Journal of Combinatorial Optimization
https://doi.org/10.1007/s10878-019-00421-1

An improved algorithm for the $(n, 3)$-MaxSAT problem: asking branchings to satisfy the clauses

Chao Xu[1] · Wenjun Li[2] · Jianxin Wang[1] · Yongjie Yang[1,3]

Abstract

We study the $(n, 3)$-MaxSAT problem where we are given an integer k and a CNF formula with n variables, each of which appears in at most 3 clauses, and the question is whether there is an assignment that satisfies at least k clauses. Based on refined observations, we propose a branching algorithm for the $(n, 3)$-MaxSAT problem which significantly improves the previous results. More precisely, the running time of our algorithm can be bounded by $O^*(1.175^k)$ and $O^*(1.194^n)$, respectively. Prior to our study, the running time of the best known exact algorithm can be bounded by $O^*(1.194^k)$ and $O^*(1.237^n)$, respectively.

Keywords CNF formula · 3-SAT · Branching algorithm · Complexity · Parameterized complexity

A preliminary version of this paper appeared in Proceedings of the International Conference on Combinatorial Optimization and Applications (COCOA 2017) (Li et al. 2017). This work is supported by the National Natural Science Foundation of China (Grant Nos. 61502054, 61672536, 61702557, 61420106009, 61872048), the Natural Science Foundation of Hunan Province, China (Grant No. 2017JJ3333), the Scientific Research Fund of Hunan Provincial Education Department (Grant No. 17C0047), the China Postdoctoral Science Foundation (Grant No. 2017M612584), and the Postdoctoral Science Foundation of Central South University.

✉ Yongjie Yang
yyongjiecs@gmail.com

Chao Xu
xuchaofay@csu.edu.cn

Wenjun Li
liwenjun@csu.edu.cn

Jianxin Wang
jxwang@mail.csu.edu.cn

[1] School of Computer Science and Engineering, Central South University, Changsha, China

[2] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha, China

[3] Chair of Economic Theory, Saarland University, Saarbrücken, Germany

1 Introduction

Given a propositional formula F in the conjunctive normal form (CNF formula), the Maximum Satisfiability problem (MaxSAT) asks for an assignment satisfying the maximum number of clauses in F. It is well-known that the MaxSAT problem is NP-hard even in several special cases (Raman et al. 1998). Due to the significant applications of the MaxSAT problem in a wide range of areas (see, e.g., Berg et al. 2015; Poloczek et al. 2017; Saikko et al. 2015), much effort has been made in order to solve the MaxSAT problem by both theorists and practitioners. In particular, a number of heuristic, approximation, randomized, and exact algorithms have been studied in the literature (Bliznets and Golovnev 2012; Bliznets 2013; Goemans and Williamson 1994, 1995; Hochbaum 1997; Karloff and Zwick 1997; Kulikov 2005; Luo et al. 2017; Poloczek et al. 2017; Raman et al. 1998), and numerous competitive MaxSAT solvers have been developed over the past few years (Argelich and Manyà 2006; Hutter et al. 2017).

In the parameterized version of the MaxSAT problem (parameterized MaxSAT), we are given additionally an integer k and are asked whether there is an assignment satisfying at least k clauses. Let n be the number of variables in F. For two positive integers s and t, let (s, t)-MaxSAT denote the special case of the parameterized MaxSAT problem where each clause includes at most s literals and each variable appears in at most t clauses. It is clear that if $s = 1$ or $t = 1$, the (s, t)-MaxSAT problem is polynomial-time solvable. In addition, it is proved that for $t = 2$ the (s, t)-MaxSAT problem remains polynomial-time solvable for all positive integers s (Raman et al. 1998). However, if t further increases by one, the problem becomes NP-hard even for every $s \geq 2$, i.e., the $(2, 3)$-MaxSAT problem is NP-hard (Raman et al. 1998).

In the special case where k is equal to the number of clauses, i.e., all clauses should be satisfied, we obtain the Satisfiability problem (SAT) which has also been extensively studied from different aspects in the literature (Calabro et al. 2006; Gu 1994; Hirsch 2000; Hirsch and Kojevnikov 2005; Patrascu and Williams 2010; Selman et al. 1996). In contrast to the NP-hardness of the $(2, 3)$-MaxSAT problem, the special case of the SAT problem where every clause consists of at most two literals is well-known to be polynomial-time solvable (Aspvall et al. 1979). However, if we allow every clause to contain one more literal, the problem becomes NP-hard (Cook 1971; Garey and Johnson 1979). The SAT problem also plays a significant role in showing algorithmic lower bounds for many computationally hard problems (Cygan et al. 2015; Impagliazzo and Paturi 2001).

