Electronic Colloquium on Computational Complexity, Report No. 192 (2015)

Satisfiability on Mixed Instances

Ruiwen Chen* Rahul Santhanam†

November 26, 2015

Abstract

The study of the worst-case complexity of the Boolean Satisfiability (SAT) problem has seen considerable progress in recent years, for various types of instances including CNFs [16, 15, 20, 21], Boolean formulas [18] and constant-depth circuits [6]. We systematically investigate the complexity of solving *mixed* instances, where different parts of the instance come from different types. Our investigation is motivated partly by practical contexts such as SMT (Satisfiability Modulo Theories) solving, and partly by theoretical issues such as the exact complexity of graph problems and the desire to find a unifying framework for known satisfiability algorithms.

We investigate two kinds of mixing: conjunctive mixing, where the mixed instance is formed by taking the conjunction of pure instances of different types, and compositional mixing, where the mixed instance is formed by the composition of different kinds of circuits. For conjunctive mixing, we show that non-trivial savings over brute force search can be obtained for a number of instance types in a generic way using the paradigm of *subcube partitioning*. We apply this generic result to show a meta-algorithmic result about graph optimisation problems: any optimisation problem that can be formalised in Monadic SNP can be solved exactly with exponential savings over brute-force search. This captures known results about problems such as Clique, Independent Set and Vertex Cover, in a uniform way. For certain kinds of conjunctive mixing, such as mixtures of k-CNFs and CNFs of bounded size, and of k-CNFs and Boolean formulas, we obtain improved savings over subcube partitioning by combining existing algorithmic ideas in a more fine-grained way.

We use the perspective of compositional mixing to show the first non-trivial algorithm for satisfiability of quantified Boolean formulas, where there is no depth restriction on the formula. We show that there is an algorithm which for any such formula with a constant number of quantifier blocks and of size n^c, where $c < 5/4$, solves satisfiability in time $2^{n-n^{\Omega(1)}}$.

1 Introduction

Boolean Satisfiability (SAT) is the canonical NP-complete problem. Much effort has gone into designing and analyzing exact algorithms for SAT. Unless NP = P, we cannot hope to find a polynomial-time algorithm for SAT. So we adopt a milder goal: finding algorithms that beat the trivial brute-force search algorithm, which runs in time $2^n \operatorname{poly}(m)$ on Boolean circuits of size m with n variables. There has been significant progress on this over the past couple of decades, motivated by the theoretical significance of the problem and the practical success of SAT solvers [11] in domains such as verification and automated planning.

*School of Informatics, University of Edinburgh, Edinburgh, United Kingdom; rchen2@inf.ed.ac.uk
†School of Informatics, University of Edinburgh, Edinburgh, United Kingdom; rsanthan@inf.ed.ac.uk

ISSN 1433-8092

However, the performance of satisfiability algorithms depends critically on the *type* of the instances. For k-SAT, which is the satisfiability problem on k-CNFs, algorithms running in time $2^{n-n/k}$ are known [16, 20, 15]. Contrastingly, for satisfiability on general Boolean circuits, no upper bound better than the trivial one is known. In general, the more expressive the class of instances on which we are trying to solve SAT, the less we know about how to improve on brute-force search. A partial explanation for this phenomenon is provided by the recent work of Williams [22, 23], which connects progress on SAT algorithms to breakthroughs in circuit lower bounds.

Traditionally, satisfiability is studied for instances of a single fixed type, e.g., k-CNFs, CNFs with a prescribed number of clauses, Boolean formulas of a prescribed size, constant-depth Boolean circuits of a prescribed depth and size etc. We call such instances *pure* instances of the given type. In this paper, we systematically investigate the worst-case complexity of SAT, when the instance is *mixed*, i.e., has different parts of different types. A simple kind of mixing is *conjunctive mixing*, where the instance is formed by the conjunction of pure instances of different types. A more complex kind of mixing which we also study is *compositional mixing*, where the mixed instance is formed by composing circuits corresponding to different types.

Our study of satsifiability of mixed instances is motivated by various considerations. From a practical point of view, there has been a great deal of work recently on SMT (Satisfiability Modulo Theories) solvers [13], which extend SAT solvers by being able to deal with instances which contain not just propositional connectives, but various arithmetic operations, inequalities, etc. Studying the exact complexity of mixed instances is a way of connecting with this work from the theoretical side.

From a theoretical point of view, though we do have many interesting non-trivial satisfiability algorithms now, we do not clearly understand what unifies these algorithms, and what their limits are. Mixed instances provide a test for the flexibility of these algorithms, and analyzing mixed instances gives us a deeper understanding of existing algorithmic ideas.

More compellingly, mixed instances of satisfiability can be used to capture, in a fairly direct way, various NP-hard optimisation problems such as Clique, Vertex Cover, Dominating Set, etc. Upper and lower bounds on the exact complexity of satisfiability for mixed instances translate to these other problems. As a further example, the Max-SAT problem, where we ask for an assignment maximizing the number of satisfied clauses of a CNF formula, can be modelled easily by compositional mixing, where a CNF formula is composed with a threshold gate. The critical thing about these correspondences between different problems is that they preserve the number of variables of the instance, and thus results on exact complexity are easily transferrable from one problem to the other.

Mixed instances also come up fairly naturally in the analysis of important algorithms, such as the Sparsification Lemma of Impagliazzo, Paturi and Zane [7]. More specifically, the question of how best to solve a conjunctive mixture of CNFs with different widths and sizes is still unresolved, and might well play an important role in settling open questions such as deterministic counting of satisfying assignments for k-CNFs.

To state our results, we need some notation. Given classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of instances, we let $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT denote the satisfiability problem for conjunctive mixing of $\mathcal{C}_1$ and $\mathcal{C}_2$. Also, given a function $f : \mathbb{N} \to \mathbb{N}$ and a class $\mathcal{C}$ of instances, we say $\mathcal{C}$-SAT has savings f if it can be solved in time $2^{n-f(n)}\,\mathrm{poly}(m)$ time, where n is the number of variables of the instance, and m its size. We say the savings f is non-trivial if $f = \omega(\log(n))$.

If $\mathcal{C}_1$-SAT has savings f_1 and $\mathcal{C}_2$-SAT has savings f_2, the best savings we can expect for $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-

SAT is $\min\{f_1, f_2\}$, without improving on the known algorithms for $\mathcal{C}_1$-SAT or $\mathcal{C}_2$-SAT. An initial question is whether $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT can be shown to have non-trivial savings, given bounds on the savings for $\mathcal{C}_1$-SAT and $\mathcal{C}_2$-SAT.

We give a positive answer to this question for a large number of types of interest, including k-CNFs, Boolean formulas and constant-depth circuits. We critically use the fact that most known algorithms for these problems can be captured by a paradigm called *subcube partitioning*. Algorithms based on subcube partitioning "compose" well with each other, and therefore work well on mixed instances.

As an application of this result, we show a *meta-algorithmic* result for exact complexity of certain NP-hard optimisation problems, which are "expressible" by Monadic SNP formulas. Such problems include Weighted Independent Set, Weighted Clique and Weighted Vertex Cover.

Theorem 1.1. *Let r be a constant. Any Monadic SNP-expressible weighted optimisation problem on graphs or r-uniform hypergraphs has savings $\Omega(n)$.*

We then show how to get improved savings beyond the bound given by subcube paritioning for mixtures such as conjunctive mixtures of k-CNFS and CNFs with m clauses, for which we get an optimal result by exploiting more carefully the properties of the known algorithms.

In the final section, we move on to compositional mixing. We show how to interpret the satisfiability algorithm of Impagliazzo, Mathews, and Paturi [6] for constant-depth circuits as an algorithm for compositionally mixed instances. Then, we use the perspective of compositional mixing of constant-depth circuits and de Morgan formulas, and show the first non-trivial algorithm for satisfiability of quantified de Morgan formulas, where the formula has unbounded depth. Santhanam and Williams [19] recently showed non-trivial results for the case where the formula is a CNF, but nothing was known about the unbounded-depth case. Our algorithm and its analysis combine various ideas from recent work on satisfiability and lower bounds for Boolean formulas with the approach in [19].

Theorem 1.2. *For quantified de Morgan formulas with n variables, q quantifier blocks and size at most $n^{5/4-\epsilon}$, where $\epsilon > 0$, there is a zero-error randomized satisfiability algorithm running in time $2^{n-n^{\Omega(\epsilon/(q+1))}}$.*

Our work is not the first to study satisfiability on mixed instances. The literature on satisfiability of random formulas has results [12] on threshold phenomena concerning instances where some of the clauses are 2-clauses and the others are 3-clauses. Patrascu and Williams [14] study the satisfiability problem on mixtures of 2-SAT formulas with two clauses of arbitrary length, and show that under the Strong Exponential Time Hypothesis, such mixed instances cannot be solved in time $O(m^{2-\epsilon})$ for any $\epsilon > 0$, where m is the size of the instance. Porschen and Speckenmeyer [17] study mixtures of Horn clauses and 2-clauses, and show various positive and negative results.

