Electronic Colloquium on Computational Complexity, Report No. 112 (2015)

Improved Algorithms for Sparse MAX-SAT and MAX-k-CSP*

Ruiwen Chen, Rahul Santhanam

School of Informatics, University of Edinburgh, Edinburgh, UK;
rchen2@inf.ed.ac.uk, rsanthan@inf.ed.ac.uk

Abstract. We give improved deterministic algorithms solving sparse instances of MAX-SAT and MAX-k-CSP. For instances with n variables and cn clauses (constraints), we give algorithms running in time $\text{poly}(n) \cdot 2^{n(1-\mu)}$ for

- $\mu = \Omega(\frac{1}{c})$ and polynomial space solving MAX-SAT and MAX-k-SAT,
- $\mu = \Omega(\frac{1}{\sqrt{c}})$ and exponential space solving MAX-SAT and MAX-k-SAT,
- $\mu = \Omega(\frac{1}{ck^2})$ and polynomial space solving MAX-k-CSP,
- $\mu = \Omega(\frac{1}{\sqrt{ck^3}})$ and exponential space solving MAX-k-CSP.

The previous MAX-SAT algorithms have savings $\mu = \Omega(\frac{1}{c^2 \log^2 c})$ for running in polynomial space [15] and $\mu = \Omega(\frac{1}{c \log c})$ for exponential space [5]. We also give an algorithm with improved savings for satisfiability of depth-2 threshold circuits with cn wires.

Keywords: satisfiability algorithm, MAX-SAT, MAX-k-CSP

1 Introduction

The maximum satisfiability problem (MAX-SAT) is to find an assignment that maximizes the number of satisfied clauses in a CNF formula. MAX-k-SAT is the special case where all clauses have at most k literals. For instances with n variables and $m = cn$ clauses, a trivial brute-force search solves MAX-SAT in time $O(mn2^n)$. We are interested in better algorithms running in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu > 0$; we will call μ the *savings* over exhaustive search, and we use $\tilde{O}()$ to ignore polynomial factors. To the best of our knowledge, the best savings is $\mu = \Omega(\frac{1}{c \log c})$ obtained by Dantsin and Wolpert [5] for an exponential-space algorithm. For polynomial space algorithms, the best savings is $\mu = \Omega(\frac{1}{c^2 \log^2 c})$ shown by Sakai, Seto, and Tamaki [15] recently.

The algorithm of Sakai, Seto, and Tamaki [15] is based on concentrated shrinkage under restrictions, which was used by Santhanam [16] for solving the satisfiability problem on de Morgan formulas. Santhanam [16] observed that, by

* This work is accepted to appear in SAT'15. Supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ ERC Grant Agreement no. 615075.

ISSN 1433-8092

greedily restricting the most frequent variables in a formula until p fraction of the variables are left, the formula size shrinks with high probability by a factor of p^Γ for $\Gamma \geqslant 1.5$. We will call Γ the *shrinkage exponent* for de Morgan formulas with respect to greedy restrictions. The satisfiability algorithm [16] recursively restricts $n - \Omega(n)$ variables, and then gets nontrivial savings since almost all restricted formulas have size much smaller than the number of variables left. Sakai, Seto, and Tamaki [15] showed that a similar shrinkage property holds for MAX-k-SAT instances, which leads to an algorithm with savings $\mu = \Omega(\frac{1}{c^2k^2})$ for instances with cn clauses. For solving MAX-SAT, they applied Schuler's width reduction [17, 1] to reduce MAX-SAT to MAX-k-SAT for $k = O(\log n)$; their final MAX-SAT algorithm [15] has savings $\mu = \Omega(\frac{1}{c^2 \log^2 c})$.

In this work, we improve the savings in [15] further to the following.

Theorem 1. *There is a polynomial-space algorithm solving MAX-SAT instances with n variables and cn clauses in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{c})$.*

Our algorithm is based on an improvement of concentrated shrinkage under greedy restrictions. We define a *measure* on MAX-SAT instances, which takes into account the numbers of clauses of different widths. We show that by a greedy restriction of all but p fraction of the variables, the measure shrinks with high probability by a factor of p^Γ for $\Gamma \geqslant 2$. This improved shrinkage exponent allows us to get better savings in the algorithm for the maximization problem. Furthermore, since the measure does not depend on the clause width, we do not need Schuler's width reduction which was used by [15], and our algorithm does not differentiate between MAX-SAT and MAX-k-SAT.

We further improve the savings when the algorithm is allowed to run in exponential space. Here we use Williams' algorithm [19] for MAX-2-SAT as a black-box, and improve the shrinkage exponent to $\Gamma \geqslant 3$ by defining a different measure on MAX-SAT instances. This improved shrinkage exponent again implies better savings in the algorithm.

