Improved MaxSAT Algorithms for Instances of Degree 3

Chao Xu[1], Jianer Chen[1,2], and Jianxin Wang[1(✉)]

[1] School of Information Science and Engineering,
Central South University, Changsha, People's Republic of China
jxwang@mail.csu.edu.cn
[2] Department of Computer Science and Engineering,
Texas A&M University, College Station, USA

Abstract. The degree of a variable x_i in a MaxSAT instance is the number of times x_i and $\bar{x}_i$ appearing in the given formula. The degree of a MaxSAT instance is equal to the largest variable degree in the instance. In this paper, we study techniques for solving the MaxSAT problem on instances of degree 3 (briefly, $(n, 3)$-MaxSAT), which is NP-hard. Two new non-trivial reduction rules are introduced based on the resolution principle. As applications, we present two algorithms for the $(n, 3)$-MaxSAT problem: a parameterized algorithm of time $O^*(1.194^k)$, and an exact algorithm of time $O^*(1.237^n)$, improving the previous best upper bounds $O^*(1.2721^k)$ and $O^*(1.2600^n)$, respectively.

1 Introduction

The Maximum Satisfiability problem (MaxSAT) plays a key role in the study of computational optimization [9]. The *Strong Exponential Time Hypothesis* [10] conjectures that MaxSAT cannot be solved in time $O^*(2^{cn})$ for any constant $c < 1$, where n is the number of variables in the input instance, which is a CNF formula. Algorithms for MaxSAT and various restricted versions of MaxSAT have been studied extensively (see, for example, [6] and its references). Define the *degree* of a variable x_i in a MaxSAT instance to be the number of times x_i and $\bar{x}_i$ appearing in the formula. The (s, t)-MaxSAT problem is a well-known restricted version of the MaxSAT problem in which each clause in an instance contains at most s literals and each variable has degree bounded by t [12,14]. It is shown that $(n, 2)$-MaxSAT problem can be solved in polynomial time [7].

This paper focuses on algorithms for the $(n, 3)$-MaxSAT problem. Since the $(2, 3)$-MaxSAT problem is NP-hard [12], the $(n, 3)$-MaxSAT problem is NP-hard for $n \geq 2$. Exact and parameterized algorithms (e.g., [2,3,12,13]) have been extensively studied for the problem. Two main parameters used for evaluating the performance of the algorithms have been used: the number n of variables and the number k of satisfied clauses.

This work is supported by the National Natural Science Foundation of China, under grants 61173051, 61232001, 61472449, and 61420106009.

© Springer International Publishing Switzerland 2015
Z. Lu et al. (Eds.): COCOA 2015, LNCS 9486, pp. 20–30, 2015.
DOI: 10.1007/978-3-319-26626-8_2

The main results of the current paper are two new non-trivial reduction rules (R-Rules 7–8), which are based on the resolution principle. As applications, we propose two algorithms for the $(n, 3)$-MaxSAT problem in terms of the two main parameters.

We first give formal definitions of the problems we are focused on. *The* $(n, 3)$-MaxSAT *problem* asks for an assignment satisfying the maximum number of clauses in a given formula in which each variable has degree bounded by 3. The (*parameterized*) $(n, 3)$-MaxSAT *problem* consists of instances of the form (F, k), where F is a formula of the $(n, 3)$-MaxSAT and k is an integer, asking whether there is an assignment to the variables that satisfies at least k clauses in F.

The table in Fig. 1 lists the current literature on algorithms for the $(n, 3)$-MaxSAT problem. For comparison, we also include our result in the current paper in the table.[1]

Bound(n)	Bound(k)	Reference	Year
$O^*(1.732^n)$		Raman *et al* [14]	1998
$O^*(1.3248^n)$		Bansal, Raman [1]	1999
	$O^*(1.3247^k)$	Chen, Kanj [5]	2002
$O^*(1.27203^n)$		Kulikov [15]	2005
	$O^*(1.2721^k)$	Bliznets, Golovnev [3]	2012
$O^*(1.2600^n)$		Bliznets [4]	2013
$O^*(1.237^n)$	$O^*(1.194^k)$	this paper	2015

Fig. 1. Progress in $(n, 3)$-MaxSAT algorithms

Most algorithms for MaxSAT (as well as for $(n, 3)$-MaxSAT) are based on the branch-and-bound technique [8]. The *Strong Exponential Time Hypothesis* [10] conjectures, to some extent, a popular opinion that branch-and-bound is perhaps unavoidable to solve the MaxSAT problem and its variations. Therefore, how to branch more efficiently in algorithms solving $(n, 3)$-MaxSAT becomes crucial.