However, as far as we are aware, we are the first to study satisfiability on mixed instances in a systematic way, for various types of instances where polynomial-time algorithms are not known, and indeed do not exist unless NP = P. We believe that our perspective might be useful in unifying results on exact algorithms for various NP-hard problems, as well as in gaining a deeper understanding of known satisfiability algorithms. In particular, the notion of compositional mixing is novel, to the best of our knowledge.

2 Preliminaries

2.1 Circuit Complexity and Satisfiability

Given a circuit class $\mathcal{C}$, $\mathcal{C}$-SAT denotes the satisfiability problem for circuits from $\mathcal{C}$, and $(\wedge\mathcal{C})$-SAT denotes the satisfiability problem for a conjunction of circuits from $\mathcal{C}$. Given circuit classes $\mathcal{C}_1$ and $\mathcal{C}_2$, $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT denotes the satisfiability problem for circuits C where C is the conjunction of C' and C'', for $C' \in \mathcal{C}_1$ and $C'' \in \mathcal{C}_2$. Given circuit classes $\mathcal{C}_1$ and $\mathcal{C}_2$, $(\mathcal{C}_1 \circ \mathcal{C}_2)$-SAT denotes the satisfiability problem for circuits from $\mathcal{C}_1$ each of whose inputs is the output of a circuit from $\mathcal{C}_2$.

There are some standard circuit classes we will use repeatedly. Given a positive integer k, $k\mathsf{CNF}$ is the class of CNFs of width k. Given a function $m : \mathbb{N} \to \mathbb{N}$, $\mathsf{CNF}[m]$ is the class of CNFs which have n variables and at most $m(n)$ clauses for some n. CNF is the class of CNFs which have n variables, with no restriction on the number of clauses. Similarly, $\mathsf{Formula}[m]$ is the class of Boolean formulas (with no depth restriction) which have n variables and at most $m(n)$ literals for some n, with $\mathsf{Formula}$ defined as the class of formulas with no restriction on the number of literals. Given a positive integer d and m as before, $\mathsf{AC}^0_d[m]$ is the class of unbounded fan-in Boolean circuits with AND, OR and NOT gates which have n variables, size at most $m(n)$ and depth at most d, for some n. AC^0_d is defined as the class of depth-d circuits with AND, OR and NOT gates, with no restriction on the size. THR is the class of threshold functions, i.e., functions of the form $\Sigma_i a_i x_i \geqslant b$, where x_i's are input variables, and a_i's and b are arbitrary integers. Given a finite field R, LIN_R is the class of systems of linear equations over R.

We review some known results about satisfiability algorithms. Several of these algorithms exploit a structural property of certain circuit classes, namely the existence of better-than-trivial subcube partitions. We first define this notion.

A *restriction* is a string over the alphabet $\{0, 1, *\}$. A *subcube partition* of size s over $\{0,1\}^n$ is a family of s restrictions in $\{0, 1, *\}^n$ such that for each string in $\{0,1\}^n$, there is precisely one restriction in the family which agrees with the string on all non-$*$ co-ordinates. A circuit C on n variables is said to admit a subcube partition of size s if there is a subcube partition of size s over $\{0,1\}^n$ such that for every restriction in the partition, C is a constant under that restriction. If C admits a subcube partition of size s, a *subcube partition labelling* for C is a list of pairs $(\rho_i, b_i), i = 1, \ldots, s$ such that $\{\rho_i\}_{i=1}^s$ form a subcube partition, and for each ρ_i, C restricted to ρ_i, denoted by $C|_{\rho_i}$, has value b_i.

Given a circuit class $\mathcal{C}$ and a function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $f(n, \cdot) \leqslant n$ for all $n \in \mathbb{N}$, $\mathcal{C}$-SAT is said to have *savings* f is there is an algorithm for $\mathcal{C}$-SAT running in time $2^{n-f(n,m)}\,\mathrm{poly}(m)$ time, where n is the number of variables of the instance from $\mathcal{C}$ and m its size. On occasion, when the parameter m does not appear in the analyzed savings, or is implicit, we model f as a function purely of the number variables. By default, we assume our algorithms to be zero-error randomized algorithms. $\mathcal{C}$-SAT is said to have a *subcube partitioning* algorithm with savings f if there is an algorithm, which for C from $\mathcal{C}$ of size m on n variables, outputs in time $2^{n-f(n,m)}\,\mathrm{poly}(m)$ a subcube partition labelling for C. Note that the existence of such an algorithm implies in particular that any circuit from $\mathcal{C}$ of size m on n variables admits a subcube partition of size $2^{n-\Omega(f(n,m))}\,\mathrm{poly}(m)$.

The following is a folklore result, which can be shown by analyzing a natural randomized branching algorithm using Hastad's Switching Lemma [5, 1].

Theorem 2.1. *For any fixed k, $k\mathsf{CNF}$-SAT has a zero-error randomized subcube partitioning algorithm with savings $\Omega(n/k)$.*

Theorem 2.2 ([21, 6]). *For a fixed d, $\mathsf{AC}^0_d[m]$-SAT has a zero-error randomized subcube partitioning algorithm with savings $\Omega(n/\log(m/n)^{d-1})$. In particular, the algorithm for $\mathsf{CNF}[m]$-SAT has savings $\Omega(n/\log(m/n))$.*

Theorem 2.3 ([18]). *$\mathsf{Formula}[m]$-SAT has a deterministic subcube partitioning algorithm with savings $\Omega(n^3/m^2)$. In particular, for $m = O(n)$, the algorithm has savings $\Omega(n)$.*

2.2 Logical Complexity

We first recall first-order logic on graphs. A graph is represented by the binary edge relation $E(x, y)$, whose arguments range over vertices of the graph. Assume an infinite supply of individual variables ranging over vertices of the graph, denoted by (possibly subscripted) lowercase letters $x, y, z, \ldots$. Formulas of first-order logic over graphs are constructed from atomic formulas $E(x, y)$ and $x = y$ using the propositional connectives $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation), as well as existential quantification $\exists$ and universal quantification $\forall$ over individual variables.

Now also assume an infinite supply of set variables ranging over subsets of vertices of the graph, denoted by (possibly subscripted) uppercase letters $X, Y, Z, \ldots$. Formulas of monadic second-order (MSO) logic over graphs are constructed from atomic formulas $E(x, y)$, $x = y$ and $X(x)$ (expressing that vertex x belongs to the set X) using the propositional connectives, universal and existential quantification over individual variables and existential quantification over the set variables. We assume wlog that MSO formulas are in the prenex normal form, with the quantifiers over set variables appearing first. Formulas of monadic second-order logic over graphs (MSNP) are MSO formulas whose first-order part contains only universal quantification over variables.

We will be interested in expressing weighted graph optimisation problems such as Independent Set, Vertex Cover and Dominating Set using these logical formalisms. A weighted graph $G = (V, E, w)$ is a graph (V, E) with a weight function $w : V \to \mathbb{R}^+$. The weight function naturally extends to subsets $S \subset V$ by $w(S) = \sum_{v \in S} w(v)$. Given a MSO formula $\phi = \exists X_1 X_2 \ldots X_k \psi(X_1, X_2 \ldots X_k)$, where ψ is first-order, the max-weighted (resp. min-weighted) optimisation problem $O^{\max}_\phi$ (resp. $O^{\min}_\phi$) corresponding to ϕ takes as input a weighted graph $G = (V, E, w)$ (where the weights are representable in poly$(|V|)$ bits) and outputs the maximum (resp. minimum) of $\sum_{i=1}^k w(X_i)$ over $X_1 \ldots X_k \subset V$ satisfying $\psi(X_1, X_2 \ldots X_k)$ in G. A weighted graph optimisation problem O is said to be MSO-representable (resp. MSNP-representable) if there is a MSO formula (resp. a MSNP formula) ϕ such that O is either $O^{\max}_\phi$ or $O^{\min}_\phi$.

We are interested in solving MSO-representable and MSNP-representable optimisation problems better than exhaustive search. Note that given an MSO formula ϕ quantifying existentially over q subset variables, the optimisation problems $O^{\max}_\phi$ and $O^{\min}_\phi$ can be solved using exhaustive search in time $2^{qn}\,\mathrm{poly}(n)$, where n is the number of vertices of the input graph. Given a function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) \leqslant n$ for all n and a formula ϕ as above, we say that $O^{\max}_\phi$ (resp. $O^{\min}_\phi$) has savings $f(n)$ if there is an algorithm for the problem running in time $2^{qn-f(n)}\,\mathrm{poly}(n)$.