In this paper, we study an algorithm for the $(n, 3)$-MaxSAT problem. Over the last two decades, many researchers have devoted themselves to developing faster algorithms for the $(n, 3)$-MaxSAT problem. See Table 1 for the recent progress. The algorithms in the table are measured in terms of the number n of variables and the number k of clauses that are desired to be satisfied. We would like to point out that Lokshtanov (2009) (Theorem 4.2.1) derived a polynomial-time algorithm (in fact, this is a kernelization algorithm from the parameterized complexity point of view) which takes as input an instance (F, k), and outputs an equivalent instance (F', k') such that $k' \leq k$ and the number of variables in the new instance is at most k'. Given this polynomial-time algorithm, if there is an $O^*(c^n)$-time algorithm for the $(n, 3)$-

Table 1 Recent progress of exact algorithms for the $(n, 3)$-MaxSAT problem

Bound w.r.t. k	Bound w.r.t. n	References	Years
	$O^*(1.73206^n)$	Raman et al. (1998)	1998
	$O^*(1.32472^n)$	Bansal and Raman (1999)	1999
$O^*(1.3247^k)$ [1]		Chen and Kanj (2004)	2004
	$O^*(1.27203^n)$	Kulikov (2005)	2005
$O^*(1.2721^k)$		Bliznets and Golovnev (2012)	2012
	$O^*(1.25993^n)$	Bliznets (2013)	2013
$O^*(1.194^k)$	$O^*(1.237^n)$	Xu et al. (2016)	2016
$O^*(1.175^k)$	$O^*(1.194^n)$	This paper	–

This result follows from the proof of Chen and Kanj (2004) for the general MaxSAT problem

MaxSAT problem where c is a constant, we can obtain an $O^*(c^k)$-time algorithm by firstly running this polynomial-time algorithm, and then running the $O^*(c^n)$-time algorithm on the instance returned by the polynomial-time algorithm.

We present an algorithm whose running time can be bounded by $O^*(1.175^k)$ and $O^*(1.194^n)$. Our result significantly improves the previous algorithm whose running time is bounded by $O^*(1.194^k)$ and $O^*(1.237^n)$ (Xu et al. 2016), respectively. The main ingredients of our algorithm are reduction rules and branching rules. In particular, a reduction rule transforms an instance in polynomial time into an equivalent new instance with several useful properties. A branching rule prescribes how to iteratively divide an instance into several subinstances, such that the original instance is a Yes-instance if and only if at least one of the subinstances is a Yes-instance. We present the reduction rules in Sect. 3 and branching rules in Sect. 4. Branching algorithms are commonly used to solve hard problems, and the current best exact algorithms for many combinatorial problems are branching algorithms (see, e.g., Chen et al. 2010; Shen and Zhang 2005; Xiao and Nagamochi 2016).

2 Preliminaries

We assume familiarity with propositional logic. A *Boolean variable* is a variable that can take the values 1 (TRUE) or 0 (FALSE). We will simply use "variable" for "Boolean variable" as we consider only Boolean variables in the paper. Let V be a set of variables. For a variable $x \in V$, $\overline{x}$ is the *negation* of x. So, we have $\overline{\overline{x}} = x$. A *literal* is either a variable or its negation. In particular, the literal is called a *positive literal* in the former case and a *negative literal* in the latter case. For ease of exposition, for a literal x we use $v(x)$ to denote the variable associated with x. So, $v(x) = v(\overline{x})$.

A *clause* C is a disjunction of literals, for example, $C = x_1 \vee x_2 \vee x_3$ is a clause of three literals x_1, x_2, and x_3. For simplicity, we omit the symbol $\vee$ in clauses. Hence, $x_1x_2x_3$ is the clause $x_1 \vee x_2 \vee x_3$. Note that the order of the literals in a clause is irrelevant to the clause's identity. Hence, $x_1x_2x_3 = x_2x_1x_3$. We use $v(C)$ to denote the set of variables associated to literals in C. For example, for $C = xyz\overline{y}$, we have

$v(C) = \{v(x), v(y), v(z)\}$. For a literal x, we write $x \in C$ for the fact that x occurs in C. We say a clause C includes a variable if it includes at least one of its literals. The *size* of a clause C, denoted by $|C|$, is the number of literals in C. A clause is an *h-clause* if it is of size h. A 1-clause is also referred to as a *unit clause*. For two clauses C_1 and C_2, we use C_1C_2 to denote the clause consisting of literals in C_1 and C_2. For instance, if $C_1 = xy$ and $C_2 = z$, then C_1C_2 is the clause xyz.

A *CNF formula* F over V is a conjunction of a number of clauses including variables in V. A literal x is an *(i, j)-literal* in F if x and $\overline{x}$ occur exactly i and j times in F, respectively. Therefore, a literal x is an (i, j)-literal if and only if $\overline{x}$ is a (j, i)-literal. The *degree* of the variable $v(x)$ associated to x is defined as $i + j$.

An *assignment* is a function $f : V \to \{0, 1\}$. A clause C is *satisfied* by f if there is a positive literal $x \in C$ such that $f(v(x)) = 1$, or a negative literal $\overline{x} \in C$ such that $f(v(\overline{x})) = 0$. In the former case, we also say that x is assigned 1 under f (denoted by $x = 1$ or $\overline{x} = 0$), and in the latter case, we say that x is assigned 0 under f (denoted by $x = 0$ or $\overline{x} = 1$).

3 Reduction rules and some properties

A *reduction rule* converts in polynomial time an instance (F, k) of the $(n, 3)$-MaxSAT problem into another instance (F', k') such that $k' \leq k$ and (F, k) is a Yes-instance if and only if (F', k') is a Yes-instance. An instance is *reduced* by a set of reduction rules if none of these reduction rules applies to the instance. A reduction rule is *sound* if the answer to the instance remains the same before and after the application of the reduction rule. In what follows, $(F, k) \to (F', k')$ describes a reduction rule that transforms the instance (F, k) into the instance (F', k'). Before applying the ith reduction rule, $i \geq 2$, we assume that the instance is reduced by all reduction rules introduced before.

