate over all possible seeds, from which most are good (in the sense that it samples a set S_{FREE} of the right size that ensures shrinkage with high probability), one still need to be able to check if a given seed is good. To do so, we use the coloring technique of subformulas introduced by Impagliazzo et al. [IMZ12] to *analyze* shrinkage of formulas. We employ this coloring in the actual *algorithm* in order to check whether a given seed is good. For each seed, we deterministically check in time $\exp(O(n^{5\varepsilon}))$ if S_{FREE} ensures shrinkage, to get overall

a deterministic algorithm for the entire problem.

2 Preliminaries

We denote by $[n] = \{1, \ldots, n\}$. We denote by $\exp(a) = 2^a$. Recall the definition of a de Morgan formula from Section 1. A de Morgan formula is called read-once if every variable appears at most once in the tree.

Definition 2.1 (Restriction)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. A* restriction *ρ is a vector of length n of elements from $\{0, 1, \star\}$. We denote by $f|_\rho : \{0,1\}^n \to \{0,1\}$ the function f restricted according to ρ, defined by*

$$f|_\rho(x) = f(y), \quad \text{where} \quad y_i = \begin{cases} x_i, & \rho_i = \star \\ \rho_i, & \text{otherwise} \end{cases}.$$

*A p-*random restriction *is a restriction that is sampled in the following way. For every $i \in [n]$, independently with probability p set $\rho_i = \star$ and with probability $\frac{1-p}{2}$ set ρ_i to be 0 and 1, respectively. We denote this distribution of restrictions by $\mathcal{R}_p$.*

For a set of variables $S \subseteq [n]$, $\rho' \in \{0,1\}^S$ naturally describes a restriction of length n by setting $\rho_i = \rho'_i$ for $i \in S$ and $\rho_i = \star$ otherwise. We abuse notation and interchange ρ' and ρ.

Probability We state a well known variant of the Chernoff/Hoeffding inequality.

Lemma 2.2 (Chernoff-Hoeffding)**.** *Let $X = \sum_{i=1}^n X_i$ be the sum of independent random variables such that each X_i is in the range $[0, b]$, and $\mathbf{E}[X] \leq E$. Then $\mathbf{Pr}[X \geq 8E] \leq 2^{-E/b}$. In addition, $\mathbf{Pr}[X \leq 2\,\mathbf{E}[X]/3] \leq 2^{-\mathbf{E}[X]/18b}$.*

We define (k,p)-wise independent distributions.

Definition 2.3 ((k,p)-wise independent distribution)**.** *A distribution $\mathcal{D}$ over $\{0,1\}^n$ is (k,p)-*wise independent *if for all distinct $i_1, \ldots, i_k \in [n]$ and all $a_1, \ldots, a_k \in \{0,1\}$,*

$$\mathop{\mathbf{Pr}}_{X \sim \mathcal{D}}[(X_{i_1}, \ldots, X_{i_k}) = (a_1, \ldots, a_k)] = \prod_{j=1}^{k} \mathbf{Pr}[X_{i_j} = a_j] \,,$$

and for each $i \in [n]$ it holds that $\mathbf{Pr}_{X \sim \mathcal{D}}[X_i = 1] = p$.

The following is a classical construction of a (k,p)-wise independent distribution.

Theorem 2.4 ([Jof74])**.** *Let $k \leq n$ be two natural numbers, and let $p \in (0,1]$ be some power of $1/2$. Then for $s = O(k \cdot (\log n + \log 1/p))$, there exists a pseudo random generator $Gen : \{0,1\}^s \to \{0,1\}^n$, computable in $\mathrm{poly}(n \cdot \log 1/p)$ time, such that the distribution of G's output on a random input from $\{0,1\}^s$ is a (k,p)-wise independent distribution over $\{0,1\}^n$.*

Shrinkage Results

Theorem 2.5 ([Tal14]). *There exists a universal constant $c_0 \geq 1$ such that for any Boolean function f and for any $p \in (0,1]$:*

$$\mathop{\mathbf{E}}_{\rho \sim \mathcal{R}_p}[L(f|_\rho)] \leq c_0 \cdot (L(f)p^2 + 1)$$