A number of natural graph optimisation problems can be captured by the formalism above. It is easy to see that Maximum Weight Independent Set, Maximum Weight Clique, Minimum Weight Vertex Cover and Maximum Weight Dominating Set are all MSO-representable. The first three of these problems are also MSNP-representable. In each of these cases, the corresponding MSO formula existentially quantifies over a single subset of vertices.

The framework described above can be extended easily to optimisation problems on r-uniform hypergraphs, with the change that the edge relation E is r-ary rather than binary.

2.3 Random Restrictions

Let $\mathcal{R}_p$ be the random restriction where a subset U of pn variables is chosen uniformly at random, and each variable not in U is assigned 0 or 1 each with probability 1/2. We give below the shrinkage of formulas under random restrictions; although the parameters here are slightly weaker than those obtained via greedy restrictions (as used in Theorem 2.3 [18] and also in [10]), we can use the results to get satisfiability algorithms for mixed and composed instances. For completeness, we provide proofs in the Appendix.

Lemma 2.4. *Let F be a de Morgan formula of size cn where each variable appears at most $O(c)$ times. Then, for $p \leqslant 1/(20c)^2$,*

$$\begin{aligned} & \mathbf{Pr}_{\rho\sim\mathcal{R}_p}[\ F|_\rho \text{ depends on } \geqslant \tfrac{3}{5}pn \text{ variables }] \\ < \ & 2^{-\Omega(\min\{n/c^4,\ pn\})}. \end{aligned}$$

Lemma 2.5. *Let F be a de Morgan formula of size $L \leqslant n^{5/4-\epsilon}$ for $\epsilon > 0$, where each variable appears at most $O(L/n)$ times. Then, for $p = n^{\epsilon/2}/n$ and any constant $\epsilon' < \epsilon/2$,*

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[\ F|_\rho \text{ depends on } \geqslant n^{\epsilon'} \text{ variables }] < 2^{-\Omega(n^{\epsilon'})}.$$

Let ϕ be a CNF. We assume clauses and literals in ϕ are in a canonical order. The *canonical decision tree* for ϕ is constructed as follows: If there is no clause left, return 1. If any clause is empty, return 0. Otherwise, query all variables in the first clause; when a literal in a clause is fixed to 0, remove the literal from the clause, and when a literal in a clause is fixed to 1, remove the whole clause; then recurse. We denote by $D(\phi)$ the depth of the canonical decision tree for ϕ. Canonical decision trees for DNFs can be defined analogously.

Lemma 2.6 (Hastad's switching lemma [5, 1])**.** *Let ϕ be a k-CNF or k-DNF on n variables. Then for any $s \geqslant 0$ and $p \leqslant 1/7$,*

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[D(\phi|_\rho) \geqslant s] < (7pk)^s.$$

3 Exploiting Subcube Partitioning for Mixed Instances

We are interested in designing and analyzing algorithms for $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT which have performance comparable to the best known algorithms for $\mathcal{C}_1$-SAT and $\mathcal{C}_2$-SAT.

First, we show a negative result. There are circuit classes $\mathcal{C}_1$ and $\mathcal{C}_2$ for which the satisfiability problem is in polynomial time, but satisfiability of mixed instances requires exponential time to solve under standard complexity-theoretic hypotheses.

Theorem 3.1. *($2\mathsf{CNF}\wedge\mathsf{THR}$)-SAT requires time $2^{\Omega(n)}$, assuming the Exponential Time Hypothesis.*

Proof. Let $\mathcal{C}_1$ be 2-CNF and $\mathcal{C}_2$ be THR. Note that $\mathcal{C}_1$-SAT and $\mathcal{C}_2$-SAT are both polynomial-time solvable. We show that $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT is hard under the Exponential Time Hypothesis, by encoding Independent Set into this problem.

Let (G, k) be an instance of the Independent Set problem, where the question is whether G has an independent set of size at least k. Let n be the number of vertices of G. We reduce such an instance to an instance of $(2\mathsf{CNF} \wedge \mathsf{THR})$-SAT preserving the number of variables as follows. Let

$x_1 \dots x_n$ be propositional variables. The intended interpretation of the variables is that the set of all variables assigned to true should form an independent set in G. To enforce this interpretation, we define clauses as follows: for each edge (i, j) of G, where $i, j \in [n]$, we add a clause $(\neg x_i \vee \neg x_j)$. Note that the collection of clauses of this form is a 2-CNF. We also add a single threshold gate which checks if $\Sigma_i x_i \geqslant k$. Clearly, the resulting instance ϕ is satisfiable iff (G, k) is a YES instance of Independent Set. It is known [8] that Independent Set requires time $2^{\Omega(n)}$ if the Exponential Time Hypothesis holds, hence the same is true for $(\mathsf{2CNF} \wedge \mathsf{THR})$-SAT. □

Next, we show a positive result when $\mathcal{C}_1$-SAT has a subcube partitioning algorithm with non-trivial savings, and $\mathcal{C}_2$-SAT is polynomial-time solvable. We do assume that $\mathcal{C}_2$ satisfies a certain natural condition. We say that a circuit class $\mathcal{C}$ is closed under restrictions if for any circuit C belonging to the class and for any partial restriction of the variables of C, the resulting circuit C' belongs to the class as well. All commonly studied circuit classes satisfy this condition.

Theorem 3.2. *Let $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a function such that $f(n, \cdot) \leqslant n$ for all $n \in \mathbb{N}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be circuit classes such that $\mathcal{C}_1$-SAT has a subcube partitioning algorithm with savings $f(\cdot, \cdot)$ and $\mathcal{C}_2$-SAT has a polynomial-time algorithm. Moreover, assume $\mathcal{C}_2$ is closed under restrictions. Then $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT has an algorithm with savings $f(\cdot, \cdot)$.*

Proof. By assumption, there is an algorithm A_1 which given any $C \in \mathcal{C}_1$ of size m over n variables, outputs a subcube partition labelling for C in time $2^{n-f(n,m)} \operatorname{poly}(m)$. We define an algorithm A to solve mixed instances over $\mathcal{C}_1$ and $\mathcal{C}_2$. Let $C \wedge C'$ be an input to algorithm A, where $C \in \mathcal{C}_1$, $C' \in \mathcal{C}_2$ and the total size of the instance is m. Algorithm A first runs A_1 to give a subcube partition labelling for C. For every element of the list which has $b_i = 1$, the algorithm applies the corresponding restriction ρ_i to C' and solves satisfiability on the resulting instance using the polynomial-time algorithm for $\mathcal{C}_2$-SAT. If any of these calls accept, A accepts, otherwise it rejects. Correctness follows from the definition of subcube partition labellings and closure of C' under restrictions. The resulting algorithm has the same savings as for A_1. □

Corollary 3.3. *Let R be a finite field. For any fixed k, $(k\mathsf{CNF} \wedge \mathsf{THR})$-SAT and $(k\mathsf{CNF} \wedge \mathsf{LIN}_R)$-SAT have savings $\Omega(n/k)$ using a randomized algorithm.*

Proof. The result follows from Theorem 3.2 by using Theorem 2.1 and the facts that THR-SAT and LIN_R-SAT have polynomial-time algorithms. □

There are several interesting cases of mixed instances, where there are no polynomial-time algorithms for pure instances of the constituent types unless $\mathsf{NP} = \mathsf{P}$, however there are algorithms with non-trivial savings for the pure instances, and we would like to get non-trivial savings also for the mixed instances. The following result shows generically how to achieve this, in the case that the algorithms exploit subcube paritioning.

Theorem 3.4. *Let $f_1 : \mathbb{N} \times \mathbb{N}$ and $f_2 : \mathbb{N} \times \mathbb{N}$ be monotone functions such that $f_1(n) \leqslant n$ and $f_2(n) \leqslant n$ for all $n \in \mathbb{N}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be circuit classes such that $\mathcal{C}_1$-SAT has a subcube partitioning algorithm with savings f_1 and $\mathcal{C}_2$-SAT has an algorithm with savings f_2. Moreover, assume $\mathcal{C}_2$ is closed under restrictions. Then $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT has an algorithm with savings $f_2(\Omega(f_1))$.*

Proof. Let the subcube partitioning algorithm for $\mathcal{C}_1$-SAT be A_1 and the algorithm for $\mathcal{C}_2$-SAT be A_2. Given a mixed instance $\phi_1 \wedge \phi_2$, where $\phi_1 \in \mathcal{C}_1$ and $\phi_2 \in \mathcal{C}_2$, we first run A_1 on ϕ_1 to obtain

a subcube partition labelling for ϕ_1 of size at most $2^{n-f_1(n)}\,\mathrm{poly}(m)$, where m is the size of the mixed instance. For each pair (ρ_i, b_i) in the labelling with $b_i = 1$, we run the A_2 on $\phi_2|_{\rho_i}$, where $\phi_2|_{\rho_i}$ denotes ϕ_2 with all values fixed by ρ_i substituted into ϕ_2. We accept iff A_2 accepts for some pair in the labelling with $b_i = 1$.