Theorem 2. *There is an exponential-space algorithm solving MAX-SAT instances with n variables and cn clauses in time $\tilde{O}(2^{n(1-\mu)})$, for $\mu = \Omega(\frac{1}{\sqrt{c}})$.*

This improves the previous best-known result with savings $\mu = \Omega(\frac{1}{c \log c})$ by Dantsin and Wolpert [5] for solving MAX-SAT in exponential space.

Our approach is quite generic; we also apply it to solve sparse MAX-k-CSP. Specifically, we give a measure for MAX-k-CSP instances, and show that the measure shrinks nontrivially with probability 1 under greedy restrictions. This allows us to give the following algorithms.

Theorem 3. *For MAX-k-CSP instances with n variables and cn constraints, there is a polynomial-space algorithm running in time $\tilde{O}(2^{n(1-\mu)})$ with savings $\mu = \Omega(\frac{1}{ck^2})$, and an exponential-space algorithm with savings $\mu = \Omega(\frac{1}{\sqrt{ck^3}})$.*

All our algorithms extend to *counting* the number of optimal assignments for the *weighted* version of the problem, where each clause/constraint is given a weight, and the goal is to maximize the total weight of satisfied clauses/constraints.

We also consider depth-2 threshold circuits, which can be viewed as a generalization of MAX-SAT. Impagliazzo, Paturi, and Schneider [10] recently gave a satisfiability algorithm with savings $\mu = \frac{1}{c^{O(c^2)}}$ for depth-2 threshold circuits of cn wires. Using our shrinkage approach, we improve the savings slightly to $\frac{1}{c^{O(c)}}$, with a much simpler analysis.

1.1 Related work

Exact algorithms for sparse MAX-SAT and MAX-k-SAT have been well studied. For MAX-SAT instances with n variables and $m = cn$ clauses, the best savings of polynomial-space algorithms was $\Omega(\frac{1}{c^2 \log^2 c})$ [15] (and $\Omega(\frac{1}{c \log^3 c})$ for a randomized algorithm), improving a previous result $\Omega(\frac{1}{2^{O(c)}})$ [13]. The best savings of exponential-space algorithms was $\Omega(\frac{1}{c \log c})$ [5]. In this work, we improve the savings for both polynomial-space and exponential-space algorithms.

There are also algorithms with running time expressed as $\tilde{O}(2^{\delta m})$ for a constant $\delta < 1$, where m is the number of clauses/constraints. For example, the best such algorithms achieved $\delta \leqslant 0.4057$ for MAX-SAT [2], $\delta \leqslant 0.1583$ for MAX-2-SAT [7], and $\delta \leqslant 0.1901$ for MAX-2-CSP [7]. However, such algorithms are not better than exhaustive search for $m > n/\delta$.

For general (non-sparse) instances, the only non-trivial exact algorithm is Williams' algorithm for MAX-2-CSP (and MAX-2-SAT), which runs in time $\tilde{O}(2^{n\omega/3})$ for $\omega < 2.376$ and in exponential space. It is open whether we have non-trivial MAX-2-CSP algorithms running in polynomial space, or generalize Williams' algorithm for MAX-3-CSP or MAX-3-SAT. In this work, we will use Williams' algorithm as a blackbox for improving algorithms for sparse MAX-k-CSP and MAX-SAT.

The shrinkage approach to satisfiability algorithms was initiated by Santhanam [16] for de Morgan formulas. The algorithm was improved later [3, 4, 12] by improving the shrinkage exponents with respect to certain greedy restrictions. In particular, the improvement in [4] follows from a measuring technique of [9, 14] for de Morgan formulas.

The measuring and shrinkage technique we use in this work is also related to the "measure and conquer" approach [6], which was used to give improved exact algorithms for graph problems such as maximum independent set. The main difference is that, the usual "measure and conquer" approach reduces the measure additively in each recursive step, whereas the shrinkage approach reduces the measure by a multiplicative factor (depending on the shrinkage exponent), and moreover the reduction only occurs with high probability in the latter case.

1.2 Organization of the paper

We give preliminaries in Section 2. Since our MAX-SAT algorithms require more involved analysis than MAX-k-CSP, we first present our MAX-k-CSP algorithms in Section 3, and then MAX-SAT algorithms in Section 4. In Section 5, we apply a similar approach to improve satisfiability algorithms for depth-2 threshold circuits.

2 Preliminaries

2.1 MAXSAT, MAX-k-SAT, MAX-k-CSP

Let $x_1, \ldots, x_n$ be boolean variables. A *literal* is either a variable or its negation. A *clause* is a disjunction of literals; a *k-clause* is a clause on k literals. The *MAX-SAT* problem is to find, given a collection of clauses, an assignment to the variables maximizing the number of satisfied clauses (we call such an assignment *optimal*). MAX-k-SAT is the special case of MAX-SAT where all clauses have at most k literals. The *weighted MAX-SAT* problem generalizes MAX-SAT by associating with each clause an integer weight, and the goal is to find an assignment maximizing the total weight of satisfied clauses.