Observe that if a literal occurs several times in a clause in F, we can simply remove all of them except any arbitrary one without changing the answer to the instance. Moreover, if two literals x and $\overline{x}$ are in a clause C, then C is satisfied by any assignment. Therefore, we can safely remove C from F and decrease k by 1 without changing the answer to the instance. These observations lead to the following two reduction rules, which have been studied in the literature (Chen and Kanj 2004; Niedermeier and Rossmanith 2000).

R-Rule 1 (Chen and Kanj 2004; Niedermeier and Rossmanith 2000) *Let x be a literal. If there is a clause xxC, then replace it with xC, i.e.,* $(F = F' \wedge (xxC), k) \to (F = F' \wedge (xC), k)$.

R-Rule 2 (Chen and Kanj 2004; Niedermeier and Rossmanith 2000) *Let x be a literal. If there is a clause $x\overline{x}C$, then remove the clause and decrease k by 1, i.e.,* $(F = F' \wedge (x\overline{x}C), k) \to (F = F', k - 1)$.

The following two reduction rules have also been studied in the literature (Bansal and Raman 1999; Chen and Kanj 2004; Niedermeier and Rossmanith 2000). Throughout this paper, when we assign $x = 1$ (or $\overline{x} = 0$) for some literal x, we assume the following operations are done automatically: (1) remove all clauses including x (as they are all satisfied when $x = 1$); and (2) remove $\overline{x}$ from all clauses including the literal $\overline{x}$.

R-Rule 3 (Bansal and Raman 1999; Chen and Kanj 2004; Niedermeier and Rossmanith 2000) *If there is an* (i, j)*-literal* x *such that* $i \geq j \geq 0$ *and there are at least* j *unit clauses* x *in* F*, then* $(F, k) \rightarrow (F_{x=1}, k - i)$*, where* $F_{x=1}$ *is the formula obtained from* F *by assigning* $x = 1$.

Generally speaking, in the case stated in the above rule, if there is a solution under which $x = 0$, we can change $x = 1$ to obtain another solution because doing so not only changes all the unsatisfied j unit clauses x into satisfied but also guarantees that all the clauses including x are satisfied. Note that if there is an $(i, 0)$ literal x, then due to R-Rule 3 we can assign $x = 1$.

R-Rule 4 (Bansal and Raman 1999; Chen and Kanj 2004; Niedermeier and Rossmanith 2000) *If there is a* $(1, 1)$*-literal* x*, then*

$$(F = F' \wedge (xC_1) \wedge (\overline{x}C_2), k) \rightarrow (F = F' \wedge (C_1C_2), k - 1),$$

i.e., replacing the two clauses xC_1 *and* $\overline{x}C_2$ *including* x *and* $\overline{x}$ *respectively with the clause* C_1C_2*, and decreasing* k *by 1.*

Clearly, if none of R-Rules 1–4 is applicable, each variable is of degree 3. Precisely, every literal x is either a (1, 2)-literal or a (2, 1)-literal. In addition, for each (2, 1)-literal x, there is no unit clause x.

R-Rule 5 (Bonet et al. 2007; Chen and Kanj 2004) *If there are three clauses* xy, $\overline{x}$, $\overline{y}$*, then replace these three clauses with the clause* $\overline{x}\,\overline{y}$ *and decrease* k *by 1, i.e.,*

$$(F = F' \wedge (xy) \wedge (\overline{x}) \wedge (\overline{y}), k) \rightarrow (F = F' \wedge (\overline{x}\,\overline{y}), k - 1).$$

Now we study two new reduction rules.

R-Rule 6 *If there are five clauses* xy_1, xy_2, $\overline{x}y_3$, $\overline{y_3}y_1$, $\overline{y_3}y_2$*, then assign* $x = 1$ *and decrease* k *by 2, i.e.,*

$$(F' \wedge (xy_1) \wedge (xy_2) \wedge (\overline{x}y_3) \wedge (\overline{y_3}y_1) \wedge (\overline{y_3}y_2), k)$$
$$\rightarrow (F' \wedge (y_3) \wedge (\overline{y_3}y_1) \wedge (\overline{y_3}y_2), k - 2).$$

Now we prove the soundness of R-Rule 6.

Lemma 1 *R-Rule 6 is sound.*

Proof To prove the theorem, it suffices to show that there is an optimal assignment which assigns x the value 1. We refer to Table 2 for the satisfiability of the five clauses stated in R-Rule 6 under different assignments over x and y_3.