Correctness of the algorithm follows from correctness of A_1 and A_2, and from the fact that $\mathcal{C}_2$ is closed under restrictions. We now argue the stated bound on the time complexity. Let the subcube partition labelling be $\{(\rho_i, b_i)\}_{i=1}^{s}$, where $s \leqslant 2^{n-f_1(n)}\,\mathrm{poly}(m)$. For each restriction ρ_i in the labelling, let n_i be the number of $*$'s in ρ_i and let $V(\rho_i) = 2^{n_i}$. Since the ρ_i's constitute a subcube partition, we have that $\Sigma_i V(\rho_i) = 2^n$.

We consider two types of restrictions in the labelling, based on whether $n_i \geqslant f_1(n)/2$. If this condition is satisfied, we call the restriction ρ_i fat; otherwise, we call it thin. Now, the time complexity of the algorithm is at most $\Sigma_i 2^{n_i - f_2(n_i)}$, using the assumption on the savings of A_2. We break up this sum into the corresponding sums for fat and thin restrictions. For thin restrictions, we have that the sum is at most $s2^{f_1(n)/2} \leqslant 2^{n-f_1(n)/2}\,\mathrm{poly}(m)$, as there are at most s terms in the sum, and each term is at most $2^{n_i} \leqslant 2^{f_1(n)/2}$. For fat restrictions, we have that the sum is at most $\Sigma_i V(\rho_i)/2^{f_2(f_1(n)/2)}$, using monotonicity of f_2. This sum is at most $2^{n-f_2(f_1(n)/2)}$, using the fact that $\Sigma_i V(\rho_i) \leqslant 2^n$. Thus, the total sum is at most $2^{n-f_2(f_1(n)/2)}\,\mathrm{poly}(m)$, using the fact that $f_2(n) \leqslant n$ for all n. This bound corresponds to savings $f_2(\Omega(f_1))$. □

In the case that f_2 grows much smaller than n, we could optimise the parameters in the proof of Theorem 3.4 to achieve savings $f_2(f_1 - f_2(f_1))$. But in most cases of interest, this optimisation does not give us significant benefits. The proof technique of Theorem 3.4 yields some other consequences. If the algorithms for $\mathcal{C}_1$-SAT and $\mathcal{C}_2$-SAT are polynomial-space, the algorithm for mixed instances can be designed to be polynomial-space as well. Moreover, if the algorithms for pure instances count the number of satisfying assignments, a slight modification to the proof yields an algorithm for mixed instances also counting the number of satisfying assignments. This consequence critically uses the subcube partitioning.

Theorem 3.4 yields the following corollaries, using Theorem 2.1, Theorem 2.2 and Theorem 2.3.

Corollary 3.5. *There are algorithms achieving:*

- *Savings* $\Omega(n/(c^2k))$ *for* $(k\mathsf{CNF} \wedge \mathsf{Formula}[cn])$*-SAT*
- *Savings* $\Omega(n/k\log(m/n))$ *for* $(k\mathsf{CNF} \wedge \mathsf{CNF}[m])$*-SAT*
- *Savings* $\Omega(n/(c^2\log(m/n)))$ *for* $(\mathsf{Formula}[cn] \wedge \mathsf{CNF}[m])$*-SAT*

All of the above algorithms can be modified to count the number of satisfying assignments exactly. We will show in the next section how to get improved savings for items (1) and (2) above. Also note a subtlety here: when there are subcube partitioning algorithms for both $\mathcal{C}_1$-SAT and $\mathcal{C}_2$-SAT, the order in which we run the algorithms might matter in terms of the savings analysis.

We now explain how Theorems 3.2 and 3.4 have interesting consequences for exact algorithms for commonly studied NP-hard graph problems. This involves using the connection in the proof of Theorem 3.1 in the opposite direction, using satisfiability algorithms for mixed instances to get graph algorithms. We are interested particularly here in *meta-algorithmic* results, which show that interesting algorithms exist for a wide class of problems at once. A famous example of such a result is Courcelle's theorem [3], which states that any Monadic Second Order property of graphs can be decided in linear time on graphs of bounded treewidth. Analogously, we wish to have a single

result which implies non-trivial exact algorithms for a large number of NP-hard graph optimisation problems. It is natural to use logical formalisms for graph properties to formulate such a result.

Theorem 3.6. *For any MSO formula ϕ on graphs, the weighted optimisation problems $O_\phi^{\max}$ and $O_\phi^{\min}$ have savings $\Omega(n/\mathsf{polylog}(n))$. For any MSNP formula ϕ on graphs, the weighted optimisation problems $O_\phi^{\max}$ and $O_\phi^{\min}$ have savings $\Omega(n)$.*

Proof. Let ϕ be a MSO formula. The idea of the proof is to "circuitify" the first-order part of ϕ, converting it into a constant-depth circuit, and then to express the weighted optimisation problem as a satisfiability problem on mixed instances, where one part of the mixed instance is a constant-depth circuit and the other part is a linear inequality. When ϕ is MSNP, the circuitification yields a bounded-width CNF formula rather than a constant-depth circuit, which enables us to get better savings.

Assume without loss of generality that ϕ is of the form $\exists X_1 \exists X_2 \ldots \exists X_q \psi(X_1, \ldots, X_q)$, where ψ is a first-order formula. Suppose we are given a weighted graph $G = (V, E, w)$. We give an inductive procedure to construct a constant-depth circuit C_G corresponding to G, which encodes whether $\psi(X_1, \ldots, X_q)$ holds. Let $|V| = n$. The circuit will have qn variables, denoted by $y_{ij}, i = 1, \ldots, q, j = 1, \ldots, n$. We will use $y_{ij} = 1$ to encode that vertex j of the graph belongs to X_i.

By the definition of first-order formulas, ψ is either of the form $\exists x \psi'(X_1, \ldots X_q, x)$, or of the form $\forall x \psi'(X_1, \ldots X_q, x)$, or a propositional sentence constructed from the atomic formulas. In the first case, we cycle over the n vertices of the graph. For each vertex k, by induction, there is a constant-depth circuit $C'(k)$ in the y variables for ψ' - we define C_G to be the OR over all n of $C'(k)$. Similarly, in the second case, we define C_G to be the AND over all n of $C'(k)$. In the third case, we can express the sentence as a bounded-width CNF in the variables $\{y_{ij}\}$ after substituting occurrences of $E(j, k)$ for vertices $j, k \in V$ by true or false depending on whether $(j, k) \in E$ or not, and similarly substituting occurrences of $j = k$ by true or false, depending on whether $j = k$ or not. Any occurrence of $X_i(j)$ is replaced by the variable y_{ij}.

The circuit C_G has a fixed depth which depends on the quantifier depth of the MSO formula ϕ. Now to solve the weighted optimisation problem $O_\phi^{\max}$, we construct mixed instances of the form $C_G \wedge T$, where T is a linear inequality we choose adaptively. Let $W_{\max} = q\Sigma_{v \in V} w(v)$. We use binary search to find the maximum weight W for which the mixed instance $C_G \wedge T$ is satisfiable, where T is the linear inequality $\Sigma_{i,j} w_j y_{ij} \geqslant W$. We initialize W to $W_{\max}/2$, using the binary search method to update W depending on the result of our satisfiability query. We will use at most $\log(|W_{\max}|)$ calls to the satisfiability algorithm, and by our assumption that the weights are representable by $\mathrm{poly}(n)$ bits, this will incur at most a polynomial overhead over the running time of a single call to the satisfiability algorithm. For the satisfiability algorithm itself, we apply Theorem 3.2 with Theorem 2.2 and the fact that THR-SAT has a polynomial-time algorithm. This gives us savings $\Omega(n/\mathsf{polylog}(n))$.

Solving $O_\phi^{\min}$ is completely analogous, except that we attempt to find the minimum W for which the mixed instance is satisfiable, where the linear inequality is now that $\Sigma_{i,j} w_j y_{ij} \leqslant W$.

In the case where ϕ is MSNP, the circuit C_G produced by our procedure is in fact a k-CNF for some fixed k. Hence in this case, we can use Corollary 3.3 to achieve linear savings. □

Algorithms with linear savings are known for a large number of graph optimisation problems, and Theorem 3.6 brings these linear savings results under one umbrella, for problems such as Independent Set, Clique and Vertex Cover.