A contribution of the current paper is to show that the resolution principle [7] can be applied to solve the $(n, 3)$-MaxSAT problem, while keeping all variables of degree 3. It has been well-known that the resolution principle is a very powerful tool to solve the satisfiability problem [7]. In particular, variable resolutions in a CNF formula preserve the satisfiability of the formula. Unfortunately, variable resolutions cannot be used directly to solve the $(n, 3)$-MaxSAT problem in general case, since not all clauses are presumed to be satisfied by an optimal assignment to an instance of the $(n, 3)$-MaxSAT problem.

We begin with some preliminary definitions.

[1] Following the current convention in the research in exact and parameterized algorithms, we will use the notation $O^*(f)$ to denote the bound $f \cdot m^{O(1)}$, where f is an arbitrary function and m is the instance size.

A (Boolean) *variable* x can be assigned value either 1 (TRUE) or 0 (FALSE). A variable x has two corresponding literals: the *positive literal* x and the *negative literal* $\bar{x}$, which will be called the *literals* of x. A *clause* C is a disjunction of a set of literals, also regarded as a set of the literals. So, $C_1 = zC_2$ indicates that the clause C_1 is in consist of the literal z plus all literals in the clause C_2, and use C_1C_2 to denote the clause that consists of all literals that are in either C_1 or C_2, or both. Without loss of generality, we assume that a literal can appear in a clause at most once. A clause C is *satisfied* by an assignment if under the assignment, at least one literal in C gets a value 1. A (CNF Boolean) *formula* F is a conjunction of clauses $C_1, \ldots, C_m$, regarded as a collection of the clauses. The formula F is *satisfied* by an assignment to the variables in the formula if all clauses in F are satisfied by the assignment.

The *size* of a clause C is the number of literals in C. A clause is an *h-clause* if its size is h, and an *h^+-clause* if its size is at least h. A clause is *unit* if its size is 1 and is *non-unit* if its size is larger than 1. The *size* of a CNF formula F is equal to the sum of the sizes of the clauses in F.

A literal z is an *(i, j)-literal* in a formula F if z and $\bar{z}$ appear i times and j times in F, respectively. Thus, a variable x is of degree h if x is an (i, j)-literal such that $i + j = h$. An $(i, 1)$-literal z is an *$(i, 1)$-singleton* if $\bar{z}$ occurs in a unit clause $(\bar{z})$. When i is not critical, we also call an $(i, 1)$-singleton simply a *singleton*. A variable of degree h (resp. at least h) is also called an *h-variable* (resp. *h^+-variable*).

A *resolvent* on a variable x in a formula F is a clause of a new form CD such that xC and $\bar{x}D$ are clauses in F. The *resolution* on the variable x in F, written as $DP_x(F)$, is a formula that is obtained by first removing all clauses that contain either x or $\bar{x}$ from F and then adding all resolvents on x into F.

2 Reduction Rules

A *reduction rule* converts, in polynomial time, an instance (F, k) of $(n, 3)$-MAXSAT into another instance (F', k') with $k \geq k'$ such that (F, k) is a Yes-instance if and only if (F', k') is a Yes-instance. Note that a reduction rule can be acknowledged as a special case of branching steps.

We present 9 reduction rules, R-Rules 1–9. The reduction rules are supposed to be applied *in order*, i.e., R-Rule j cannot be applied until none of R-Rules i with $i < j$ is applicable. In the following, F is always supposed to be a conjunction of clauses.

The first three reduction rules are from [4].

R-Rule 1 ([4]). $(F \wedge (x\bar{x}C), k) \to (F, k-1)$, and $(F \wedge (x) \wedge (\bar{x}), k) \to (F, k-1)$.

R-Rule 2 ([4]). If there is an (i, j)-literal z in the CNF formula F, with at least j unit clauses (z), then $(F, k) \to (F_{z=1}, k-i)$, where $F_{z=1}$ is the formula F with an assignment $z = 1$ on the literal z.

Assume that R-Rule 2 is not applicable to F, and then each literal in F has its negation also in F. Thus, all variables are 2^+-variables. Under this condition, we

can process 2-variables based on the resolution principle [7], whose correctness can be easily verified.

R-Rule 3 ([4]). For any 2-variable x, $(F \wedge (xC_1) \wedge (\bar{x}C_2), k) \rightarrow (F \wedge (C_1C_2), k-1)$.

Note that each variable appears at most 3 times in F. In case none of R-Rules 1–3 is applicable, every variable is a 3-variable. Moreover, for each $(2,1)$-literal z, there is no unit clause (z). Now we describe two reduction rules based on variations of the resolution principle, which are from [2].