Clearly, due to Table 2, if we have an assignment f under which $x = 0$, $y_3 = 1$, we can obtain a new assignment from f by reassigning $x = 1$ and keeping other

Table 2 Comparison of assignments $x = 1$ and $x = 0$ in the proof of Lemma 1

	xy_1	xy_2	$\overline{x}y_3$	$\overline{y_3}y_1$	$\overline{y_3}y_2$
$x = 1, y_3 = 1$	✓	✓	✓	y_1	y_2
$x = 0, y_3 = 0$	y_1	y_2	✓	✓	✓
$x = 0, y_3 = 1$	y_1	y_2	✓	y_1	y_2

The entries marked with ✓ mean that the corresponding clauses are satisfied

assignments unchanged, which satisfies at least the same number of clauses as f does. Similarly, if under an assignment f we have $x = y_3 = 0$, we can obtain another assignment from f by reassigning $x = y_3 = 1$ and keeping other assignments unchanged, which satisfies at least the same number of clauses as f does. Hence, we can conclude that there is an optimal assignment under which $x = 1$. Notice that we did not show that an optimal assignment must assign $x = y_3 = 1$; it may exist an optimal assignment under which $x = 1$ and $y_3 = 0$. □

R-Rule 7 *If there are eight clauses xy_1y_2, xy_3y_4, $\overline{x}y_5$, $y_5y_1y_3$, $\overline{y_5}$, $\overline{y_1}y_4$, $\overline{y_3}$, and $\overline{y_4}$, then*

$$(F, k) \to (F_{x=y_5=1, y_1=y_3=y_4=0}, k - 7),$$

where $F_{x=y_5=1, y_1=y_3=y_4=0}$ is the formula obtained from F by assigning $x = y_5 = 1$ and $y_1 = y_3 = y_4 = 0$.

Lemma 2 *R-Rule 7 is sound.*

Proof Clearly, xy_1y_2, xy_3y_4, $\overline{x}y_5$, $y_5y_1y_3$, $\overline{y_5}$, $\overline{y_1}y_4$, $\overline{y_3}$, and $\overline{y_4}$ are all the clauses including the variables $v(x)$, $v(y_1)$, $v(y_3)$, $v(y_4)$, and $v(y_5)$. Observe that there is no assignment satisfying all these eight clauses. Assigning $x = y_5 = 1$ and $y_1 = y_3 = y_4 = 0$ satisfies all but only one of these clauses. Hence, if there is an optimal assignment, we can obtain another optimal assignment from this assignment by reassigning $x = y_5 = 1$ and $y_1 = y_3 = y_4 = 0$. □

Apart from the above reduction rules, many other reduction rules have been explored in the literature. A significant consequence of an exhaustive use of these reduction rules is a reduced instance with some useful properties. These properties have been explicitly given in Xu et al. (2016). In order not to distract the reader and also because that only the properties of the reduced instance matter for our study, we ignore the details of these reduction rules but give a lemma to summarize these properties. We provide the details of R-Rules 1–7 since they are used in the analysis of the branching algorithm in the next section.

We say a CNF formula F is *linear* if no two variables $v(x)$ and $v(y)$ are simultaneously included in two clauses, i.e., there are no two clauses C and C' such that $v(x), v(y) \in v(C)$ and $v(x), v(y) \in v(C')$.

Lemma 3 (Xu et al. 2016) *There is a polynomial-time algorithm which takes as input an instance (F, k) of the $(n, 3)$-MaxSAT problem and outputs a new instance (F', k') of the $(n, 3)$-MaxSAT problem such that*

(1) $k' \leq k$*;*
(2) (F, k) *is a Yes-instance if and only if* (F', k') *is a Yes-instance; and*
(3) (F', k') *is linear.*

The following lemma copes with some special cases of the $(n, 3)$-MaxSAT problem.

Lemma 4 (Bliznets and Golovnev 2012) *If for all* $(1, 2)$*-literals* x*, the clause including* x *is a unit clause, then the instance can be solved in polynomial time.*

Lemma 5 (Bliznets and Golovnev 2012; Xu et al. 2016) *Let* x *be a* $(2, 1)$*-literal and* C_1*,* C_2 *the two clauses including* x*. If there is an optimal assignment* f *such that* $f(x) = 0$*, then* f *either satisfies both* C_1 *and* C_2*, or we can reset* $f(x) = 1$ *and still keep* f *optimal.*

Proof Assume that an optimal assignment f does not satisfy at least one of C_1 and C_2, say, without loss of generality, C_i, where i is either 1 or 2. Then, it must be that $f(x) = 0$. If we reassign $x = 1$ and keep the assignments of other variables unchanged in f, then the clause including $\overline{x}$ may become unsatisfied. However, C_i becomes satisfied (C_{3-i} is also satisfied in this case). Hence, the number of satisfied clauses does not decrease after reassigning $x = 1$. □

Assigning a value to a variable in an instance reduced by some reduction rules might lead the reduction rules to be applicable again. The following lemma offers the extent of the decrease of k and n in such cases.

Lemma 6 *Let* (F, k) *be an instance of the* $(n, 3)$*-MaxSAT problem such that* F *is linear and* (F, k) *is reduced by R-Rules 1–4. Let* x *be a* $(2, 1)$*-literal in* F*, and let* xC_1 *and* xC_2 *be the two clauses including the literal* x*. Then, after assigning* $x = 1$ *we can exhaustively apply R-Rules 1–4 in a way so that all variables in* $v(C_1C_2)$ *are reduced, and both* k *and* n *decrease by at least* $|C_1| + |C_2|$*.*

Proof As F is linear, it must be that $v(C_1) \cap v(C_2) = \emptyset$. Let $i = |C_1| + |C_2|$. After assigning $x = 1$, all variables in $v(C_1C_2)$ are of degree 2. Hence, after an exhaustive application of R-Rules 1–4, all variables in $v(C_1C_2)$ are reduced. Therefore, the number of variables n decreases by at least i.