4 Beating the Generic Subcube Partitioning Bound

Corollary 3.5 illustrates how Theorem 3.4 can be used to give non-trivial savings for several satisfiability problems with mixed instances. However, the savings obtained in these cases are not the best for which one could hope. In general, if $\mathcal{C}_1$-SAT has savings f_1 and $\mathcal{C}_2$-SAT has savings f_2, we could hope for savings $\min\{f_1, f_2\}$ for $(\mathcal{C}_1 \wedge \mathcal{C}_2)$-SAT. Note that savings asymptotically better than this would imply better algorithms for the pure satisfiability problems $\mathcal{C}_1$-SAT or $\mathcal{C}_2$-SAT. We are able to achieve the optimal savings for mixed instances where one part of the instance is a k-CNF and the other part consists of a prescribed number of clauses of arbitrary length. To achieve these improved savings, we exploit the fact that the best known algorithms for CNF-SAT themselves proceed through reductions to satisfiability of bounded-width formulas.

Theorem 4.1. *For any positive integer $k \geqslant 2$ and any function $m : \mathbb{N} \to \mathbb{N}$, $(k\mathsf{CNF} \wedge \mathsf{CNF}[m])$-SAT has savings $\Omega(\min\{n/k, n/\log(m/n)\})$.*

Proof. Let $\phi = \phi_1 \wedge \phi_2$ be the input formula on n variables, where ϕ_1 is a k-CNF and ϕ_2 is a CNF with $m' \leqslant m(n)$ clauses. We apply a width reduction procedure due to Schuler [21], and then use a standard algorithm for bounded-width satisfiability [16]. We use the tighter analysis of width-reduction due to [2].

Let $K \geqslant k$ be a parameter to be fixed later. We apply the following recursive procedure to solve satisfiability on ϕ. If ϕ is a K-CNF, we use the PPZ satisfiability algorithm [16], which runs in time $2^{n-n/K}\,\mathrm{poly}(n)$, to check if ϕ is satisfiable. If not, then pick the lexicographically first clause C in ϕ_2 with width greater than K. Assume wlog that the first K literals of C are $x_1, \ldots, x_K$. We construct instances ϕ' and ϕ'' as follows and recursively check satisfiability on these instances. ϕ' is produced by substituting $x_1, \ldots, x_K$ to false in ϕ and simplifying the resulting formula. ϕ'' consists of ϕ, but with the clause C removed. Clearly, ϕ is satisfiable iff at least one of ϕ' and ϕ'' are satisfiable.

This recursive procedure for satisfiability corresponds to a recursion tree where left branches represent substitutions for K literals, and right branches represent removals of a clause. The leaves of this tree are labelled with K-CNFs, corresponding to using the PPZ satisfiability algorithm rather than continuing to use recursion. This tree is highly skewed: there can be at most n/K left branches, as each left branch gets rid of K variables, but there can be as many as m' right branches. The number of paths in the tree with r left branches is at most $\binom{m'+r}{r}$, and each leaf corresponding to such a path is labelled with a formula on at most $n - Kr$ variables.

Let $\alpha = 2^{1-1/K}$. We can estimate the total running time of the satisfiability procedure as at most:

$$\begin{aligned}
\sum_{r=0}^{n/K} \binom{m'+r}{r} \alpha^{n-Kr} &\leqslant \sum_{r=0}^{m'+n/K} \binom{m'+n/K}{r} \alpha^{n-Kr} \\
&\leqslant \alpha^n (1+\alpha^{-K})^{m'+n/K} \\
&\leqslant \alpha^n e^{\alpha^{-K}(m'+n/K)} \\
&\leqslant 2^{n-n/K+2(m'+n/K)/2^{K-1}}
\end{aligned}$$

To fix K, we consider two cases: either $m' \geqslant 2^k n$, or it is not. In the first case, by setting $K = C\log(m'/n)$ for large enough constant C and using the fact that $k \geqslant 2$, it can easily be checked that the savings of the satisfiability procedure is $\Omega(n/K)$. In the second case, by setting

$K = 3k + 1$, it can be checked that the savings of satisfiability procedure is $\Omega(n/K)$. Thus the savings of the procedure is $\Omega(\min\{n/k, n/\log(m'/n)\}) \geqslant \Omega(\min\{n/k, n/\log(m/n)\})$, as promised, since $m' \leqslant m$. □

For mixed instances where one part of the instance is a Boolean formula with a prescribed number of literals, and the other part is a k-CNF, we can again use specific properties of known algorithms to combine them more cleverly than in the proof of the generic subcube paritioning bound. However, we aren't quite able to manage optimal savings in this case.

Theorem 4.2. ($\mathsf{Formula}[cn] \wedge k\mathsf{CNF}$)*-SAT has (randomized zero-error) savings at least* $\Omega(\min\{n/c^4, n/k\})$.

Proof. Let $\phi_1 \wedge \phi_2$ be a given instance where ϕ_1 is a de Morgan formula of size cn, and ϕ_2 is a k-CNF formula, We first greedily restrict heavy variables in ϕ_1. Let H be the set of variables appearing at least $2c$ times in ϕ_1. Then $|H| \leqslant n/2$. We build $2^{|H|}$ branches by restricting variables in H. Let $n' = n - |H| \geqslant n/2$. For each restriction τ of variables in H, we have that $L(\phi_1|_\tau) \leqslant cn - 2c|H| \leqslant cn'$, and each variable in $\phi_1|_\tau$ appears less than $2c$ times.

For each τ, let $\phi_1' = \phi_1|_\tau$ and $\phi_2' = \phi_2|_\tau$; we next apply random restrictions to $\phi_1' \wedge \phi_2'$. Let $p = \min\{1/(20c)^2, 1/28k\}$. Consider the random restriction $\mathcal{R}_p = (U, \sigma)$ where we first choose a random subset U of pn' variables, and then fix variables not in U by a random assignments $\sigma \in \{0,1\}^{n'-|U|}$.

For a random $\rho \sim \mathcal{R}_p$, by Lemma 2.4, the probability that $\phi_1'|_\rho$ depends on at least $\frac{3}{5}pn'$ variables is at most $2^{-\Omega(\min\{n'/c^4,\ pn'\})}$. By Lemma 2.6,

$$\mathbf{Pr}_{\rho \sim (U,\sigma)}[D(\phi_2'|_\rho) \geqslant s] \leqslant (7pk)^s \leqslant 4^{-s}.$$

For each ρ, if $\phi_1'|_\rho$ depends on at least $\frac{3}{5}pn'$ variables, then we enumerate assignments to all remaining pn' variables and check the satisfiability of $\phi_1'|_\rho \wedge \phi_2'|_\rho$. Otherwise, we enumerate assignments to at most $\frac{3}{5}pn'$ variables on which $\phi_1'|_\rho$ depends (this fixes $\phi_1'|_\rho$ to a constant), and then build decision trees for the restricted $\phi_2'|_\rho$.

For a random $\rho \sim \mathcal{R}_p$, the expected number of branches we build for checking the satisfiability of $\phi_1'|_\rho \wedge \phi_2'|_\rho$ is at most

$$\begin{aligned} M &= 2^{pn'} \cdot 2^{-\Omega(\min\{n'/c^4,\ pn'\})} + 2^{3pn'/5} \cdot \sum_s 2^s 4^{-s} \\ &\leqslant 2^{pn' - \Omega(\min\{n'/c^4,\ pn'\})}. \end{aligned}$$

Finally, the expected total number of branches (and the expected running time of the algorithm) is bounded by

$$2^{|H|+n'-pn'} M \leqslant 2^{n - \Omega(\min\{n/c^4,\ pn\})} = 2^{n - \Omega(\min\{n/c^4,\ n/k\})}.$$

□

Note that Theorem 4.2 doesn't always yield savings better than Corollary 3.5, however for large k the savings is better, and in general we can achieve savings of the form $\Omega(\max\{n/(c^2k), \min\{n/c^4, n/k\}\})$ by using either the algorithm of Corollary 3.5 or the algorithm of Theorem 4.2, based on the relationship between c and k.

5 Exploiting Subcube Partitioning for Composed Instances

We now consider composed instances of the form $\mathcal{C} \circ \mathcal{D}$, where each instance has a $\mathcal{C}$-circuit at the top whose inputs are $\mathcal{D}$-circuits. We hope to design efficient algorithms for satisfiability checking or truth-table enumeration, by exploiting existing efficient algorithms for pure instances $\mathcal{C}$ and $\mathcal{D}$.