R-Rule 4 ([2]). For a $(2,1)$-literal x in a 2-clause (xy) and the clause with $\bar{x}$ contains at least two literals, $(F_1 = F \wedge (xy) \wedge (xC_1) \wedge (\bar{x}D), k) \rightarrow (F_2 = F \wedge (yD) \wedge (\bar{y}C_1D), k-1)$.

After R-Rule 4, the degree of all variables in yD becomes 4. In order to keep all variables being 3-variables, a branching for variable y must be applied. Naturally, the branching vector is (3+1,1+1), whose root is 1.2721. However, since we only need consider $|D| \geq 1$, we can do better.

R-Rule 5 ([2]). If two variables x and y of degree 3 appear together in 3 clauses, then all these 3 clauses can be satisfied by assigning x and y properly.

Next rule is just a part from Corollary 1 in [2].

R-Rule 6 ([2]). If there are two clauses xyC_1 and $\bar{x}\bar{y}C_2$ in the formula F such that each variable appears in three times, then we can safely do resolution on x, such that F is replaced with $DP_x(F)$.

After R-Rule 6, R-Rule 1 must be followed, keeping that each variable appears three times in the obtained formula. The next 2 rules are based on resolution, but its transformation is non-trivial, i.e. there is no direct relation to other MaxSAT algorithms. To be convenient, let $maxsat(F)$ be the maximum number of clauses satisfied in F. Note that since R-Rule 5 is not applicable, any two 3-variables appear in at least 4 clauses.

R-Rule 7. For a CNF formula $F_1 = F \wedge (xyC_1) \wedge (x\bar{y}C_2) \wedge (\bar{x}D_1) \wedge (\bar{y}D_2)$, where x is a $(2,1)$-literal in F_1, $(F_1 = F \wedge (xyC_1) \wedge (x\bar{y}C_2) \wedge (\bar{x}D_1) \wedge (\bar{y}D_2), k) \rightarrow (F_2 = F \wedge (x'C_2) \wedge (x'D_2) \wedge (\bar{x}'C_1D_1), k-1)$.

Lemma 1. *R-Rule 7 converts the instance (F_1, k) of $(n,3)$-MaxSAT into an instance $(F_2, k-1)$ such that $(F_1 = F \wedge (xyC_1) \wedge (x\bar{y}C_2) \wedge (\bar{x}D_1) \wedge (\bar{y}D_2), k)$ is a Yes-instance if and only if $(F_2 = F \wedge (x'C_2) \wedge (x'D_2) \wedge (\bar{x}'C_1D_1), k-1)$ is a Yes-instance.*

Proof. For an optimal assignment satisfying $C_1 = 1$ or $D_1 = 1$ or $C_2 = D_2 = 1$, by reassigning properly to variables x, y, x', $maxsat(F_1) = maxsat(F) + 4$ and $maxsat(F_2) = maxsat(F) + 3 = maxsat(F_1) - 1$. For an optimal assignment satisfying $C_1 = D_1 = 0$ and $C_2 = 0$ (resp. $D_2 = 0$), $maxsat(F_1) = maxsat(F)+3$ and $maxsat(F_2) = maxsat(F) + 2 = maxsat(F_1) - 1$. Thus, for both cases $maxsat(F_2) = maxsat(F_1) - 1$. □

Now, we introduce the following new reduction rule.

R-Rule 8. For any two $(2,1)$-literals xy both in two clauses (xyC_1) and (xyC_2), $(F_1 = F \wedge (xyC_1) \wedge (xyC_2) \wedge (\bar{x}D_1) \wedge (\bar{y}D_2), k) \rightarrow (F_2 = F \wedge (x'C_1) \wedge (x'C_2) \wedge (\bar{x}'D_1D_2), k-1)$.

Lemma 2. *R-Rule* 8 *converts the instance* (F_1, k) *of* $(n,3)$-MaxSAT *into an instance* $(F_2, k-1)$ *such that* $(F_1 = F \wedge (xyC_1) \wedge (xyC_2) \wedge (\bar{x}D_1) \wedge (\bar{y}D_2), k)$ *is a Yes-instance if and only if* $(F_2 = F \wedge (x'C_1) \wedge (x'C_2) \wedge (\bar{x}'D_1D_2), k-1)$ *is a Yes-instance.*