Definition 2.6. *Let $\mathcal{B}$ be a set of Boolean gates. The* shrinkage exponent *of $\mathcal{B}$, denoted by $\Gamma_{\mathcal{B}}$ is defined to be the supremum over $c \in \mathbb{R}^+$, such that any formula F of size s using gates from $\mathcal{B}$ shrinks to a formula of expected size $O(p^c s + 1)$ under p-random restrictions.*

3 Shrinkage with High Probability

In the rest of the paper, we shall fix a set of Boolean gates, $\mathcal{B}$. For a Boolean function f, we shall denote by $L(f)$ the minimal number of gates of any formula over $\mathcal{B}$ computing f. We shall assume that the set of gates $\mathcal{B}$ contains the binary AND and OR gates. We shall further assume that formulas over $\mathcal{B}$ shrink by a factor of $O(p^\Gamma)$ under p-random restrictions, i.e.

$$\exists c_0 \forall f : \mathop{\mathbf{E}}_{\rho \sim \mathcal{R}_p} L(f|_\rho) \leq c_0 \cdot (p^\Gamma L(f) + 1). \tag{1}$$

Choosing Γ to be $\Gamma_{\mathcal{B}} - \delta$ for any constant $\delta > 0$ makes Eq. (1) valid by the definition of $\Gamma_{\mathcal{B}}$ and, in some cases, we can even take $\Gamma = \Gamma_{\mathcal{B}}$, keeping the validity of Eq. (1). More specifically,

- By Theorem 2.5, the assumption is valid for $\mathcal{B} = \{\text{AND}, \text{OR}, \text{NOT}\}$ and $\Gamma = 2$.
- For $\mathcal{B} = \{\text{AND}, \text{OR}, \text{NOT}\}$ it is conjectured that the assumption is valid for $\Gamma = 3.27$ ([PZ93]).
- For $\mathcal{B} = U_k$, the set of all unate functions on at most k inputs, the assumption is valid for $\Gamma = 1 + 1/(3k-4)$ ([CZ01]).[3]

Our results are based only on the two assumptions mentioned above, and may be instantiated with all aforementioned choices of $\mathcal{B}$.

Claim 3.1. *Let $p \in (0,1]$, and let $H, X_1, \ldots, X_t$ be disjoint sets of variables. Let $f_1, \ldots, f_t$ be Boolean functions, where each f_i can be computed by a formula of size at most $\ell := 1/p^\Gamma$ on the variables X_i and H. We say that a set S is good for $f_1, \ldots, f_t$ if*

$$\Pr_{\rho \in \{0,1\}^S}\left[\sum_{i=1}^{t} L(f_i|_\rho) \geq 16tc_0\right] < \exp(-t/\ell)\ . \tag{2}$$

Then, $\mathbf{Pr}[S$ is good for $f_1, \ldots, f_t] \geq 1 - \exp(-t/\ell)$, where S is chosen to include H and to include any $i \in \bigcup_j X_j$ with probability p, independently at random.

[3] Unlike the result of Chen [Che15], we use the shrinkage result for U_k as a black-box. Thus, our result holds also for the true value of Γ, which may possibly be larger than $1 + 1/(3k-4)$.

Proof. Let $P_S := \mathbf{Pr}_{\rho\in\{0,1\}^S}\left[\sum_{i=1}^{t} L(f_i|_\rho) \geq 8t\cdot c_0\cdot(p^\Gamma\ell+1)\right]$. Then, by splitting the expectation according to the variables in H and the variables outside H we get

$$\begin{aligned}\mathop{\mathbf{E}}_{S}[P_S] &= \mathop{\mathbf{E}}_{S}\mathop{\mathbf{Pr}}_{\rho\in\{0,1\}^S}\left[\sum_{i=1}^{t} L(f_i|_\rho) \geq 8t\cdot c_0\cdot(p^\Gamma\ell+1)\right] \\ &= \mathop{\mathbf{E}}_{\rho'\in\{0,1\}^H}\mathop{\mathbf{Pr}}_{S,\rho''\in\{0,1\}^{S\setminus H}}\left[\sum_{i=1}^{t} L(f_i|_{\rho'\circ\rho''}) \geq 8t\cdot c_0\cdot(p^\Gamma\ell+1)\right] .\end{aligned}$$