To start, we describe a simple strategy. Let $C(D_1, \ldots, D_m)$ be an instance of $\mathcal{C} \circ \mathcal{D}$, where D_i's are over the same set of n variables. We first run a partitioning which works for all circuits $D_1, \ldots, D_m$ such that in each part j, each D_i reduces to a $\mathcal{D}'$ instance $D'_{i,j}$, and we get a $\mathcal{C} \circ \mathcal{D}'$ instance $C(D'_{1,j}, \ldots, D'_{m,j})$; then, run an efficient algorithm for $\mathcal{C} \circ \mathcal{D}'$. Obviously, to implement this strategy we need both (1) an efficient partitioning for a collection of $\mathcal{D}$-circuits, and (2) an efficient algorithm for $\mathcal{C} \circ \mathcal{D}'$.

Indeed, the AC^0 satisfiability algorithm of [6] can be viewed as a recursive application of the above strategy. First, an AC^0_d circuit (with OR gates at the bottom) can be viewed as an AC^0_{d-1} circuit fed by clauses. Using Schuler's width reduction [21], clauses of arbitrary length can be reduced to clauses of length at most k; this gives AC^0_d circuits with bottom fan-in at most k, which can be viewed as $\mathsf{AC}^0_{d-2} \circ k\mathsf{CNF}$. By an extension of Hastad's switching lemma [6], there is a (randomized) partitioning for a collection of $k\mathsf{CNF}$'s such that restricted $k\mathsf{CNF}$'s can be written as $k\mathsf{DNF}$'s; by merging into AC^0_{d-2}, this gives $\mathsf{AC}^0_{d-3} \circ k\mathsf{DNF}$ circuits. Then by recursively reducing the depth, we finally get a $k\mathsf{CNF}$ or $k\mathsf{DNF}$ for each part. One subtlety is that, the partitioning given in [6] there is not a subcube partitioning, but instead each part is defined by a $k\mathsf{CNF}$; however, the $k\mathsf{CNF}$ specifying a part can be combined with the final restricted instance ($k\mathsf{CNF}$ or $k\mathsf{DNF}$), which has a subcube partitioning by the extension of Hastad's switching lemma in [6].

5.1 QBF-SAT

We next apply the strategy to give a non-trivial satisfiability algorithm for quantified de Morgan formulas of superlinear size, by designing an efficient truth-table enumeration algorithm for composed instances $\mathsf{AC}^0 \circ \mathsf{Formula}$.

Theorem 5.1. *For quantified de Morgan formulas with n variables, q quantifier blocks and size at most $n^{5/4-\epsilon}$ for any $\epsilon > 0$, there is a randomized satisfiability algorithm running in time $2^{n-n^{\Omega(\epsilon/(q+1))}}$.*

Before stating the proof, we outline the main ideas below. Given a QBF, we first use the approach of [19] to "blowup" the instance; that is, enumerate assignments to the innermost n^δ quantified variables and construct an instance of $\mathsf{AC}^0 \circ \mathsf{Formula}$. Following [19], if we can enumerate the truth table of $\mathsf{AC}^0 \circ \mathsf{Formula}$ efficiently, then the original QBF can be evaluated efficiently (with the savings coming from the blowup). To produce the truth table of $\mathsf{AC}^0 \circ \mathsf{Formula}$, we first apply random restrictions to shrink all formulas such that they can be merged into AC^0, and then use fast truth-table enumeration for AC^0 [19].

We first show that the truth table of $\mathsf{AC}^0 \circ \mathsf{Formula}$ (when the formula is small) can be efficiently enumerated. We need the following lemma on AC^0 truth-table enumeration [19].

Lemma 5.2 ([19])**.** *There is a randomized zero-error algorithm that, given an AC^0 circuit of depth d and size s, outputs the truth table in time $\mathrm{poly}(n) \cdot (2^n + s2^{n-\Omega(n/\log^{d-1} s)})$. In particular, there is some small constant $a > 0$ such that when $s \leqslant 2^{an^{1/d}}$, the algorithm runs in time $2^n \,\mathrm{poly}(n)$.*

Lemma 5.3. *There is a randomized zero-error algorithm running in time* $2^n \operatorname{poly}(n)$ *which outputs truth tables for* $\mathsf{AC}^0 \circ \mathsf{Formula}$ *instances satisfying the following conditions:*

- *the* AC^0 *circuit has depth* d *and size* $s \leqslant 2^{n^\delta}$*;*
- *each formula feeding into* AC^0 *has size* $L \leqslant n^{5/4-\epsilon}$*, and each variable appears* $O(L/n)$ *times in each formula;*
- $\delta \leqslant a\epsilon/(d+1)$ *for some small constant* $a > 0$*.*

Proof. Consider the random restriction $\mathcal{R}_p = (U, \sigma)$, where we first choose a random subset U of pn variables, and then choose a random assignment $\sigma \in \{0,1\}^{n-pn}$ to variables not in U. Let $p = n^{\epsilon/2}/n$. Let $\epsilon' = 2\delta < \epsilon/2$. For a formula G feeding into AC^0, define

$$P_U = \mathbf{Pr}_{\rho\sim(U,\sigma)}[\ G|_\rho \text{ depends on } \geqslant n^{\epsilon'} \text{ variables }].$$

We say U is *good* for G if $P_U < 2^{-\Omega(n^{\epsilon'})}$. Lemma 2.5 gives that, $\mathbf{E}_U[P_U] < 2^{-cn^{\epsilon'}}$, for some constant $c > 0$. Then by Markov's inequality,

$$\mathbf{Pr}_U[P_U \geqslant 2^{-cn^{\epsilon'}/2}] \leqslant \frac{\mathbf{E}_U[P_U]}{2^{-cn^{\epsilon'}/2}} < 2^{-cn^{\epsilon'}/2}.$$

That is, a random U is good for G with probability $1 - 2^{-\Omega(n^{\epsilon'})}$.

By a union bound on the $s \leqslant 2^{n^\delta}$ formulas feeding to AC^0, a random U is good for all formulas with probability $1 - 2^{n^\delta - \Omega(n^{\epsilon'})} \geqslant 1 - 2^{-\Omega(n^{\epsilon'})}$,

The algorithm runs by choosing U randomly, and then enumerating assignments to variables not in U. For a fixed good U, and a randomly chosen $\sigma \in \{0,1\}^{n-|U|}$, by another union bound on all formulas, the probability that any of the restricted formula depends on more than $n^{\epsilon'}$ variables is at most $2^{n^\delta - \Omega(n^{\epsilon'})} \leqslant 2^{-\Omega(n^{\epsilon'})}$.

This means, over the $2^{n-|U|}$ assignments to variables not in U, all but $2^{-\Omega(n^{\epsilon'})}$ fraction will give restricted instances with an AC^0 of depth d and size 2^{n^δ} at the top, and formulas each depending on at most $n^{\epsilon'}$ variables at the bottom. For each such instance, we express formulas as CNFs/DNFs (of size $2^{n^{\epsilon'}}$) and merge them into AC^0. This gives an AC^0 circuit of depth $d+1$ and size $2^{n^\delta + n^{\epsilon'}}$. (Note that, it has n^ϵ variables unfixed.) Then we use the truth-table enumeration algorithm for AC^0 by Lemma 5.2 [19]. Since $\epsilon' = 2\delta$ and $\delta = a\epsilon/(d+1)$, for sufficiently small a, by Lemma 5.2, this can be done in time $2^{n^\epsilon} \operatorname{poly}(n)$.

Thus, the running time for branches where all formulas depend on at most $n^{\epsilon'}$ variables is at most $2^{n-n^\epsilon} \cdot 2^{n^\epsilon} \cdot \operatorname{poly}(n) = 2^n \operatorname{poly}(n)$.

For branches where at least one of the formulas depend on more than $n^{\epsilon'}$ variables, we use brute-force enumeration for the remaining n^ϵ variables to evaluate the circuit. Each such branch takes time $2^{n^\epsilon} \cdot 2^{n^\delta} \cdot \operatorname{poly}(n)$. The running time for all such branches is

$$2^{n-n^\epsilon} \cdot 2^{-\Omega(n^{\epsilon'})} \cdot (2^{n^\epsilon} \cdot 2^{n^\delta} \cdot \operatorname{poly}(n)) \leqslant 2^n \operatorname{poly}(n).$$

□

Proof of Theorem 5.1. Consider a QBF with n variables and q quantifier blocks. Let F be the de Morgan formula in the QBF, where the size $L(F) \leqslant n^{5/4-\epsilon}$. Let H be the set of variables appearing at least $2L(F)/n$ times; then $|H| \leqslant n/2$. Our algorithm has the following stages:

We first enumerate on the innermost n^δ quantified variables, for $\delta = a\epsilon/(d+1)$ for some small constant $a > 0$. Let B be the innermost n^δ quantified variables. Build an AC^0 circuit by enumerating assignments to all variables in B (replacing existential quantifiers by ORs and universal quantifiers by ANDs), and let the bottom layer be fed by F under corresponding restrictions of B. This gives an $\mathsf{AC}^0 \circ \mathsf{Formula}$ instance over $n - n^\delta$ variables, where the top AC^0 circuit has depth q and size 2^{n^δ}, and each formula feeding into AC^0 has size at most $L(F)$.