A *k-constraint* is a boolean function on k variables. The *(weighted) MAX-k-CSP* problem generalizes (weighted) MAX-k-SAT by allowing arbitrary constraints rather than disjunctions of literals. We will also consider the problems of counting the number of optimal assignments for the above optimization problems.

2.2 Concentration bounds

A sequence of random variables $X_0, X_1, \ldots, X_n$ is a *supermartingale* with respect to a 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 $1 \leqslant i \leqslant n$. We need the following variant of Azuma's inequality.

Lemma 1 ([3]). *Let $\{X_i\}_{i=0}^n$ be a supermartingale with respect to $\{R_i\}_{i=1}^n$. Let $Y_i = X_i - X_{i-1}$. If, for every $1 \leqslant i \leqslant n$, the random variable Y_i (conditioned on $R_{i-1}, \ldots, R_1$) assumes two values each with probability $1/2$, and there exists a constant $c_i \geqslant 0$ such that $Y_i \leqslant c_i$, then, for any λ, we have*

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

3 MAX-k-CSP

3.1 Known algorithms for MAX-2-CSP

Williams [19] gave an algorithm with constant savings solving general (non-sparse, weighted) MAX-2-CSP and MAX-2-SAT. In fact, Williams's algorithm also counts the number of optimal assignments.

Theorem 4 ([19, 11]). *For MAX-2-CSP instances with n variables and m constraints where each constraint has a weight at most W, there is an algorithm which counts the number of optimal assignments in time $O(\mu(nmW) \cdot 2^{n\omega/3})$, where $\omega < 2.376$ and $\mu(b) = b \log b \log\log b$.*

Note that, Williams' algorithm requires exponential space; it is not known whether there are polynomial-space algorithms with constant savings.

For sparse instances of MAX-2-CSP with cn constraints, several known algorithms [18, 8] have savings of the form $\mu = \Omega(\frac{1}{c})$. In the following, we show one simple algorithm with savings $\Omega(\frac{1}{c})$. The algorithm is based on a greedy restriction of the most frequent variables appearing in 2-constraints. Although the hidden constant in the savings $\Omega(\frac{1}{c})$ is not the best, we wish to use it as a warm-up for our later algorithms.

Lemma 2. *There is a polynomial-space algorithm solving MAX-2-CSP with n variables and cn 2-constraints in time $\tilde{O}(2^{n(1-\Omega(\frac{1}{c}))})$.*

Proof. Given a MAX-2-CSP instance with cn 2-constraints, we assume each 2-constraint is over two distinct variables. Let x be a variable which appears the maximum number of times in the 2-constraints. This means x appears in at least $2c$ of 2-constraints. We make two branches by fixing $x = 0$ in one branch and $x = 1$ in the other. In each branch, the number of remaining 2-constraints is at most $cn - 2c = cn(1 - 2/n) \leqslant cn(1 - 1/n)^2$. Then recursively restrict the most frequent variables one at a time. After $n - pn$ steps, for $p = 1/(2c)$, there will be pn variables left unfixed, but the number of remaining 2-constraints will be at most $cn(1-1/n)^2 \cdots (1-1/(pn+1))^2 = cn \cdot p^2 = pn/2$. Then after at most $pn/2$ more steps (restricting the most frequent variable in each step), all 2-constraints will be eliminated, and we get a MAX-1-CSP instance (on $pn/2$ variables).

We can maintain the number of satisfied constraints along each recursive branch, and, at the end of each branch, solve MAX-1-CSP by setting each variable to a value which satisfies at least as many constraints as the other. The total number of branches is at most $2^{n-pn/2}$. Therefore, the running time is $\tilde{O}(2^{n-pn/2}) = \tilde{O}(2^{n(1-\frac{1}{4c})})$, and the algorithm uses polynomial space. □

Note that, this algorithm can be extended to solve weighted MAX-2-CSP and also count the number of optimal assignments, by maintaining necessary information along the recursive branches. If W is the maximum weight of the constraints, the running time will be $\tilde{O}(2^{n(1-\frac{1}{4c})} \cdot \log W)$.

3.2 A polynomial-space algorithm for MAX-k-CSP

We first extend the algorithm in Lemma 2 to solve MAX-k-CSP. We introduce a *measure* on the instances, and use greedy restrictions such that the measure (and thus, the size of the instance) reduces non-trivially.

Let F be a MAX-k-CSP instance on n variables. For each i-constraint C in F, we define $\sigma(C) = \sigma_i \equiv i(i-1)$. Let $\sigma(F) = \sum_{C \in F} \sigma(C)$. Consider a restriction ρ where we randomly pick a variable and fix it. For an i-constraint C,

$$\mathbf{E}_\rho[\sigma(C|_\rho)] \leqslant \frac{n-i}{n}\sigma_i + \frac{i}{n}\sigma_{i-1} = \sigma_i \cdot \left[1 - \frac{i}{n}\left(1 - \frac{\sigma_{i-1}}{\sigma_i}\right)\right] = \sigma_i \cdot \left(1 - \frac{2}{n}\right).$$

We then have $\mathbf{E}_\rho[\sigma(F|_\rho)] \leqslant \sigma(F)(1 - 2/n) \leqslant \sigma(F)(1 - 1/n)^2$. By averaging, we can deterministically find one variable (in polynomial time) such that, after

fixing it to either 0 or 1, $\sigma(F)$ reduces by a factor of $(1-1/n)^2$. If we repeat this recursively until pn variables left, the measure on the restricted instance will be at most $\sigma(F)p^2$. Our algorithm follows from this non-trivial shrinkage.