It remains to analyze the amount of the decrease of k. Recall that all variables are of degree 3 in F. Hence, after assigning $x = 1$, $v(C_1C_2)$ is exactly the set of variables of degree 2.

First, we exhaustively apply R-Rule 4. Each application reduces one variable from $v(C_1C_2)$ and decreases k by 1. Hence, if i' variables in $v(C_1C_2)$ are reduced in the applications of R-Rule 4, then k decreases by i'. To proceed our proof, we need the following claim. Let F' be the new CNF formula after an exhaustive application of R-Rule 4.

Claim Every clause in F' includes at most two literals of $v(C_1C_2)$.

Proof As F is linear and reduced by R-Rules 1–2, every clause in F, except xC_1 and xC_2, includes at most two literals of $v(C_1C_2)$. Each application of R-Rule 4 replaces two clauses zC and $\overline{z}C'$ with the clause CC', where $v(z) \in v(C_1C_2)$. Hence, if both C and C' include at most one literal in $v(C_1C_2)$, CC' includes at most two literals in $v(C_1C_2)$. This completes the proof of the claim.

After exhaustively applying R-Rule 4, we exhaustively apply R-Rule 2 over literals of $v(C_1C_2)$. Each application of R-Rule 2 removes a clause $z\bar{z}C$ such that $v(z) \in v(C_1C_2)$ and, moreover, decreases k by 1. Note that due to the above claim, C does not include any other variable of $v(C_1C_2)$ except $v(z)$. Hence, if R-Rule 2 is applied i'' times, then i'' variables in $v(C_1C_2)$ are reduced and k decreases by i''. Let F'' be the new CNF formula. Clearly, every clause in F'' is a clause in F'. Hence, due to the above claim, every clause in F'' includes at most 2 literals of $v(C_1C_2)$.

Finally, we exhaustively apply R-Rule 3. Note that in this case, each literal z such that $v(z) \in v(C_1C_2)$ is either a $(2,0)$-literal or a $(0,2)$-literal in F''. Assume that in F'' there are j variables from $v(C_1C_2)$. So, $i'+i''+j = i$. Let A be the set of clauses in F'' that include some variable from $v(C_1C_2)$. As every clause in F'' includes at most 2 literals from $v(C_1C_2)$, it holds that $|A| \geq j$. Moreover, an exhaustive application of R-Rule 3 can reduce all clauses in A and decrease k by $|A|$ (Due to R-Rule 3, every $(2,0)$-literal z is assigned the value 1 and every $(0,2)$-literal z is assigned the value 0, making all clauses in A be satisfied).

In total, the parameter k decreases by at least $i' + i'' + |A| \geq i' + i'' + j = i$. $\square$

4 Branching rules

Branching on an instance of a problem divides the instance into several subinstances such that the original instance is a Yes-instance if and only if at least one of the subinstances is a Yes-instance. To measure the running time of a branching algorithm, we need a parameter associated with the problem so that the parameter of each subinstance is strictly smaller than that of the original instance. In particular, if we have a parameter d in the original instance and a branching divides the instance into t subinstances with parameters respectively being $d_1, d_2, \ldots, d_t$, we call $\langle d-d_1, d-d_2, \ldots, d-d_t\rangle$ the *branching vector* of the branching. The number t is called the *branching number*. The *branching root* is the unique positive root of the following polynomial:

$$x^d - x^{d-d_1} - x^{d-d_2} - \cdots - x^{d-d_t} = 0.$$

We say that a branching is *superior* to another branching if the branching root of the former one is no greater than that of the latter one. If the branching roots of all branchings in a branching algorithm are bounded from above by a constant c, the running time of the algorithm is bounded by $O^*(c^d)$, where d is the associated parameter in the original instance. We refer to Fomin and Kratsch (2010) for a gentle introduction to branching algorithms.

In our study, we consider particularly the two parameters k and n, where k is the number of clauses that are desired to be satisfied and n is the number of variables in the given CNF formula. We consider only branchings with branching number 2. For ease of exposition, we call a branching with a branching vector $\langle d_1, d_2\rangle$ with respect to k (resp. n) a $\langle d_1, d_2\rangle$-k-branching (resp. $\langle d_1, d_2\rangle$-n-branching).

Let (F, k) be an instance reduced by the reduction rules R-Rules 1–7 and the polynomial-time algorithm stated in Lemma 3, i.e., F is linear and none of the reduc-

Table 3 Summary of branching vectors and branching roots of branchings on a (2, 1)-literal x

$\lvert C_i\rvert$	$\lvert C_{3-i}\rvert$	$\lvert C_3\rvert$	k Branching vector	Branching root	n Branching vector	Branching root	Evidence
≥ 2	≥ 2	≥ 2	$\langle 6, 3\rangle$	1.174	$\langle \mathbf{5}, \mathbf{3}\rangle$	**1.194**	Lemma 7
1	≥ 2	≥ 2	$\langle 5, 4\rangle$	1.168	$\langle 4, 4\rangle$	1.190	Lemma 8
1	1	≥ 2	$\langle 4, 5\rangle$	1.168	$\langle \mathbf{3}, \mathbf{5}\rangle$	**1.194**	Lemma 9
1	1	1	$\langle 4, 5\rangle$	1.168	$\langle \mathbf{3}, \mathbf{5}\rangle$	**1.194**	Lemma 10
≥ 2	1	1	$\langle 6, 3\rangle$	1.174	$\langle \mathbf{5}, \mathbf{3}\rangle$	**1.194**	Lemma 11
≥ 3	≥ 2	1	$\langle \mathbf{8}, \mathbf{2}\rangle$	**1.175**	$\langle 7, 2\rangle$	1.191	Lemma 12
2	2	1	$\langle \mathbf{8}, \mathbf{2}\rangle$	**1.175**	$\langle 7, 2\rangle$	1.191	Lemma 13