Proof. Similar to the proof in Lemma 2, for an optimal assignment satisfying $D_1 = 1$ or $D_2 = 1$ or $C_1 = C_2 = 1$, by reassigning properly to variables x, y, x', $maxsat(F_1) = maxsat(F) + 4$ and $maxsat(F_2) = maxsat(F) + 3 = maxsat(F_1) - 1$. For an optimal assignment satisfying $D_1 = D_2 = 0$ and $C_1 = 0$ (resp. $C_2 = 0$), at least one clause is not satisfied, and $maxsat(F_1) = maxsat(F) + 3$ and $maxsat(F_2) = maxsat(F) + 2 = maxsat(F_1) - 1$. For both cases $maxsat(F_2) = maxsat(F_1) - 1$. □

After R-Rules 1–3, 5–8, for any 2-literal x, the two clauses with x, each of which has at least two literals, and each pair of distinct clauses share at most one common variables. According to this property, similar to R-Rule 5, we can prove that for each quadruple of 3-variables must appear in at least 6 clauses.

Lemma 3. *If any quadruple variables of degree* 3 *appear together in at most* 5 *clauses, then all these clauses can be satisfied by assigning properly.*

Proof. Suppose variables x_1, x_2, x_3 and x_4 of degree 3 are in 5 clauses. Note that $3 \times 4 = 12$ literals of x_1, x_2, x_3, x_4 must be in these clauses. We first choose a 2-literal, i.e. the literal occurring in two clauses, denoted by x_1. Let $x_1 = 1$, and two clauses x_1C_1, x_1C_2 are satisfied. Since R-Rules 5–8 are not applicable, for any literal x_i, no two clauses containing literal x_i share another common variables. Thus, there are at most 3 literals (of variables from x_2, x_3 and x_4) satisfied by $x_1 = 1$ in C_1C_2. In all, at most 6 literals (of variables from x_1, x_2, x_3 and x_4) are reduced by at this time. Then there are two cases to consider. One is that there is still a 2-literal in the remaining of 3 variables, denoted by x_2. Let $x_2 = 1$, and at most 2 extra literals (of variables x_3 and x_4) are satisfied. In all, there are $6 + 2 + 3 = 11$ literals (of variables from x_1, x_2, x_3 and x_4) satisfied. Thus, there are at least one literal not satisfied and denoted by x_3. Let $x_3 = 1$, and all clauses are satisfied. The other case is that each literal (of variables from x_2, x_3 and x_4) appearing in the remaining of three clauses is a 1-literal. Since no two clauses share two common variables, there is at most one unit clause in the remaining of the 3 clauses, and can be simultaneously satisfied by assigning x_2, x_3 and x_4 properly. □

Next, we introduce another reduction rule, which is implemented in practical MaxSAT solver MaxSatz [1].

R-Rule 9 ([1]). $(F_1 = F \wedge (xy) \wedge (\bar{x}) \wedge (\bar{y}), k) \rightarrow (F_2 = F \wedge (\bar{x}\bar{y}), k-1)$.

3 An $O^*(1.194^k)$-time Parameterized Algorithm

Branching on an instance (F, k) of $(n, 3)$-MaxSAT leads to a collection $\{(F_1, k - d_1), \ldots, (F_r, k - d_r)\}$ of instances of $(n, 3)$-MaxSAT, such that (F, k) is a Yes-instance if and only if at least one of $(F_1, k - d_1)$, ..., $(F_r, k - d_r)$ is a Yes-instance. Such a branching step is called a $(d_1, \ldots, d_r)$-*branching*, and the vector $t = (d_1, \ldots, d_r)$ is called the *branching vector* for the branching. Each branching vector corresponds to a polynomial (see, e.g. [5]), and has a unique positive root that is larger than or equal to 1. Let the root $\rho(t)$ of a branching be the root of the branching vector t.

Let t_1 and t_2 be two branching vectors. We say that the t_1-branching is *inferior* to the t_2-branching if $\rho(t_1) > \rho(t_2)$. Based on the branch-and-bound technique, if every branching step in the algorithm has its root bounded by a constant $c \geq 1$, then the running time of the algorithm is bounded by $O^*(c^k)$.

We called formula F *reduced* if none of R-Rules 1–3, 5–9 and Lemma 3 is applicable.

Next we first give two lemmas for branching.