Next, note that after we fixed ρ', the functions $f_1|_{\rho'},\ldots,f_t|_{\rho'}$ are defined on independent sets of variables. Each $L(f_i|_{\rho'\circ\rho''})\leq\ell$ surely, and its expectation under S and ρ'' is at most $c_0(p^\Gamma\ell+1)$ by Eq. (1). Thus, using Chernoff/Hoeffding's bound (Lemma 2.2), for any assignment ρ' to H,

$$\mathop{\mathbf{Pr}}_{S,\rho''\in\{0,1\}^{S\setminus H}}\left[\sum_{i=1}^{t} L(f_i|_{\rho'\circ\rho''}) \geq 8t\cdot c_0\cdot(p^\Gamma\ell+1)\right] \leq \exp(-t\cdot c_0\cdot(p^\Gamma\ell+1)/\ell) \leq \exp(-2t/\ell) .$$

Thus, $\mathbf{E}_S[P_S]\leq\exp(-2t/\ell)$. By Markov's inequality $\mathbf{Pr}_S[P_S\geq\exp(-t/\ell)]\leq\exp(-t/\ell)$. Since $\mathbf{Pr}[S \text{ is good}]$ is the complement event, we complete the proof. □

4 The Algorithm

In this section, we shall refer to the following parameters. Let n be the number of input variables of the (original) formula. Let $\varepsilon>0$ be some small constant, $p=n^{-\varepsilon}$, $\ell=1/p^\Gamma$, and $k=\ell^2n^\varepsilon$. Let c_1,c_2 be large constants to be specified later.

Enumerating formulas: For the next two algorithms, we use the fact that any formula of size s on n variables may be encoded by a binary string of length $O(s\log n)$: we may encode the formula tree using a pre-order tree traversal, specifying for each node which gate or literal it is by $O(\log n)$ bits. This encoding allows reconstruction of the tree in linear time, trivially. Hence, to go over all formulas of size s, one may iterate over all binary strings of length at most $O(s\cdot\log n)$, interpret them as tree traversals, and apply the reconstruction (the reconstruction will fail on strings that do not encode formulas, but we shall still encounter all possible formulas during the enumeration).

Algorithm 2 Pre-Computing A

1: **for all** formulas F of size at most $n^{1-\varepsilon}$ **do**
2: $A[F]\leftarrow$ the fraction of satisfying assignments to F.

By the above discussion, Algorithm 2 may run in time $\exp(O(n^{1-\varepsilon}\cdot\log n))$: the enumeration goes over $\exp(O(n^{1-\varepsilon}\cdot\log n))$ many formulas, where each such formula is supported only on at most $n^{1-\varepsilon}$ variables. Hence, we can find the fraction of satisfying assignments to each such formula by a $\exp(O(n^{1-\varepsilon}))$ time exhaustive search.

Algorithm 3 Pre-Computing B

1: **for all** formulas F of size at most ℓ **do**
2: Let S be the variables appearing in F.
3: **for all** formulas F' of size at most ℓ on S, ordered by size **do**
4: **if** $F' \equiv F$ **then**
5: $B[F] \leftarrow F'$, and break (terminate inner for loop).

Similarly, Algorithm 3 may run in time $\exp(O(\ell \cdot \log n))$: the enumeration over F and F' each goes over $\exp(O(\ell \cdot \log n))$ many possibilities, and we can check the equivalence between F and F' in time $\exp(\ell) \cdot \mathrm{poly}(n)$, by examining F and F' under all assignments to S.

Algorithm 4 is described in [Tal14, Claim. 6.2], and is based on the work of [IMZ12]. It runs in polynomial time in the size of the given formula, and works over any basis of gates which contains the binary OR and AND gates.

Algorithm 4 Decompose(F,ℓ)

Input: a formula F and a natural number ℓ

Output: H,$F_1,\ldots,F_m$ such that $F \equiv H(F_1,\ldots,F_m)$, F_i is a formula of size at most ℓ, H is a read-once formula, and $m\ell = O(L(F))$.

The next algorithm derandomizes the choice of sets $(S_{\mathrm{FIX}}, S_{\mathrm{FREE}})$ in Steps 3 and 4 in Algorithm 1. We color the formulas $F_1,\ldots,F_m$ in C colors, and say that a set S_{FIX} is good with respect to some color $c \in [C]$, if the set is good as in Claim 3.1 for the formulas colored by the color c. We denote by f_i the Boolean function that the formula F_i computes.