Theorem 5. *For MAX-k-CSP instances on n variables with cn constraints, there is an algorithm running in $\tilde{O}(2^{n(1-\Omega(\frac{1}{ck^2}))})$ time and polynomial space.*

Proof. Let F be an instance with cn constraints; then $\sigma(F) \leqslant cn\sigma_k = cnk(k-1)$. The algorithm recursively restricts one variable at a time. At the i-th step, restrict a variable x such that the measure reduces by a factor of $(1-1/(n-i+1))^2$ for both restrictions $x=0$ and $x=1$. After $n-pn$ steps, for $p = \frac{1}{ck(k-1)}$, the restricted instance F' has $\sigma(F') \leqslant \sigma(F)p^2 \leqslant pn$. Note that, this holds for all recursive branches.

Suppose the number of i-constraints left in F' is b_i; then since $\sigma(F') = \sum_{i=2}^{k} b_i i(i-1) \leqslant pn$, we have $\sum_{i=2}^{k} b_i(i-1) \leqslant pn/2$. Therefore, after $pn/2$ recursive steps (fix all but one variable in each i-constraint), all remaining constraints have width 1, and we can solve MAX-1-CSP in polynomial time. The recursion tree has at most $2^{n-pn/2}$ branches. The total running time is at most $\mathrm{poly}(n) \cdot 2^{n(1-p/2)}$, and the algorithm uses polynomial space. □

3.3 An exponential-space algorithm for MAX-k-CSP

We next give an algorithm with improved running time but using exponential space. The algorithm reduces MAX-k-CSP to MAX-2-CSP via greedy restrictions as before, and then solves MAX-2-CSP using Williams' algorithm [19]. Using a different measure on the instances, we can improve the shrinkage exponent, and get better savings in the running time.

Let F be an instance with n variables and cn constraints. For each i-constraint C, we change the measure to $\sigma(C) = \sigma_i \equiv i(i-1)(i-2)$. Then $\sigma(F) \leqslant cn \cdot k(k-1)(k-2)$. Under a restriction ρ which randomly fixes one variable, we have, for $i \geqslant 3$,

$$\mathbf{E}[\sigma(C|_\rho)] \leqslant \sigma_i \cdot \left[1 - \frac{i}{n}\left(1 - \frac{\sigma_{i-1}}{\sigma_i}\right)\right] = \sigma_i \cdot \left(1 - \frac{3}{n}\right) \leqslant \sigma_i \cdot \left(1 - \frac{1}{n}\right)^3.$$

Then $\mathbf{E}[\sigma(F|_\rho)] \leqslant \sigma(F)\left(1-\frac{1}{n}\right)^3$. By averaging, we can deterministically find a variable such that $\sigma(F)$ shrinks by $\left(1-\frac{1}{n}\right)^3$.

Theorem 6. *For MAX-k-CSP instances with n variables and cn constraints, there is an algorithm running in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{\sqrt{ck^3}})$.*

Proof. We recursively restrict variables one by one. At the i-th step, restrict a variable x such that the measure reduces by $(1-1/(n-i+1))^3$. After $n-pn$ steps for $p = \sqrt{\frac{1}{ck(k-1)(k-2)}}$, let F' be the restricted instance; we have $\sigma(F') \leqslant \sigma(F) \cdot p^3 \leqslant pn$. Then we can further restrict $pn/6$ variables such that

all remaining constraints have width at most 2 (restrict all but two variables in each constraint). We get MAX-2-CSP instances for each branch and solve by Williams' algorithm in Theorem 4. The running time is at most

$$\mathrm{poly}(n) \cdot 2^{n-pn+pn/6} \cdot 2^{(pn-pn/6)\cdot 2.376/3} = \mathrm{poly}(n) \cdot 2^{n(1-\Omega(\frac{1}{\sqrt{ck^3}}))}.$$

□

Note that, the algorithm uses exponential space as required by Williams' algorithm. In general, this shrinkage approach gives a reduction from sparse instances of large width to dense instances of small width.

Theorem 7. *If, for some r, MAX-r-CSP is solvable in time $2^{n(1-\delta)}$, where δ is independent of the number of clauses, then, for all $k > r$, MAX-k-CSP instances with cn constraints are solvable in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{\delta}{c^{1/r}k^{1+1/r}})$.*

The proof is essentially the same as in Theorem 6, by changing $\sigma_i = i(i-1)\cdots(i-r)$. We omit the proof here.