Lemma 4. *F_0 is a reduced formula where each variable is a 3-variable. Let x be a 3-variable such that $F_0 = F \wedge (xy) \wedge (xC) \wedge (\bar{x}D)$, with $|D| \geq 1$. First apply R-Rule 4, and then: (1) if y is a $(2,1)$-literal in F_0, branch on y; (2) if y is a $(1,2)$-literal in F_0, branch with $y = 1$ and $y = D = 0$. As a result, we have a $(5,3)$-branching.*

Proof. Since R-Rules 1–3, 5–8 are not applicable, $|C| \geq 1$, and $y \cup C \cup D$ contains no same literals and no conflict literals (e.g. z and $\bar{z}$). By applying R-Rule 4, $(xy) \wedge (xC) \wedge (\bar{x}D)$ is replaced with $(yD) \wedge (\bar{y}CD)$. Consequently, only variables in yD become 4-variables. Since D is contained both in clauses (yD) and $(\bar{y}CD)$, both branchings with $y = 1$ and $y = 0$ can reduce D. So, the obtained formula is also a $(n, 3)$-MaxSAT formula. Next, we show it is not inferior to $(5, 3)$-branching.

Case 1: if y is a $(2, 1)$-literal in F_0, then since R-Rules 5–8, except clause (xy), there is no other clause both containing variables xy. Hence, denote the clauses with variable y as $(xy), (yE_1), (\bar{y}E_2)$, where $|E_1| \geq 1$, and let $E_1 = y'E'$. Moreover, y is a $(2, 2)$-literal after R-Rule 4 (i.e. $(yD), (\bar{y}CD), (yy'E'_1), (\bar{y}E_2)$, and k is reduced by 1). Thus, we simply branch on variable y. When $y = 1$, two clauses containing literal y are satisfied. At least one other variable y', either $y' \in D$ or $y' \notin D$, becomes a 2-variable. Therefore, R-Rules 1–3 are applicable and the number of satisfied clauses is 3. When $y = 0$, clauses $(\bar{y}E_2)$ and $(\bar{y}CD)$ are satisfied. Thus, at least one literal $z \in C$, noted that $|C| \geq 1$ and $z, \bar{z} \notin D$, becomes a 2-variable and R-Rules 1–3 become applicable again. Similarly, 3 clauses are satisfied. As a result, besides the satisfied clause by R-Rule 4 (where k is reduced by 1), we give a (4,4)-branching.

Case 2: if y is a $(1, 2)$-literal in F_0, then after R-Rule 4, y becomes a $(1, 3)$-literal. It is safe to branch with $y = 1$ and $y = D = 0$, when y is a $(1, 3)$-literal

and there is a clause (yD), because for each optimal assignment σ with $y = 0$ and $D = 1$, there is another assignment σ' by only replacing $y = 0$ with $y = 1$.

Note that $\bar{y}$ is a non-singleton, with clauses $(yD), (\bar{y}CD), (\bar{y}E_1)$ and $(\bar{y}E_2)$, because $|D| \geq 1$, and let $D = z_1 \cdots z_h$. Thus, we branch with: B1)$y = 0$, satisfying 3 clauses. Similarly, R-Rule 2 becomes applicable, since each literal in C appears at least 1 (by R-Rules 5–8) and at most 2 times in the obtained formula after R-Rule 4 and branch with $y = 0$. Therefore, at least 4 clauses are satisfied; B2)$y = 1, z_1 = \cdots = z_h = 0$, and at least a clause with y and a clause with $\bar{z}_1$ are satisfied. This is a $(5, 3)$-branching (besides R-Rule 4). □

Next, we show the branching is still better if R-Rule 4 is not applicable on a chosen variable.

Lemma 5. *F_0 is a reduced formula where each variable is a 3-variable. Let x be a non-singleton such that: $F_0 = F \wedge (xy_1y_2C_1) \wedge (xy_3y_4C_2) \wedge (\bar{x}y_5D)$. Branch on variable x, we have $(7, 2)$-branching and the obtained formula is a $(n, 3)$-*MaxSAT *formula.*

Proof. If $|D| \geq 1$, then let $D = y_6$. Since R-Rules 5–8 are not applicable, no two clauses share two common variables, so that x, y_1-y_6 are 7 different variables. Also, according to Lemma 3, there are at least 6 clauses containing at least one of 3-variables y_1, y_2, y_3 and y_4. When $x = 1$, all the 3-variables y_1, y_2, y_3 and y_4 become 2-variables. Hence, Rule 1–3 can be applied and the number of satisfied clauses is reduced by at least $2 + 4 = 6$. When $x = 0$, at least y_5 and y_6 become 2-variables, where R-Rules 1–3 are applicable. The number of satisfied clauses is reduced by at least $1 + 2 = 3$. This is a $(6, 3)$-branching.