If we can enumerate the truth table of this composed instance efficiently (in time $2^{n-n^\delta}\,\mathrm{poly}(n)$), then by the techniques of [19], the original QBF can be evaluated by traversing an AND-OR tree based on the remaining quantified variables, and looking up in the truth table; the running time for QBF-SAT will be $2^{n-n^\delta}\,\mathrm{poly}(n)$.

In the rest, we focus on enumerating the truth table of the composed $\mathsf{AC}^0 \circ \mathsf{Formula}$ instance. We first enumerate assignments to variables in $H \setminus B$, get at most $2^{|H|}$ branches each with a restricted instance where the formulas have no heavy variables. For each fixing of variables in $H \cup B$, since $|H| \leqslant n/2$ and $|B| = n^\delta$, the number of unfixed variables is $n' = n - n^\delta - |H \setminus B| = \Theta(n)$. Thus, each restricted $\mathsf{AC}^0 \circ \mathsf{Formula}$ instance has an AC^0 of depth q and size $2^{O(n'^\delta)}$) at the top, and each formula feeding into AC^0 has size at most $L' = O(n'^{5/4-\epsilon})$, with all variables appear $O(L'/n')$ times. Then by Lemma 5.3, the truth table of each such restricted instance can be enumerated in time $2^{n'}\,\mathrm{poly}(n)$.

Therefore, we can enumerate the truth table of the constructed $\mathsf{AC}^0 \circ \mathsf{Formula}$ instance in time $2^{|H\setminus B|} \cdot 2^{n'}\,\mathrm{poly}(n) = 2^{n-n^\delta}\,\mathrm{poly}(n)$, and thus QBF-SAT is in time $2^{n-n^\delta}\,\mathrm{poly}(n) = 2^{n-n^{\Omega(\epsilon/(q+1))}}\,\mathrm{poly}(n)$. □

6 Acknowledgments

Supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agreement no. 615075

References

[1] P. Beame. A switching lemma primer, 1994.

[2] C. Calabro, R. Impagliazzo, and R. Paturi. A duality between clause width and clause density for sat. In *Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, CCC '06, 2006.

[3] B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[4] B. Doerr. Analyzing randomized search heuristics: Tools from probability theory. In A. Auger and B. Doerr, editors, *Theory of Randomized Search Heuristics*, pages 1–20. World Scientific Publishing, 2011.

[5] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[6] R. Impagliazzo, W. Matthews, and R. Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 961–972, 2012.

[7] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science*, pages 653–662, 1998.

[8] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[9] I. Komargodski and R. Raz. Average-case lower bounds for formula size. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 171–180, 2013.

[10] I. Komargodski, R. Raz, and A. Tal. Improved average-case lower bounds for demorgan formula size. In *Proceedings of the Fifty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2013.

[11] S. Malik and L. Zhang. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76–82, 2009.

[12] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. 2+p-SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Structures and Algorithms*, 15:414–440, 1999.

[13] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract davis-putnam-logemann-loveland procedure to DPLL(T). *Journal of the Association for Computing Machinery*, 53(6):937–977, 2006.

[14] M. Patrascu and R. Williams. On the possibility of faster SAT algorithms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075, 2010.

[15] R. Paturi, P. Pudlák, M. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

[16] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Chicago Journal of Theoretical Computer Science*, 1999.

[17] S. Porschen and E. Speckenmeyer. Satisfiability of mixed horn formulas. *Discrete Applied Mathematics*, 155(11):1408–1419, 2007.

[18] R. Santhanam. Fighting perebor: New and improved algorithms for formula and qbf satisfiability. In *Proceedings of the Fifty-First Annual IEEE Symposium on Foundations of Computer Science*, pages 183–192, 2010.

[19] R. Santhanam and R. Williams. Beating exhaustive search for quantified boolean formulas and connections to circuit complexity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 231–241, 2015.

[20] U. Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, FOCS'99*, pages 410–414, 1999.

[21] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, 54(1):40–44, 2005.

[22] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, 2010.

[23] R. Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the Twenty-Sixth Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.

A Shrinkage of Formulas under Random Restrictions

We prove concentrated shrinkage of de Morgan formulas under random restrictions. The results here, in terms of the concentration parameters, are weaker than those obtained via greedy restrictions [18], or mixed greedy and random restrictions [9, 10], but they can be applied to get satisfiability algorithms for mixed and composed instances.

We need the following version of Chernoff bounds on hypergeometric distributions.

Theorem A.1 (Chernoff bound [4])**.** *Let U be a subset of size pn chosen uniformly at random from $\{1,\ldots,n\}$. For $i=1,\ldots,n$, let $X_i=1$ if $i\in U$, and $X_i=0$ otherwise. Let $X=\sum_{i=1}^n a_iX_i$ where $a_i\in[0,b]$. Then, $\mathbf{E}[X]=p\sum_{i=1}^n a_i$, and, for $t\geqslant 6\mathbf{E}[X]$,*

$$\mathbf{Pr}[X\geqslant t]\leqslant 2^{-t/b}.$$

A sequence of random variables $X_0,X_1,\ldots,X_n$ is called a *supermartingale* with respect to another sequence of random variables $R_1,\ldots,R_n$ if $\mathbf{E}[X_i\mid R_{i-1},\ldots,R_1]\leqslant X_{i-1}$, for $1\leqslant i\leqslant n$.

Theorem A.2 (Azuma-Hoeffding Inequality [4])**.** *If $\{X_i\}_{i=0}^n$ is a supermartingale such that $|X_i-X_{i-1}|\leqslant c_i$ for $i=1,\ldots,n$, then, for any $\lambda\geqslant 0$,*

$$\mathbf{Pr}[X_n-X_0\geqslant\lambda]\leqslant\exp\left(-\frac{2\lambda^2}{\sum_{i=1}^n c_i^2}\right).$$

Let $\mathcal{R}_p$ be the random restriction where a subset U of pn variables is chosen uniformly at random, and each variable not in U is assigned 0 or 1 each with probability 1/2.

Lemma A.3. *Let F be a de Morgan formula of size cn on n variables, where each variable appears at most $O(c)$ times in F. Then for $pn\geqslant l^2c^2$,*

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[L(F|_\rho)\geqslant 2p^{1.5}cn=2pn/l]\leqslant 2^{-\Omega(pn/l^2c^2)}.$$

In particular, for $l=20$ and $p=1/(20c)^2$,

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[L(F|_\rho)\geqslant pn/10]\leqslant 2^{-\Omega(n/c^4)};$$

for $c=n^{\frac{1}{4}-\epsilon}$, $l=20n^{\epsilon/2}$ and $p=1/(400n^{\frac{1}{2}-\epsilon})$,

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[L(F|_\rho)\geqslant pn/10n^{\epsilon/2}]\leqslant 2^{-\Omega(n^{2\epsilon})}.$$

Proof. Consider a step-by-step process where at each step a variable is randomly picked and fixed, and we run this process for $n - pn$ steps. Let $F_0 = F$, and F_i be the restricted formula after the i-th step (which is simplified by eliminating redundant leaves). Note that F_i is over $n-i$ variables. We distinguish two cases based on the size of F_i.

First consider the case that F_i is already small for some $i \leqslant n - pn$, that is, $L(F_i) \leqslant (n-i)/3l$. We then apply a random restriction $\mathcal{R}_{p'}$ for $p' = pn/(n-i)$ on F_i; the restricted formula size is at most the number of leaves left. By Theorem A.1,

$$\mathbf{E}_{\rho\sim\mathcal{R}_{p'}}[L(F_i|_\rho)] \leqslant p' L(F_i) \leqslant pn/3l,$$

and, since each variable appears at most $O(c)$ times,

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_{p'}}[L(F_i|_\rho) \geqslant 2pn/l] \leqslant 2^{-\Omega(pn/lc)}.$$

In the rest, we assume that $L(F_i) > (n-i)/3l$ for $i = 1, \ldots, n - pn$.