We also note that, the algorithm in Theorem 6 can be generalized to count optimal assignments for even weighted instances. As required by Williams' algorithm in Theorem 4, for maximum constraint weight W, the running time increases by a factor of $\mathrm{poly}(W)$. This is in contrast with the polynomial-space algorithms in Lemma 2 and Theorem 5, where the running time increases by a factor of $O(\log W)$.

4 MAX-SAT and MAX-k-SAT

The algorithms for MAX-k-CSP also apply to the special case MAX-k-SAT. However, we can still improve the savings by eliminating the dependency on the clause width, and also generalize the algorithms to solve MAX-SAT. Here we still use the greedy restriction approach, but need a more involved analysis on the shrinkage of the instance size.

4.1 A polynomial-space algorithm for MAX-SAT

Let F be a MAX-SAT instance on n variables and cn clauses. We associate with each i-clause a measure σ_i. Let C be an i-clause, for $i \geqslant 2$. Let ρ be a restriction which randomly picks and fixes one variable. Then C becomes an $(i-1)$-clause or a constant each with probability $i/2n$. Thus,

$$\mathbf{E}[\sigma(C|_\rho)] \leqslant \frac{n-i}{n}\sigma_i + \frac{i}{2n}\sigma_{i-1} = \sigma_i \cdot \left[1 - \frac{i}{n}\left(1 - \frac{\sigma_{i-1}}{2\sigma_i}\right)\right].$$

We can choose

$$\sigma_1 = 0,\ \sigma_2 = 1,\ \text{and } \sigma_i = 2,\ i \geqslant 3.$$

It is easy to check that, for all $i \geqslant 2$, $\mathbf{E}[\sigma(C|_\rho)] \leqslant \sigma_i(1-2/n) \leqslant \sigma_i(1-1/n)^2$. Then we have $\mathbf{E}[\sigma(F|_\rho)] \leqslant \sigma(F)(1-1/n)^2$. By averaging, we can deterministically find

a variable x such that $[\sigma(F|_{x=1}) + \sigma(F|_{x=0})]/2 \leqslant \sigma(F)(1-1/n)^2$. Note that, this only bounds the average of $\sigma(F|_{x=1})$ and $\sigma(F|_{x=0})$.

The MAX-SAT algorithm then follows by restricting variables recursively. Although we only have shrinkage on average in each step, we can argue that, shrinkage happens with high probability over the whole process, using a similar approach as in [16, 3].

Theorem 8 (Theorem 1 restated). *There is a polynomial-space algorithm solving MAX-SAT instances with n variables and cn clauses in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{c})$.*

Proof. Let F be an instance with n variables and cn clauses, and thus $\sigma(F) \leqslant 2cn$. The algorithm recursively restricts one variable at a time.

Let $F_0 := F$, and F_i be the restricted instance after the i-th step. At the i-th step, we find a variable x in F_{i-1} such that, by randomly fixing x to 0 or 1, $\mathbf{E}[\sigma(F_i)] \leqslant \sigma(F_{i-1})(1 - \frac{1}{n-i+1})^2$.

Define

$$Z_i = \log \sigma(F_i) - \log \sigma(F_{i-1}) - 2 \log \left(1 - \frac{1}{n-i+1}\right).$$

By Jensen's inequality, conditioned on the random bits assigned to the first $i-1$ variables, $\mathbf{E}[Z_i] \leqslant 0$. We also have $Z_i \leqslant c_i := -2 \log \left(1 - \frac{1}{n-i+1}\right)$, since $\sigma(F_i) \leqslant \sigma(F_{i-1})$.

Then $\{\sum_{j=1}^{i} Z_j\}$ is a supermartingale with respect to the random bits assigned to restricted variables. By Lemma 1, for any λ,

$$\mathbf{Pr}\left[\sum_{j=1}^{i} Z_j \geqslant \lambda\right] \leqslant \exp\left(-\frac{\lambda^2}{2\sum_{j=1}^{i} c_j^2}\right).$$

The left-hand side is

$$\mathbf{Pr}\left[\log \sigma(F_i) - \log \sigma(F) - 2 \log\left(\frac{n-i}{n}\right) \geqslant \lambda\right] = \mathbf{Pr}\left[\sigma(F_i) \geqslant e^{\lambda} \sigma(F) \left(\frac{n-i}{n}\right)^2\right].$$

For each $1 \leqslant j \leqslant i$, by $\log(1+x) \leqslant x$, we have $c_j = 2\log(1 + \frac{1}{n-j}) \leqslant \frac{2}{n-j}$. Thus, $\sum_{j=1}^{i} c_j^2 \leqslant \frac{4}{n-i-1}$ since

$$\sum_{j=1}^{i} \left(\frac{1}{n-j}\right)^2 \leqslant \sum_{j=1}^{i} \left(\frac{1}{n-j-1} - \frac{1}{n-j}\right) \leqslant \frac{1}{n-i-1}.$$