Algorithm 5 FindGoodPartition($F_1,\ldots,F_m$, S)

Input: formulas $F_1,\ldots,F_m$, each of size at most ℓ over the set of variables S. Assuming $|S| > k\ell$.

Output: a partition of S to $S_{\mathrm{FIX}} \cup S_{\mathrm{FREE}}$ such that $|S_{\mathrm{FREE}}| \geq p|S|/2$ and $\mathbf{Pr}_{\rho\in\{0,1\}^{S_{\mathrm{FIX}}}}[\sum_{i=1}^{m} L(f_i|_\rho) \geq c_1 p^\Gamma \ell m] \leq m \cdot \exp(-k/\ell^2)$

1: Let $h = 10 \cdot \sum_{i=1}^{m} L(F_i)/|S|$.
2: Let H be the set of heavy variables that appear in at least h leaves in $F_1 \cup \ldots \cup F_m$.
3: Let $G = (V, E)$ where $V = \{1,\ldots,m\}$ and $(i,j) \in E$ iff F_i and F_j share a non-heavy variable.
4: Color G's vertices greedily, such that no edge is monochromatic, and such that each color is used at most k/ℓ times. Denote by C the number of colors and by $V = V_1 \cup \cdots \cup V_C$ the partition induced by the coloring.
5: Let $Gen : \{0,1\}^{O(k\cdot(\log n + \log 1/p))} \to \{0,1\}^n$ be the (k,p)-wise independent generator in Thm. 2.4.
6: **for all** seeds $s \in \{0,1\}^{O(k\cdot(\log n+\log 1/p))}$ **do**
7: Let $S_{\mathrm{FIX}} \leftarrow H \cup \{i \in S : Gen(s)_i = 0\}$
8: Let $S_{\mathrm{FREE}} \leftarrow S \setminus S_{\mathrm{FIX}}$.
9: **if** $|S_{\mathrm{FREE}}| > p|S|/2$ **and** S_{FIX} is good for all colors $c \in [C]$ **then**
10: **return** $(S_{\mathrm{FIX}}, S_{\mathrm{FREE}})$.

We remark that Steps 3 and 4 in Algorithm 5 are borrowed from [IMZ12], and were used in the *analysis* of [KRT13] to prove that de Morgan formulas shrink with high probability. We implement

this coloring in the *algorithm itself*, in order to deterministically check that the partition we got is good.

Algorithm 6 Count(F, S)

Input: a formula F on the set of variables S.

Output: The number of satisfying assignments for F among all $2^{|S|}$ possible assignments.

1: **if** $L(F) \leq n^{1-\varepsilon}$ **then return** $A[F] \cdot 2^{|S|}$.
2: **if** $|S| \leq k\ell$ **then return** the number of satisfying assignments by an exhaustive search over all $2^{|S|}$ possible assignments.
3: Decompose F into $H(F_1, \ldots, F_m)$ using algorithm 4 on F and ℓ.
4: $(S_{\text{FIX}}, S_{\text{FREE}}) \leftarrow$ FindGoodPartition$(F_1, \ldots, F_m, S)$.
5: $res \leftarrow 0$.
6: **for all** assignments ρ to S_{FIX} **do**
7: Simplify all $F_i|_\rho$ to F'_i using table B.
8: Let $F' = H(F'_1, \ldots, F'_m)$.
9: **if** $L(F') < c_2 \cdot p^\Gamma L(F)$ **then**
10: $res \leftarrow res + Count(F', S_{\text{FREE}})$.
11: **else**
12: Count via an exhaustive search the number of satisfying assignments to F' and
13: add this number to res.
14: **return** res.