Consider when $|D| \geq 0$ and y_5 is a $(2, 1)$-literal. When $x = 1$, y_5 is contained in unit clause (y_5). By R-Rule 2, $y_5 = 1$. Similarly, all the 3-variables $y_1y_2y_3y_4$ become 2-variables. The number of satisfied clauses is reduced by at least $2 + 1 + 4 = 7$. When $x = 0$, at least y_5 becomes a 2-variable, and the number of satisfied clauses is reduced by at least $1 + 1 = 2$. This is a $(7, 2)$-branching.

Consider when $|D| \geq 0$ and y_5 is a $(1, 2)$-literal. Similarly when $x = 1$, the number of satisfied clauses is reduced by at least $2 + 4 = 6$. When $x = 0$, no literal y_5 exists, so that $y_5 = 0$ according to R-Rule 2, and two other clauses containing y_5 are satisfied. This is a $(6, 3)$-branching.

In summary, since $\rho(7, 2) = 1.191$ and $\rho(6, 3) = 1.174$, branching on variable x, so that we have a $(7, 2)$-branching. □

Before showing the algorithm, we must first introduce a lemma.

Lemma 6. *(Bliznets [2]). If each variable of F appears once negatively and twice positively and all negative literals occur in unit clauses, then there is a polynomial time algorithm to F that returns an optimal assignment satisfying the maximum number of clauses in F.*

Summarizing all the discussions, we present our algorithm for the $(n, 3)$-MaxSAT problem in Fig. 2.

Algorithm (n,3)-MaxSAT-Solver(F, k)
INPUT: an instance (F, k) of $(n, 3)$-MaxSAT, where
each variable appears in three clauses
OUTPUT: an assignment to F that satisfies at least k clauses,
or report no such an assignment exists
1. apply R-Rules 1-3 and 5-9, in order, repeatedly until (F, k) is irreducible;
2. **if** all variables are singletons
then using Lemma 6 to solve it in polynomial time; return;
3. choose a non-singleton x: $(xC_1), (xC_2), (\bar{x}D)$ and $|D| \geq 1$;
4. **if** only one literal in C_1 or C_2 **then** using Lemma 4 for branching;
5. **else** using Lemma 5 for branching;

Fig. 2. The parameterized algorithm for $(n, 3)$-MaxSAT in time $O^*(1.194^k)$

Theorem 1. *The algorithm* **(n,3)-MaxSAT-Solver** *solves the* $(n, 3)$-MaxSAT *problem in time* $O^*(1.194^k)$.

Proof. The algorithm **(n,3)-MaxSAT-Solver** can be described as a search tree $\mathcal{T}$, where each node of T is an instance of the $(n, 3)$-MaxSAT problem. Each leaf of $\mathcal{T}$ corresponds to a reduced formula containing only singletons and their negations, where step 2 of the algorithm concludes with a decision. By Lemma 6, a leaf in the search tree $\mathcal{T}$ can be solved in polynomial time.

Each internal node of $\mathcal{T}$ is associated with an instance (F, k) and corresponds to an application of one of the branching rules in Lemmas 4 and 5, and its children correspond to the branches of the branching rule. By Lemmas 4 and 5, the root of each of the branching rules is bounded by 1.194, which is the root of the $(5, 3)$-branching. Now a simple induction shows that the search tree $\mathcal{T}$, i.e., the algorithm **Max-SAT-Solver** solves the $(n, 3)$-MaxSAT problem in time $O^*(1.194^k)$. □

4 An $O^*(1.237^n)$-time Algorithm for $(n, 3)$-MaxSAT

Note that R-Rules 2–3 and 5–9 reduce the number n of variables by at least 1, and R-Rule 1 reduces the size of the formula by at least 1. So, they can be applied in polynomial time, and we can also use them as the transformation rules for our exact algorithm in terms of n.

Next, we present our exact algorithm for $(n, 3)$-MaxSAT problem in Fig. 3. Note that a $(2, 1)$-literal z is a singleton, if $\bar{z}$ occurs in the unit clause $(\bar{z})$. A variable z is called a singleton, if literal z or $\bar{z}$ is a singleton.

Theorem 2. *The algorithm* **n3MaxSAT** *solves the* $(n, 3)$-MaxSAT *problem in time* $O^*(1.237^n)$.