For $i = n - pn$, $\lambda = \ln 2$, and $pn > 20$, we get

$$\mathbf{Pr}\left[\sigma(F_i) \geqslant 4cnp^2\right] \leqslant \mathbf{Pr}\left[\sigma(F_i) \geqslant 2\sigma(F)p^2\right] \leqslant e^{-\lambda^2(pn-1)/8} < 2^{-pn/20}.$$

Choose $p = \delta/4c$, for δ to be fixed later. We have that, after restricting $n - pn$ variables, with probability at least $1 - 2^{-pn/20}$, there are at most $4cnp^2 = \delta pn$ remaining clauses of width at least 2.

Claim. There is a polynomial-space algorithm running in time $\tilde{O}(2^{n/2})$ which solves MAX-SAT for instances with n variables and δn clauses of width at least 2, for $\delta \leqslant 0.1$.

Proof. We recursively restrict arbitrary variables that appear in clauses of width at least 2, and stop when all remaining clauses have width 1; then solve MAX-1-SAT easily. Let $m = \delta n$ be the number of clauses of width at least 2. The recursion tree has size bounded by the recurrence $T(n, m) \leqslant T(n-1, m) + T(n-1, m-1)$. This is at most $\binom{n}{\delta n} \leqslant \left(\frac{e}{\delta}\right)^{\delta n} < 2^{n/2}$, where the first inequality follows from $\binom{n}{k} \leqslant \left(\frac{ne}{k}\right)^k$.

□

Choose $\delta = 0.1$. By the above claim, the running time for branches left with at most δpn clauses of width at least 2 is $2^{n-pn} \cdot 2^{pn/2} = 2^{n-pn/2}$. For the other branches, we use brute-force search; the running time is at most $2^n \cdot 2^{-pn/20}$. Therefore, the total running time is bounded by $\mathrm{poly}(n) \cdot 2^{n(1-\Omega(1/c))}$. □

4.2 An exponential-space algorithm for MAX-SAT

To improve the savings in the running time, we can improve the shrinkage exponent by reducing the instances to MAX-2-SAT, and apply Williams' algorithm [19].

We let $\sigma_1 = \sigma_2 = 0,\ \sigma_3 = 1,\ \sigma_4 = 2$, and $\sigma_i = 3$, for $i \geqslant 5$. It is easy to see that, under one step of random restriction ρ, for an i-clause C where $i \geqslant 3$,

$$\sigma(C|_\rho) \leqslant \sigma_i \cdot \left[1 - \frac{i}{n}\left(1 - \frac{\sigma_{i-1}}{2\sigma_i}\right)\right] \leqslant \sigma_i \cdot \left(1 - \frac{3}{n}\right).$$

Thus, for an instance F,

$$\sigma(F|_\rho) \leqslant \sigma(F) \cdot \left(1 - \frac{3}{n}\right) \leqslant \sigma(F) \cdot \left(1 - \frac{1}{n}\right)^3.$$

Theorem 9 (Theorem 2 restated). *For MAX-SAT instances on n variables with cn constraints, there is an algorithm running in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{\sqrt{c}})$.*

The proof is similar to the proof of Theorem 1. We can greedily restrict $n - pn$ variables. Then with high probability (at least $1 - 2^{-pn/20}$), there are at most $4cnp^3$ clauses of width at least 3; we solve such restricted instances following the claim below for $p = \sqrt{\delta/4c}$ and $\delta = 0.01$. Otherwise, we use brute-force search. The total running time is bounded by $\tilde{O}(2^{n(1-\Omega(p))})$. We omit the complete proof.

Claim. MAX-SAT instances on n variables and δn clauses of width larger than 2, for $\delta \leqslant 0.01$, are solvable in time $\tilde{O}(2^{0.9n})$ and exponential space.

Proof. We recursively restrict variables appearing in clauses of width larger than 2; when all clauses have width at most 2, we use Williams' algorithm in Theorem 4 for MAX-2-SAT. The total running time is bounded by $\binom{n}{\delta n} \cdot 2^{2.376n/3} \leqslant \left(\frac{e}{\delta}\right)^{\delta n} \cdot 2^{2.376n/3} < 2^{0.9n}$.

□

Again, we have the following reduction from sparse instances of large width to dense instances of small width.

Theorem 10. *If for some r, MAX-r-SAT is solvable in time $2^{n(1-\delta)}$, where δ is independent of the number of clauses, then MAX-SAT for instances with cn clauses is solvable in time $\tilde{O}(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{\delta}{c^{1/r}})$.*

Sakai et al. [15] uses Schuler's width reduction to reduce MAX-SAT instances with cn clauses to MAX-k-SAT instances for $k = O(\log c)$, and then solve MAX-k-SAT in time $O(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{c^2k^2})$; their final algorithm [15] for MAX-SAT runs in time $O(2^{n(1-\mu)})$ for $\mu = \Omega(\frac{1}{c^2 \log^2 c})$. Our result avoids Schuler's width reduction and improves the running time.