Lemma 4.1 (Algorithm 5 works)**.** *Assuming $|S| > k\ell$, Algorithm 5 finds in $\exp(O(k \cdot \log n))$ time, a partition $S = S_{\text{FIX}} \cup S_{\text{FREE}}$ such that: (1) $|S_{\text{FREE}}| > p|S|/2$ (2) $\mathbf{Pr}_{\rho \in \{0,1\}^{S_{\text{FIX}}}}[\sum_{i=1}^m L(f_i|_\rho) \geq c_1 p^\Gamma \ell m] \leq m \cdot \exp(-k/\ell^2)$.*

Proof. We begin by estimating the number of colors in Step 4. Denote by $L = \ell \cdot m$. The maximal degree in the graph G is at most ℓh, since each formula has at most ℓ non-heavy variables, and each such variable appears in at most h other formulas. Say a color is *active*, if at least one vertex was colored by it and fewer than k/ℓ vertices were colored by it. Observe that when we greedily color G's vertices one by one, at each step, at most $\ell h + 1$ colors are active. Hence, the number of colors C at the end of the coloring process would be at most

$$C \leq \frac{m}{k/\ell} + \ell h + 1 \leq \frac{L}{k} + \frac{10L\ell}{|S|} + 1 = O\left(\frac{L}{k}\right),$$

where we used $|S| > k\ell$ for the last inequality.

Next, we show that there exists a seed for which the condition in Step 9 is met. We do so by proving that both the first and second part of the condition hold individually with probability at least $1 - o(1)$ over a random seed to *Gen*:

- We show that $\mathbf{Pr}[|S_{\text{FREE}}| > p|S|/2] > 1 - o(1)$. By definition of H, its size is at most $|S|/10$. The set S_{FREE} is picked from $S \setminus H$ using a (k, p)-wise distribution. Partition the variables in $S \setminus H$ into buckets of $k/2$ variables each, except for the last bucket which will get in addition the remaining $|S| \mod (k/2)$ variables. Thus, each bucket is of size between $k/2$ and k. For a bucket with k' variables, by Chernoff/Hoeffding's bound and the fact that the distribution of

Gen's output is (k,p)-wise independent, the probability to select less than $k'p \cdot 2/3$ variables to S_{FREE} is at most $\exp(-\Omega(k'p)) = \exp(-\Omega(kp))$. By union bound, the probability that there exists a bucket with less than $k'p \cdot 2/3$ variables is at most $\text{poly}(n) \cdot \exp(-\Omega(kp))$. Since $kp > n^{\varepsilon}$ we get that this probability is $o(1)$. Hence, with probability $1 - o(1)$, the set S_{FREE} is of size at least $p|S \setminus H| \cdot \frac{2}{3} \geq p|S| \cdot \frac{9}{10} \cdot \frac{2}{3} > p|S|/2$.

- We show that $\mathbf{Pr}[S_{\text{FIX}}$ is good for all colors $c \in [C]] \geq 1 - o(1)$. Let $c \in [C]$ be some color, and let $F_{i_1}, \ldots, F_{i_t}$ be the formulas with color c, where $t \leq k/\ell$. Without loss of generality, $t = k/\ell$, since otherwise we can add dummy formulas of size 0. Note that the since S_{FIX} contains H and is chosen using a (k,p)-wise independent distribution, its marginal distribution to the variables in $F_{i_1}, \ldots, F_{i_t}$ is identical to the distribution described in Claim 3.1. Hence, Claim 3.1 yields $\mathbf{Pr}[S_{\text{FIX}}$ is good for $f_{i_1}, \ldots, f_{i_t}] \geq 1 - \exp(-k/\ell^2)$. Using a union bound with probability at least $1 - C \cdot \exp(-k/\ell^2)$, we have that S_{FIX} is good for all colors.

Next, we show that (2) holds once condition 9 is met. Since S_{FIX} is good for all colors

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{\rho \in \{0,1\}^{S_{\text{FIX}}}} \left[\exists c \in [C] : \sum_{i \in V_c} L(f_i|_\rho) \geq 16tc_0 \right] \\
&\leq \sum_{c \in [C]} \mathop{\mathbf{Pr}}_{\rho \in \{0,1\}^{S_{\text{FIX}}}} \left[\sum_{i \in V_c} L(f_i|_\rho) \geq 16tc_0 \right] && \text{(union bound)} \\
&< C \cdot \exp(-t/\ell) && \text{(Definition of good, Eq. (2))} \\
&\leq m \cdot \exp(-k/\ell^2) \, . && (t = k/\ell,\ C \leq m)
\end{aligned}$$

In the complement event, which happens with probability at least $1 - m \cdot \exp(-k/\ell^2)$, for all $c \in [C]$ it holds that $\sum_{i \in V_c} L(f_i|_\rho) < 16tc_0 = 16(k/\ell)c_0$. Hence, using $C = O(L/k)$, we get

$$\sum_{c \in [C]} \sum_{i \in V_c} L(f_i|_\rho) < C \cdot 16c_0(k/\ell) = O(L/\ell) = O(Lp^{\Gamma}).$$