Proof. The process of the algorithm **n3MaxSAT** can be depicted by a search tree $\mathcal{T}$ in which each node corresponds to an instance of the $(n, 3)$-MaxSAT

Algorithm n3MaxSAT(F)
INPUT: an instance (F, k) of $(n, 3)$-MaxSAT, where
each variable appears in three clauses
OUTPUT: an assignment to F that satisfies the maximum number of clauses
1. apply R-Rules 1-3 and 5-9 repeatedly until F is irreducible;
2. **if** all variables are singletons or there is no variable in F
then return corresponding result according to Lemma 6;
3. choose a non-singleton x: $(xC_1), (xC_2), (\bar{x}D)$ and $|D| \geq 1$;
4. **if** $|D| = 2$ (i.e. $D = yy'$) and at least one of yy' is not a singleton
then max(n3MaxSAT($F[x]$), n3MaxSAT($F[\bar{x}, \bar{y}, \bar{y}']$));
5. **else** max(n3MaxSAT($F[x]$), n3MaxSAT($F[\bar{x}]$));

Fig. 3. The exact algorithm for $(n, 3)$-MaxSAT in time $O^*(1.237^n)$

problem. Each leaf of the search tree $\mathcal{T}$ corresponds to a simplified instance for which step 2 of the algorithm concludes with a decision. Therefore, by Lemma 6, a leaf in the search tree $\mathcal{T}$ associated with instance (F, k) of $(n, 3)$-MaxSAT can be solved in polynomial time.

Each internal node of the search tree $\mathcal{T}$ either branch on x or branch with $x = 1$ and $x = y = y' = 0$. Remind that it is safe to branch $x = 1$ and $x = y = y' = 0$ in Step 4, because $\bar{x}$ is a 1-literal, for each optimal assignment σ with $x = 0$ and $y = 1$ (or $y' = 1$), there is another assignment σ' by only replacing $x = 0$ with $x = 1$.

Now we analyze the branching vector of each cases. Since R-Rule 2 and R-Rules 5–8 are not applicable, $|C_1| \geq 1, |C_2| \geq 1$ and there is no common variable among C_1, C_2 and D.

Case1. consider $y \in D$ is a 1-literal. (1) When $x = 1$, at least two variables from C_1C_2 become 2-variables and can be reduced by R-Rules 2–3. In all, the number of variables is reduced by at least 3. (2) When $x = 0$, by R-Rule 2, $y = 0$. Since R-Rules 2, 5–8 are not applicable to F, at least two other variables, except for x and y, are satisfied by $y = 0$, so that they become 2-variables. Thus, R-Rules 2–3 become applicable, and the number of variables is reduced by at least 4. This is a $(4, 3)$-branching.

Case2. consider $|D| \geq 3$, and all literals in D are 2-literals. (1) When branching $x = 1$, similar to Case 1, the number of variables is reduced by at least 3. (2) When $x = 0$, at least three variables become 2-variables, so that R-Rules 2–3 become applicable. The number of variables is reduced by at least 4. This is a $(4, 3)$-branching.

Case3. consider $|D| = 2$, and both literals yy' in D are singletons. (1) When branching $x = 1$, at least two variables, except xyy', become 2-variables and can be reduced by R-Rules 2–3. Moreover, clause $(\bar{x}yy')$ becomes (yy'), plus unit clauses $(\bar{y}), (\bar{y}')$. Thus, by R-Rule 9, replace $(yy'), (\bar{y}), (\bar{y}')$ with $(\bar{y}\bar{y}')$ and both

yy' are reduced by R-Rule 2. As a result, the number of variables is reduced by at least 5. (2)When $x = 0$, at least two variables become 2-variables, so that R-Rules 2–3 become applicable. The number of variables is reduced by at least 3. This is a $(5, 3)$-branching.

Case4. consider $|D| = 2$, and a literal y in D is not a singleton, so that Step 4 is applicable. (1) When branching $x = 1$, similar to Case 1, the number of variables is reduced by at least 3. (2) When $x = D = 0$, because y is a non-singleton, at least another variable becomes a 2-variable, so that R-Rules 2–3 become applicable. The number of variables is reduced by at least 4. This is a $(4, 3)$-branching.

Case5. consider $|D| = 1$, denoted by $D = y$, and y is a 2-literal.

Case5.1. consider $|C_1C_2| \geq 3$. (1) When branching $x = 1$, at least other three variables become 2-variables and can be reduced by R-Rules 2–3. Moreover, by R-Rule 2, $y = 1$. Thus, the number of variables is reduced by at least 5. (2) When branching $x = 0$, variable y is reduced by R-Rules 2–3. This is a $(5, 2)$-branching.

Case5.2. consider $|C_1| = |C_2| = 1$.

Case5.2.1. consider at least one literal of C_1C_2 is a (2,1)-literal. (1) When branching $x = 1$, similar to Case 1, at least 2 other variables except for xy are reduced. Moreover, since $|D| = 1$, $(2, 1)$-literal y appears in a unit clause (y), so that set $y = 1$ by R-Rule 2. In all, the number of variables is reduced by at least 4. (2) When branching $x = 0$, variable y becomes a 2-variable and can be reduced by R-Rules 2–3. Also, at least one literal of C_1C_2 is a 2-literal, so that R-Rule 2 becomes applicable. The number of variables is reduced by at least 3. This is a $(4, 3)$-branching.