Our MAX-SAT algorithms can be extended to solve the weighted version and also the counting problem. For instances with clause weight at most W, the running time of the polynomial-space algorithm (Theorem 1) increases by a factor of $\log(W)$, and the running time of the exponential-space algorithm (Theorem 2) increases by a factor of $\text{poly}(W)$.

5 Sparse depth-2 threshold circuits

A *threshold circuit* is a boolean circuit where all internal gates are threshold gates; a *threshold gate* on k inputs computes a function $\text{sign}(\sum_{i=1}^{k} w_i x_i + w_0)$ where w_i's are integer weights. A depth-2 threshold circuit on n inputs has one output threshold gate at the top, a layer of threshold gates in the middle, and n input gates at the bottom. Obviously, a MAX-SAT instance with m clauses is a special case of depth-2 threshold circuits with $m + 1$ threshold gates.

Impagliazzo et al. [10] showed a nontrivial satisfiability algorithm for depth-2 threshold circuits with linear number of wires (a *wire* is an edge in the underlying graph of the circuit). For depth-2 threshold circuits with cn wires, the algorithm runs in time $\tilde{O}(2^{(1-\mu)n})$ for $\mu = \frac{1}{c^{O(c^2)}}$. They first give an algorithm for the special case where there are few threshold gates, and then applied restrictions to eliminate threshold gates non-trivially. Using our shrinkage approach, we can improve the parameters in gate elimination, which implies better savings of the algorithm.

Lemma 3. *Given a depth-2 threshold circuits with n variables and cn wires, there is a set U of at least pn variables for $p = \frac{1}{c^{O(c)}}$ such that the number of threshold gates depending on at least two variables in U is at most δpn, for any constant $\delta < 1$. Furthermore, U can be constructed in polynomial time.*

This lemma follows directly from the following claim. Impagliazzo et al. [10] showed this result for $p = \frac{1}{c^{O(c^2)}}$ using a more dedicated analysis.

Claim. Let $\mathcal{S}$ be a collection of subsets of $[n]$ such that $\sum_{A\in\mathcal{S}} |A| \leqslant cn$. Then there is a subset $U \subseteq [n]$ of size at least pn for $p = \frac{1}{c^{O(c)}}$ such that there are at most δpn subsets in $\mathcal{S}$ containing at least two elements in U, for any constant $\delta < 1$. Furthermore, U can be constructed in polynomial time.

Proof. Define $\sigma(\mathcal{S}) = \sum_{A\in\mathcal{S}} |A| - |\mathcal{S}|$. At the beginning, let $U = [n]$ and obtain $\mathcal{T}$ from $\mathcal{S}$ by eliminating singletons; note that $\sigma(\mathcal{T}) = \sigma(\mathcal{S})$. We will greedily eliminate elements from U and subsets in $\mathcal{T}$ such that $\sigma(\mathcal{T})$ reduces non-trivially. We maintain $\mathcal{T}$ as the collection of subsets each containing at least two elements from U.

At each step, check whether $|\mathcal{T}| \leqslant \delta|U|$. If so, then we are done and return U.

Otherwise, it holds that $|\mathcal{T}| > \delta|U|$; we will greedily eliminate one element and continue the process. Suppose that $\sum_{A\in\mathcal{T}} |A| = c'|U|$ for $c' \leqslant c$. Then we can easily find some $x \in U$ which appears in at least c' subsets in $\mathcal{T}$. Eliminate x from U and from all subsets in $\mathcal{T}$; we also remove any singletons from $\mathcal{T}$. Let $U' = U \setminus \{x\}$, and denote by $\mathcal{T}'$ the new collection. Then

$$\sum_{A\in\mathcal{T}'} |A| \leqslant c'|U| - c' = c'(|U| - 1) \leqslant c|U'|,$$

and since $\sigma(\mathcal{T}) = c'|U| - |\mathcal{T}| < (c' - \delta)|U|$,

$$\sigma(\mathcal{T}') \leqslant \sigma(\mathcal{T}) - c' < \sigma(\mathcal{T}) \left(1 - \frac{c'}{(c' - \delta)|U|}\right) \leqslant \sigma(\mathcal{T}) \left(1 - \frac{1}{|U|}\right)^{c/(c-\delta)}.$$

Fix $p' = (2(c - \delta))^{-(c-\delta)/\delta} = c^{-O(c)}$. If $|\mathcal{T}| \leqslant \delta|U|$ holds at the i-th step for some $i \leqslant n - p'n$, then we have returned U at the i-th step; the claim holds for $p = |U|/n = c^{-O(c)}$.