We turn to analyzing the time complexity of Algorithm 5. Steps 1-5 are done in $\text{poly}(n)$ time. The for loop in step 6 goes over $\exp(O(k \cdot \log n))$ seeds. To check if S_{FIX} is good for a specific color $c \in [C]$, one needs to calculate $\mathbf{Pr}_{\rho\ \{0,1\}^{S_{\text{FIX}}}}[\sum_{i \in V_c} L(f_i|_\rho) \geq 16tc_0]$. This can be done by enumerating over all possible assignments to the variables in S_{FIX} which appear in the formulas $\{F_i : i \in V_c\}$. and checking under each assignment whether $\sum_{i \in V_c} L(f_i|_\rho) \geq 16tc_0$. By our design, the number of variables we enumerate on is at most $\sum_{i \in V_c} |F_i| \leq \ell \cdot |V_c| \leq \ell \cdot (k/\ell) = k$, hence the enumeration takes $2^k \cdot \text{poly}(n)$ time. □

Theorem 4.2. *Algorithm 6 counts the number of satisfying assignments of a given formula F in time $2^{n-n^{\varepsilon}} \cdot \text{poly}(n)$ assuming $L(F) \leq n^{\Gamma+1-(3\Gamma^2+\Gamma+2)\varepsilon}$, for a large enough n.*

Proof. We start by proving the correctness of Algorithm 6. First, if the condition in step 1 is met, then by the correctness of the table A, we return the number of satisfying assignments for F among all $2^{|S|}$ possible assignments. Second, if the condition in step 2 is met, then we also return the correct answer trivially. Otherwise, we go over all possible assignments for the variables in S_{FIX}, and under each assignment ρ, we construct a formula F' which is equivalent to $F|_\rho$ by the correctness of the table B and the fact that each F_i is of size at most ℓ. Assuming n is large enough,

if we reach step 10, then F' is of size smaller than the size of F, and we may apply induction and assume that $Count(F', S_{\text{FREE}})$ returns the true number of satisfying assignments for $F' \equiv F|_\rho$. If we reached step 12, then trivially we count the number of satisfying assignments for $F' \equiv F|_\rho$. Thus, the returned value res is correct.

In the remainder of the proof, we upper bound the time complexity. Assume that the algorithm is invoked with a formula of size at most $n^{1+\Gamma-K\varepsilon}$, for some constant K to be determined later. Then, at recursion level d, if reached, we have a formula F supported on S where S is of size at least $(p/2)^d \cdot n$ (using $|S_{\text{FREE}}| > p|S|/2$ in Lemma 4.1) and F is of size at most $c_2^d p^{\Gamma d} n^{1+\Gamma-K\varepsilon}$. So, at depth $d = 1/\varepsilon - 3\Gamma - 1$,[4] (or before) condition 1 is met and we terminate, since the size of the formula is at most $c_2^d \frac{n^{1+\Gamma-K\varepsilon}}{n^{\Gamma\varepsilon(1/\varepsilon-3\Gamma-1)}} = O(n^{1-\varepsilon(K-3\Gamma^2-\Gamma)}) < n^{1-\varepsilon}$ for a large enough n, and $K = 3\Gamma^2 + \Gamma + 2$. However, the number of variables left alive is at least $(p/2)^d \cdot n \geq \Omega(n^{\varepsilon(3\Gamma+1)}) > n^\varepsilon$ for a large enough n. Thus, the algorithm runs in time $\text{poly}(n) \leq 2^{|S|-n^\varepsilon} \cdot \text{poly}(n)$ in the d'th level of recursion.