At the i-th step, we start with a formula F_{i-1} on $n-(i-1)$ variables. We have $(n-i+1)/3l < L(F_{i-1}) \leqslant cn$, and each variable appears $O(c)$ times. For a variable x in F_i, let c_x be the number of times where x appears in the form of $x \wedge G$ or $\overline{x} \vee G$, and let c'_x be the number of times where x appears in the form of $x \vee G$ or $\overline{x} \wedge G$. Note that we can eliminate $2c_x + c'_x$ leaves if we fix $x = 0$, and eliminate $c_x + 2c'_x$ leaves if we fix $x = 1$. We randomly choose x from the remaining variables and randomly fix it; define the random variable

$$Y_i = \begin{cases} 2c_x + c'_x, & \text{if } x \text{ is chosen and assigned 0} \\ c_x + 2c'_x, & \text{if } x \text{ is chosen and assigned 1.} \end{cases}$$

Since $\sum_x (c_x + c'_x) = L(F_{i-1})$ and $0 \leqslant \{c_x, c'_x\} \leqslant O(c)$, we have $\mathbf{E}[Y_i] = 1.5L(F_{i-1})/(n-i+1)$, and $Y_i \leqslant O(c)$.

Define $L_i^* = L(F_{i-1}) - Y_i$, then we have $L(F_i) \leqslant L_i^* \leqslant L(F_{i-1})$. For a fixed F_{i-1},

$$\begin{aligned} \mathbf{E}[L_i^*] = L(F_{i-1}) - \mathbf{E}[Y_i] &= L(F_{i-1})\left(1 - \frac{1.5}{n-i+1}\right) \\ &\leqslant L(F_{i-1})\left(1 - \frac{1}{n-i+1}\right)^{1.5}. \end{aligned}$$

We wish to compare L_i^* with $L(F_{i-1})\left(1 - \frac{1}{n-i+1}\right)^{1.5}$. Define

$$Z_i = \ln\left(\frac{L_i^*}{L(F_{i-1})\left(1 - \frac{1}{n-i+1}\right)^{1.5}}\right).$$

Then by Jensen inequality, $\mathbf{E}[Z_i] \leqslant 0$ (conditioning on a fixed F_{i-1}).

Since $L_i^* \leqslant L(F_{i-1})$, we get

$$Z_i \leqslant -1.5 \ln\left(1 - \frac{1}{n-i+1}\right) = 1.5 \ln\left(1 + \frac{1}{n-i}\right) \leqslant \frac{1.5}{n-i},$$

where the last inequality is by the fact that $\ln(1+x) \leqslant x$.

Since $Y_i \leqslant O(c)$ and $L(F_{i-1}) > (n-i+1)/3l$,

$$\begin{aligned} L_i^* &= L(F_{i-1}) - Y_i = L(F_{i-1})\left(1 - \frac{Y_i}{L(F_{i-1})}\right) \\ &\geqslant L(F_{i-1})\left(1 - \frac{blc}{n-i+1}\right), \end{aligned}$$

for some constant b. This gives that,

$$\begin{aligned} Z_i &\geqslant \ln\left(\frac{L_i^*}{L(F_{i-1})}\right) \geqslant \ln\left(1 - \frac{blc}{n-i+1}\right) \\ &= -\ln\left(1 + \frac{blc}{n-i+1-blc}\right) \\ &\geqslant -\frac{blc}{n-i+1-blc} \geqslant -\frac{2blc}{n-i}, \end{aligned}$$

where the last inequality follows from $n - i \geqslant pn \gg lc$. Therefore, $|Z_i| \leqslant \frac{2blc}{n-i}$.

Let $X_0 = 0$ and $X_i = \sum_{j=1}^{i} Z_i$. Then $\{X_i\}$ is a supermartingale with respect to the random choices at each step. By Theorem A.2 (Azuma-Hoeffding inequality), for $\lambda > 0$,

$$\begin{aligned} \mathbf{Pr}[X_i = \sum_{j=1}^{i} Z_i \geqslant \lambda] &\leqslant \exp\left(-\frac{2\lambda^2}{\sum_{j=1}^{i}(\frac{2blc}{n-j})^2}\right) \\ &\leqslant \exp\left(-\frac{2\lambda^2(n-i-1)}{(2blc)^2}\right), \end{aligned}$$

by that $\sum_{j=1}^{i} \frac{1}{(n-j)^2} < \sum_{j=1}^{i} \frac{1}{n-j-1} - \frac{1}{n-j} < \frac{1}{n-i-1}$.

Since that

$$\begin{aligned} e^{X_i} &= \prod_{j=1}^{i} \frac{L_j^*}{L(F_{j-1})\left(1 - \frac{1}{n-j+1}\right)^{1.5}} \\ &\geqslant \prod_{j=1}^{i} \frac{L(F_j)}{L(F_{j-1})\left(1 - \frac{1}{n-j+1}\right)^{1.5}} = \frac{L(F_i)}{L(F_0)\left(\frac{n-i}{n}\right)^{1.5}}, \end{aligned}$$

we have

$$\mathbf{Pr}\left[L(F_i) \geqslant e^{\lambda} L(F_0)\left(\frac{n-i}{n}\right)^{1.5}\right] \leqslant \exp\left(-\frac{2\lambda^2(n-i-1)}{(2blc)^2}\right).$$

Let $\lambda = \ln 2$ and $i = n - pn$,

$$\mathbf{Pr}\left[L(F_{n-pn}) \geqslant 2cnp^{1.5}\right] \leqslant 2^{-\Omega\left(pn/l^2c^2\right)}.$$

□

Lemma A.4 (Lemma 2.4 restated)**.** *Let F be a de Morgan formula of size $L = cn$ where each variable appears at most $O(c)$ times. Then, for $p \leqslant \frac{1}{(20c)^2}$,*

$$\mathbf{Pr}_{\rho \sim \mathcal{R}_p}[\ F|_\rho \text{ depends on } \geqslant \tfrac{3}{5}pn \text{ variables }] < 2^{-\Omega(\min\{n/c^4,\ pn\})}.$$

Proof. We first apply a random restriction $\mathcal{R}_{p_0}$ for $p_0 = 1/(20c)^2$. By Lemma A.3, with probability $1-2^{-\Omega(n/c^4)}$, the restricted formula has size at most $p_0 n/10$, and thus depending on at most $p_0 n/10$ variables (although there are $p_0 n$ variables left).

Then, apply another random restriction $\mathcal{R}_{p_1}$ for $p_1 = p/p_0$, leaving pn variables unfixed. For any restricted formula F' depending on at most $p_0 n/10$ variables, by Chernoff bound (Theorem A.1), in expectation, the restricted formula $F'|_\rho$ depends on at most $p_1 \cdot p_0 n/10 = pn/10$ variables, and

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_{p_1}}[\ F'|_\rho \text{ depends on } \geqslant \tfrac{3}{5}pn \text{ variables }] < 2^{-\Omega(pn)}.$$

Therefore,

$$\begin{aligned} & \mathbf{Pr}_{\rho\sim\mathcal{R}_p}[\ F|_\rho \text{ depends on } \geqslant \tfrac{3}{5}pn \text{ variables }] \\ < \ & 2^{-\Omega(n/c^4)} + 2^{-\Omega(pn)} < 2^{-\Omega(\min\{n/c^4,\ pn\})}. \end{aligned}$$

□

Lemma A.5 (Lemma 2.5 restated)**.** *Let F be a de Morgan formula of size $L \leqslant n^{5/4-\epsilon}$ where each variable appears at most $O(L/n)$ times. Then, for $p = n^{\epsilon/2}/n$ and any constant $\epsilon' < \epsilon/2$,*

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_p}[\ F|_\rho \textit{ depends on } \geqslant n^{\epsilon'} \textit{ variables }] < 2^{-\Omega(n^{\epsilon'})}.$$

Proof. We first apply a random restriction for $p_0 = 1/400n^{\frac{1}{2}-\epsilon}$. By Lemma A.3, with probability $1-2^{-\Omega(n^{2\epsilon})}$, the restricted formula has size at most $p_0 n/10n^{\epsilon/2}$, and thus depending on at most $p_0 n/10n^{\epsilon/2}$ variables (although there are $p_0 n$ variables left). For any such "small" restricted formula F', apply another random restriction $\mathcal{R}_{p_1}$ for $p_1 = n^{\epsilon/2}/(p_0 n)$, leaving $n^{\epsilon/2}$ variables unfixed. By Chernoff bound (Theorem A.1), in expectation, the restricted formula depends on at most $p_1 \cdot (p_0 n/10n^{\epsilon/2}) = 1/10$ variables, and

$$\mathbf{Pr}_{\rho\sim\mathcal{R}_{p_1}}[\ F'|_\rho \text{ depends on } \geqslant n^{\epsilon'} \text{ variables }] < 2^{-\Omega(n^{\epsilon'})}.$$

Therefore,

$$\begin{aligned} & \mathbf{Pr}_{\rho\sim\mathcal{R}_p}[\ F|_\rho \text{ depends on } \geqslant n^{\epsilon'} \text{ variables }] \\ < \ & 2^{-\Omega(n^{2\epsilon})} + 2^{-\Omega(n^{\epsilon'})} < 2^{-\Omega(n^{\epsilon'})}. \end{aligned}$$

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de