The bold texts refer to the worst case of the branching
The three clauses including $v(x)$ are xC_1, xC_2, and $\overline{x}C_3$. Here, either $i = 1$ or $i = 2$

tion rules applies to (F, k). If the condition stated in Lemma 4 is satisfied, we can solve (F, k) in polynomial-time. So, assume that this is not the case. As a result, there must be a (2, 1)-literal x such that there are three clauses $xC_1, xC_2, \overline{x}C_3$ where $|C_i| \geq 1$ for every $i \in \{1, 2, 3\}$. All our branchings are on such (2, 1)-literals x in F. In particular, for each such x we consider the branchings with assignments $x = 1$ and $x = 0$. In each branching, we can further reduce the instance by exhaustively applying the reduction rules and the polynomial-time algorithm stated in Lemma 3, leading to the decrease of both k and n. We analyze the branching vectors of all cases in the following lemmas, by distinguishing between the sizes of C_1, C_2, and C_3. Table 3 summarizes these results. Due to R-Rules 1–7 and the polynomial-time algorithm in Lemma 3, we only branch on linear CNF formulae F where every literal is either a (2, 1)-literal or a (1, 2) literal. Moreover, for each (2, 1)-literal (resp. (1, 2)-literal) x, no clause including the literal x (resp. $\overline{x}$) is a unit clause (due to R-Rule 3).

In the following, let x be a (2, 1)-literal and xC_1, xC_2, $\overline{x}C_3$ be the three clauses including $v(x)$ in F as discussed above. Please bear in mind that $v(C_1)$, $v(C_2)$, and $v(C_3)$ are pairwise disjoint since F is linear.

Lemma 7 *If $|C_i| \geq 2$ for each $1 \leq i \leq 3$, branching on x leads to a branching superior to a $\langle 6, 3\rangle$-k-branching (resp. $\langle 5, 3\rangle$-n-branching).*

Proof When branching with $x = 1$, clauses xC_1 and xC_2 are satisfied and removed, leading k and n to be decreased by 2 and 1, respectively. Then, due to Lemma 6, k and n can be further decreased by at least $|C_1| + |C_2| \geq 4$ by exhaustively applying R-Rules 1–4. In total, k (resp. n) decreases by at least $2 + 4 = 6$ (resp. $1 + 4 = 5$).

Consider the branching $x = 0$. Similarly, the clause $\overline{x}C_3$ is removed, and both k and n are decreased by 1. Then, at least two variables (in $v(C_3)$) become of degree 2, which can be reduced by R-Rules 1–4, leading n to be decreased by 2. Meanwhile, as F is linear, k can be decreased by at least 2. In total, both k and n are decreased by at least 3. □

Lemma 8 *Let $i \in \{1, 2\}$. If $|C_i| \geq 2$, $|C_{3-i}| = 1$, and $|C_3| \geq 2$, branching on x leads to a branching superior to a $\langle 5, 4\rangle$-k-branching (resp. $\langle 4, 4\rangle$-n-branching).*

Proof Let y_1, y_2 be any two arbitrary literals in C_i, y_3 be the literal in C_{3-i}, and y_4, y_5 be any two arbitrary literals in C_3. Hence, $xC_{3-i} = xy_3$.

Consider first the branching $x = 1$. Clearly, clauses xC_i and xy_3 are satisfied and removed. Accordingly, we decrease k by 2. After this, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are of degree 2. Due to Lemma 6, applying R-Rules 1–4 exhaustively reduces these variables and decreases k by at least 3. In total, k decreases by at least $2+3 = 5$ and n decreases by at least 4 (the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are reduced).

Consider now the branching $x = 0$. Due to Lemma 5, there is an optimal assignment satisfying xC_i and xy_3. If this optimal assignment is under this branching, then y_3 must be assigned the value 1 in this optimal assignment. Hence, we assign $y_3 = 1$, remove the satisfied clauses including $\overline{x}C_3$ and xy_3, and accordingly decrease k (by at least 2). After this, $v(y_4)$ and $v(y_5)$ are of degree 1 or 2. Then, R-Rules 3–4 can be applied to reduce $v(y_4)$ and $v(y_5)$, which leads to a decrease of k by at least 2. In total, k decreases by at least $2 + 2 = 4$ and n decreases by at least 4 (the variables $v(x)$, $v(y_3)$, $v(y_4)$, and $v(y_5)$ are reduced). □

Lemma 9 *If $|C_1| = |C_2| = 1$ and $|C_3| \geq 2$, branching on x leads to a branching superior to a $\langle 4, 5\rangle$-k-branching (resp. $\langle 3, 5\rangle$-n-branching).*

Proof Without loss of generality, let $C_1 = y_1$, $C_2 = y_2$, and $C_3 = y_3y_4C$ such that $|C| \geq 0$.