We claim by induction on $j = 0, \ldots, d$ that any invocation of the algorithm at depth $d-j$ takes at most $\text{poly}(n) \cdot 2^{|S|-n^\varepsilon}$ time. Based on the above discussion, the claim is true for $j = 0$. For $j \geq 1$, note that $|S| \geq (p/2)^{d-1} \cdot n \geq \Omega(n^{\varepsilon(3\Gamma+2)}) > k\ell$, hence condition 2 is not met. By Lemma 4.1, we get in Step 4 a set S_{FIX}, such that

$$\mathop{\mathbf{Pr}}_{\rho\in\{0,1\}^{S_{\text{FIX}}}}\left[\sum_{i=1}^{m} L(f_i|_\rho) \geq c_1 p^\Gamma \ell m\right] \leq m \cdot \exp(-k/\ell^2) \; .$$

By the guarantee of Algorithm 4 that $\ell m = O(L(F))$, there exists a universal constant c_2 such that

$$\mathop{\mathbf{Pr}}_{\rho\in\{0,1\}^{S_{\text{FIX}}}}\left[\sum_{i=1}^{m} L(f_i|_\rho) \geq c_2 p^\Gamma L(F)\right] \leq m \cdot \exp(-k/\ell^2) \; . \tag{3}$$

We calculate how many steps are spent on bad assignments for S_{FIX} (on which we reach step 12), and how many are spent on good assignments (on which we reach step 10). By Eq. (3), there are at most $m \cdot 2^{|S_{\text{FIX}}|-k/\ell^2}$ bad choices for S_{FIX}, each taking $2^{|S_{\text{FREE}}|} \cdot \text{poly}(n)$ time. On the other hand, there are at most $2^{|S_{\text{FIX}}|}$ good choices for S_{FIX}, each taking $2^{|S_{\text{FREE}}|-n^\varepsilon} \cdot \text{poly}(n)$ time by the induction hypothesis. Overall, the entire algorithm takes

$$\left(2^{O(k\cdot\log n)} + 2^{|S_{\text{FIX}}|+|S_{\text{FREE}}|-k/\ell^2} + 2^{|S_{\text{FIX}}|+|S_{\text{FREE}}|-n^\varepsilon}\right) \cdot \text{poly}(n) \leq \left(3 \cdot 2^{|S|-n^\varepsilon}\right) \cdot \text{poly}(n)$$

time, where we used the fact that $k = \ell^2 n^\varepsilon$ and that $|S| - n^\varepsilon \gg k \cdot \log n$. □

Acknowledgement I wish to thank Ilan Komargodski and Ran Raz for their encouragement. I thank the anonymous referees for their helpful comments on an earlier version of this paper.

References

[And87] A. E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42:63–66, 1987. In Russian.

[Che15] R. Chen. Satisfiability algorithms and lower bounds for boolean formulas over finite bases. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:99, 2015.

[4] Assume for convenience that $1/\varepsilon - 3\Gamma - 1$ is an integer

[CKK+14] R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel, and D. Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *CCC*, pages 262–273, 2014.

[CKS14] R. Chen, V. Kabanets, and N. Saurabh. An improved deterministic #SAT algorithm for small De Morgan formulas. In *MFCS*, pages 165–176, 2014.

[CZ01] H. Chockler and U. Zwick. Which bases admit non-trivial shrinkage of formulae? *Computational Complexity*, 10(1):28–40, 2001.

[Hås98] J. Håstad. The shrinkage exponent of De Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[IK14] R. Impagliazzo and V. Kabanets. Fourier concentration from shrinkage. In *CCC*, pages 321–332, 2014.

[IMZ12] R. Impagliazzo, R. Meka, and D. Zuckerman. Pseudorandomness from shrinkage. In *FOCS*, pages 111–119, 2012.

[IN93] R. Impagliazzo and N. Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993.

[Jof74] A. Joffe. On a set of almost deterministic k-independent random variables. *Ann. Probab.*, 2(1):161–162, 02 1974.

[KR13] I. Komargodski and R. Raz. Average-case lower bounds for formula size. In *STOC*, pages 171–180, 2013.

[KRT13] I. Komargodski, R. Raz, and A. Tal. Improved average-case lower bounds for De Morgan formula size. In *FOCS*, pages 588–597, 2013.

[PZ93] M. Paterson and U. Zwick. Shrinkage of De Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993.

[San10] R. Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *FOCS*, pages 183–192, 2010.

[ST13] K. Seto and S. Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013.

[Sub61] B. A. Subbotovskaya. Realizations of linear function by formulas using $+, \cdot, -$. *Doklady Akademii Nauk SSSR*, 136:3:553–555, 1961. In Russian.

[Tal14] A. Tal. Shrinkage of de Morgan formulae from quantum query complexity. In *FOCS*, pages 551–560, 2014.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de