If the process continues for $n - p'n$ steps ($|\mathcal{T}| > \delta|U|$ holds at each step), then the collection $\mathcal{T}$ after $n - p'n$ steps has $\sigma(\mathcal{T}) \leqslant (cn - \delta n)p'^{c/(c-\delta)} \leqslant p'n/2$. Note that we still have $p'n$ elements in U. Then, for each $A \in \mathcal{T}$, eliminate all but one arbitrary element of A from U and all subsets in $\mathcal{T}$; we also remove any singletons from $\mathcal{T}$. Since $\sigma(\mathcal{T}) \leqslant p'n/2$, we can eliminate at most $p'n/2$ elements in total. Finally, there are at least $p'n/2$ elements left in U, but $\mathcal{T}$ becomes empty; that is $|\mathcal{T}| = 0 \leqslant \delta|U|$. We return U; the claim holds for $p \geqslant p'/2 = c^{-O(c)}$.

□

We need the following algorithm [10] for the special case where there are few threshold gates.

Lemma 4 ([10]). *For depth-2 threshold circuits with n variables and δn threshold gates for $\delta < 0.099$, there is a satisfiability algorithm running in time $\tilde{O}(2^{(1-\mu)n})$ for a constant $\mu > 0$.*

We then get the following improved algorithm by combining greedy restrictions in Lemma 3 with the algorithm of Lemma 4.

Theorem 11. *There is a satisfiability algorithm for depth-2 circuits with n variables and cn wires running in time $\tilde{O}(2^{n(1-\mu')})$ for $\mu' = \frac{1}{c^{O(c)}}$.*

Proof. Given a depth-2 threshold circuit with n variables and cn wires, we fix $\delta < 0.099$, and find a subset U of pn variables for $p = \frac{1}{c^{O(c)}}$ as in Lemma 3. We restrict all variables not in U. By enumerating assignments to variables not in U, we get 2^{n-pn} branches. Each branch gives a restricted circuit on pn variables and δpn threshold gates; then apply the algorithm of Lemma 4 for each branch, which takes time $\tilde{O}(2^{(1-\mu)pn})$ for a constant μ.

The total running time is bounded by $\tilde{O}(2^{n-pn} \cdot 2^{(1-\mu)pn}) = \tilde{O}(2^{n-n/c^{O(c)}})$. □

6 Open questions

A major open question is to improve the savings to $\Omega(1/\operatorname{polylog}(c))$ for solving MAX-SAT/MAX-k-SAT on instances of cn clauses. For depth-2 threshold circuits with cn wires, it would be interesting to improve the savings to $\Omega(1/\operatorname{poly}(c))$, or give an algorithm for circuits with cn gates, instead of wires. It is challenging to get constant savings for solving non-sparse MAX-k-SAT, for any $k \geqslant 3$.

References

1. C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009*, pages 75–85, 2009.
2. J. Chen and I. Kanj. Improved exact algorithms for max-sat. *Discrete Applied Mathematics*, 142(1-3):17–27, 2004.
3. R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel, and D. Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity*, CCC '14, 2014.
4. R. Chen, V. Kabanets, and N. Saurabh. An improved deterministic #sat algorithm for small de morgan formulas. In *Proceedings of Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Part II*, pages 165–176, 2014.
5. E. Dantsin and A. Wolpert. Max-sat for formulas with constant clause density can be solved faster than in o(s) time. In *SAT*, pages 266–276, 2006.
6. F. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, August 2009.
7. S. Gaspers and G. Sorkin. A universally fastest algorithm for max 2-sat, max 2-csp, and everything in between. *J. Comput. Syst. Sci.*, 78(1):305–335, 2012.
8. A. Golovnev and K. Kutzkov. New exact algorithms for the 2-constraint satisfaction problem. *Theor. Comput. Sci.*, 526:18–27, 2014.

9. R. Impagliazzo and N. Nisan. The effect of random restrictions on formula size. *Random Structures and Algorithms*, 4(2):121–134, 1993.
10. R. Impagliazzo, R. Paturi, and S. Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 479–488, 2013.
11. M. Koivisto. Optimal 2-constraint satisfaction via sum-product algorithms. *Inf. Process. Lett.*, 98(1):24–28, 2006.
12. 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.
13. A. Kulikov and K. Kutzkov. New bounds for max-sat by clause learning. In *Computer Science - Theory and Applications, Second International Symposium on Computer Science in Russia, CSR 2007, Ekaterinburg, Russia, September 3-7, 2007, Proceedings*, pages 194–204, 2007.
14. M. Paterson and U. Zwick. Shrinkage of de Morgan formulae under restriction. *Random Structures and Algorithms*, 4(2):135–150, 1993.
15. T. Sakai, K. Seto, and S. Tamaki. Solving sparse instances of max sat via width reduction and greedy restriction. In *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 32–47, 2014.
16. 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.
17. R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, 54(1):40–44, 2005.
18. A. Scott and G. Sorkin. Linear-programming design and analysis of fast algorithms for max 2-csp. *Discret. Optim.*, 4(3-4):260–287, December 2007.
19. R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de