Case5.2.2. consider both the literals, denoted by z_1z_2, in C_1C_2 are $(1, 2)$-literals. (1) When branching $x = 1$, by R-Rule 2, $y = 1$, and $C_1 = C_2 = 0$; (2) When branching $x = 0$, then variable y can be reduced by R-Rules 2–3. There are two cases. If there is at least another variable reduced by $x = y = 1$ and $C_1 = C_2 = 0$, then the number of variables is reduced by $4 + 1 = 5$, where we have a $(5, 2)$-branching. If there is no other variable reduced by $x = y = 1$ and $C_1 = C_2 = 0$, then, when branching $x = 0$, variable y can be reduced by R-Rules 2–3 and one of variables z_1z_2 can be reduced by R-Rules 5–8, where we have a $(4, 3)$-branching. In all, this is a $(5, 2)$-branching.

In summary, the root of each branching rule is bounded by 1.237, which is the root of the $(5, 2)$-branching. Now a simple induction shows that the search tree $\mathcal{T}$, i.e., the algorithm **n3MaxSAT** solves the $(n, 3)$-MaxSAT in time $O^*(1.237^n)$. □

5 Conclusion

In this paper we present two new reduction rules, so that each pair of distinct clauses has at most one variable in common in the reduced formula, which is also called a linear CNF formula [11]. Consequently, we improve parameterized algorithm from $O^*(1.2721^k)$ [2] to $O^*(1.194^k)$-time, and exact algorithm from $O^*(1.2600^n)$ [3] to $O^*(1.237^n)$-time for the $(n, 3)$-MaxSAT problem.

References

1. Bonet, M.L., Levy, J., Manya, F.: Resolution for max-sat. Artif. Intell. **171**(8), 606–618 (2007)
2. Bliznets, I., Golovnev, A.: A new algorithm for parameterized MAX-SAT. In: Thilikos, D.M., Woeginger, G.J. (eds.) IPEC 2012. LNCS, vol. 7535, pp. 37–48. Springer, Heidelberg (2012)
3. Bliznets, I.: A new upper bound for (n, 3)-MAX-SAT. J. Math. Sci. **188**(1), 1–6 (2013)
4. Chen, J., Kanj, I.: Improved exact algorithms for Max-SAT. Discrete Appl. Math. **142**, 17–27 (2004)
5. Chen, J., Kanj, I., Jia, W.: Vertex cover: further observations and further improvements. J. Algorithms **41**, 280–301 (2001)
6. Chen, J., Xu, C., Wang, J.: Dealing with 4-variables by resolution: an improved MaxSAT algorithm. In: Dehne, F., Sack, J.-R., Stege, U. (eds.) WADS 2015. LNCS, vol. 9214, pp. 178–188. Springer, Heidelberg (2015)
7. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**, 201–215 (1960)
8. Gu, J., Purdom, P., Wah, W.: Algorithms for the satisfiability (SAT) problem: a survey. In: Satisfiability Problem: Theory and Applications. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 19–152. AMS (1997)
9. Hochbaum, D. (ed.): Approximation Algorithms for NP-Hard Problems. PWS Publishing Company, Boston (1997)
10. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**, 367–375 (2001)
11. Porschen, S., Speckenmeyer, E., Zhao, X.: Linear CNF formulas and satisfiability. Discrete Appl. Math. **157**(5), 1046–1068 (2009)
12. Raman, V., Ravikumar, B., Rao, S.S.: A simplified NP-complete MAXSAT problem. Inf. Process. Lett. **65**(1), 1–6 (1998)
13. Kulikov, A.S.: Automated generation of simplification rules for SAT and MAXSAT. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 430–436. Springer, Heidelberg (2005)
14. Kratochvil, J., Savicky, P., Tuza, Z.: One more occurrence of variables makes satisfiability jump from trivial to NP-complete. SIAM J. Comput. **22**(1), 203–210 (1993)

http://www.springer.com/978-3-319-26625-1

Combinatorial Optimization and Applications
9th International Conference, COCOA 2015, Houston, TX, USA, December 18-20, 2015, Proceedings
Lu, Z.; Kim, D.; Wu, W.; Li, W.; Du, D.-Z. (Eds.)
2015, XIII, 810 p. 198 illus. in color., Softcover
ISBN: 978-3-319-26625-1