When branching with $x = 1$, clauses xy_1 and xy_2 are satisfied, and k is decreased by 2. Due to Lemma 6, k and n can be further decreased by at least $|C_1| + |C_2| = 2$. In total, k decreases by at least $2 + 2 = 4$, and n decreases by at least 3 (the variables $v(x)$, $v(y_1)$, and $v(y_2)$ are reduced).

Consider the branching $x = 0$. Due to Lemma 5, there is an optimal assignment under which $y_1 = y_2 = 1$. Hence, we assign y_1 and y_2 the value 1. As a result, clauses xy_1, xy_2, and $\overline{x}y_3y_4C$ are satisfied, and hence k can be decreased by 3. Then, $v(y_3)$ and $v(y_4)$ become of degree 2. Applying R-Rules 1–4 reduces $v(y_3)$ and $v(y_4)$, and decreases k by at least 2. In total, k decreases by at least $3 + 2 = 5$ and n decreases by at least 5 (the variables $v(x)$, $v(y_1)$, ..., $v(y_4)$ are reduced). □

Lemma 10 *If $|C_i| = 1$, say, $C_i = y_i$, for every $1 \leq i \leq 3$, branching on x leads to a branching superior to a $\langle 4, 5\rangle$-k-branching (resp. $\langle 3, 5\rangle$-n-branching).*

Proof We prove the lemma by distinguishing between the following two cases.

Case 1. y_3 is a (2, 1)-literal.

Let y_3C denote the other clause including the literal y_3. Due to R-Rule 3, we have that $|C| \geq 1$.

If we assign $x = 1$, then clauses xy_1 and xy_2 are satisfied, and k is decreased by 2. Due to Lemma 5, there is an optimal assignment satisfying both $\overline{x}y_3$ and y_3C. In this optimal assignment, y_3 must be assigned the value 1. Hence, we assign $y_3 = 1$. As a result, the clauses $\overline{x}y_3$ and y_3C are satisfied, and hence we decrease k by 2. In addition, $v(y_1)$ and $v(y_2)$ become of degree 1 or 2, and R-Rules 3–4 are applied, reducing $v(y_1)$ and $v(y_2)$, and decreasing k by at least 2. In total, k decreases by at least $2+2+2 = 6$. Moreover, the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are reduced. Hence, n decreases by at least 4.

Consider the branching $x = 0$. Clearly, $\overline{x}y_3$ is satisfied and k can be decreased by 1. In addition, due to Lemma 5, there is an optimal assignment satisfying both xy_1 and xy_2. Hence, similar to the above argument for assigning y_3, we assign $y_1 = y_2 = 1$, and decrease k by 2 (as xy_1 and xy_2 are satisfied). After this, $v(y_3)$ is of degree 2, and hence R-Rules 1–4 are applicable and k can be decreased by at least 1. In total, k decreases by at least $1 + 2 + 1 = 4$ and n decreases by at least 4 (the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are reduced).

In summary, in this case we have a branching superior to a $\langle 6, 4\rangle$-k-branching (resp. $\langle 4, 4\rangle$-n-branching), which is superior to a $\langle 4, 5\rangle$-k-branching (resp. $\langle 3, 5\rangle$-n-branching) as stated in the lemma.

Case 2. y_3 is a $(1, 2)$-literal.

If we assign $x = 1$, clauses xy_1 and xy_2 are satisfied, and we decrease k by 2. Then, $v(y_1)$ and $v(y_2)$ are of degree 2. Due to Lemma 6, exhaustive applications of R-Rules 1–4 reduce $v(y_1)$, $v(y_2)$, and decrease k by at least 2. In total, k decreases by at least $2 + 2 = 4$. As the variables $v(x)$, $v(y_1)$, and $v(y_2)$ are reduced, n decreases by at least 3.

Consider the branching case $x = 0$ now. Let $\overline{y_3}D_1$ and $\overline{y_3}D_2$ be the two clauses including $\overline{y_3}$. Note that $\min\{|D_1|, |D_2|\} \geq 1$, since otherwise R-Rule 3 is applicable to F. Similar to the proof of Lemma 9, we assign $y_1 = y_2 = 1$, remove the satisfied clauses xy_1, xy_2, and $\overline{x}y_3$, and decrease k by 3. Now $\overline{y_3}$ is a $(2, 0)$-literal. As a result, $\overline{y_3}D_1$ and $\overline{y_3}D_2$ are removed by R-Rule 3 and k is decreased by 2 accordingly. In total, k decreases by at least $3+2 = 5$. Hence, we have already a desired branching superior to a $\langle 4, 5\rangle$-k-branching stated in the lemma. Now we focus on the parameter n which has decreased by 4 so far (the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are reduced). If there is a variable $v(y_4) \notin \{v(x), v(y_1), v(y_2), v(y_3)\}$ in one of D_1 and D_2, then $v(y_4)$ is of degree 2 now and hence can be reduced by R-Rule 3 or R-Rule 4. Accordingly, n further decreases by at least 1; we are done. In fact, as F is linear, no literal of $v(x)$ is in D_1 and D_2. As F is reduced by the reduction rules, no literal of $v(y_3)$ is in D_1 and D_2 too. Hence, it only remains to consider the cases where D_1 and D_2 include only literals of $v(y_1)$ and $v(y_2)$. If $\max\{|D_1|, |D_2|\} \geq 2$, then there must be a variable $v(y_4)$ as discussed above. Hence, assume that $|D_1| = |D_2| = 1$. Note that $v(D_1) \cap v(D_2) = \emptyset$ since F is linear.

Case 2.1. One of y_1 and y_2 is a $(1, 2)$-literal in F.

In this case, n can be decreased only by 4 in the branching $x = 0$ in the worst case, as discussed above. However, we show that when assigning $x = 1$, we can actually decrease n by at least 4, and hence achieve a branching superior to a $\langle 4, 4\rangle$-n-branching, which is superior to a $\langle 3, 5\rangle$-n-branching stated in the Lemma. Due to symmetry, assume that y_1 is a $(1, 2)$-literal. Then, one of the clauses containing $\overline{y_3}$ must be $\overline{y_3}\,\overline{y_1}$ (as F is linear, and D_1 and D_2 include only literals of $v(y_1)$, $v(y_2)$). After assigning $x = 1$, we can first assign $y_1 = 0$ and then assign $y_3 = 1$ due to R-Rule 3. Moreover, $v(y_2)$ can be also reduced by R-Rule 3 or R-Rule 4. Therefore, we have the desired $\langle 4, 4\rangle$-n-branching in this case.

Case 2.2. y_1 and y_2 are both $(2, 1)$-literals, and one of D_1 and D_2 includes a literal $\overline{y_i}$ for some $i \in \{1, 2\}$.

Assume that D_j for some $j \in \{1, 2\}$ includes $\overline{y_i}$. Hence, $v(D_{3-j}) = \{v(y_{3-i})\}$. Let $y_i H$ be the other clause including y_i. Clearly, H cannot be empty, since otherwise R-Rule 3 applies, contradicting that F is reduced by the reduction rules. Moreover, H does not include any of $v(x)$, $v(y_i)$, and $v(y_3)$. If $v(H) = \{v(y_{3-i})\}$, i.e., H includes only a literal of $v(y_{3-i})$, then xy_1, xy_2, $\overline{x}y_3$, $\overline{y_3}D_j$, $\overline{y_3}D_{3-j}$, and $y_i H$ are exactly the clauses including the variables $v(x)$, $v(y_1)$, $v(y_2)$, $v(y_3)$, where both D_{3-j} and H include only $v(y_{3-i})$. In this case, assigning $x = y_3 = 0$ and $y_1 = y_2 = 1$ satisfies all these clauses, and hence this is an optimal assignment. Therefore, in this case we do not need to branch at all. So, we assume now that there is a literal y in H such that $v(y) \notin \{v(x), v(y_1), v(y_2), v(y_3)\}$. When $x = 0$, $\overline{x}y_3$ is satisfied and removed. Due to Lemma 5, there is an optimal assignment satisfying all clauses including the literal y_1 or y_2. As xy_1, xy_2 are two clauses in F and $x = 0$, we assign $y_1 = y_2 = 1$, making the clauses including y_1 or y_2 satisfied and removed. Then, due to R-Rule 3, we assign $y_3 = 0$, making the clause $\overline{y_3}D_j$ satisfied and removed. Furthermore, the variable $v(y)$ is of degree 2 now, and can be reduced by the reduction rules. In summary, the variables $v(x)$, $v(y_1)$, $v(y_2)$, $v(y_3)$, and $v(y)$ are reduced. Hence, n decreases by at least 5.

Case 2.3. Each D_i, $i \in \{1, 2\}$, consists of a single literal y_j for some $j \in \{1, 2\}$.

Note first that Case 2.3 occurs only when none of Cases 2.1 and 2.2 occurs. In this case, R-Rule 6 applies, contradicting that F is reduced.

In summary, in this case we have a branching superior to a $\langle 4, 5\rangle$-k-branching (resp. $\langle 3, 5\rangle$-n-branching) as stated in the lemma. □

Lemma 11 *Let $i \in \{1, 2\}$. If $|C_i| \geq 2$ and $|C_{3-i}| = |C_3| = 1$, branching on x leads to a branching superior to a $\langle 6, 3\rangle$-k-branching (resp. $\langle 5, 3\rangle$-n-branching).*

Proof Let y_1, y_2 be any two arbitrary literals in C_i, y_3 the literal in C_{3-i}, and y_4 the literal in C_3. So, $xC_{3-i} = xy_3$ and $\overline{x}C_3 = \overline{x}y_4$. We distinguish between the following cases.

Case 1. y_4 is a (2, 1)-literal.

In the branching $x = 1$, xC_i and xy_3 are satisfied and removed. Accordingly, we decrease k by 2. Due to Lemma 6, applying R-Rules 1–4 exhaustively reduces the variables $v(y_1)$, $v(y_2)$, $v(y_3)$, and decreases k by at least 3. In addition, due to Lemma 5, there is an optimal assignment satisfying $\overline{x}y_4$. Hence, we assign $y_4 = 1$. As a result, $\overline{x}y_4$ is satisfied and k is accordingly decreased by 1. Therefore, in total k decreases by at least $2+3+1 = 6$ and n decreases by at least 5 (the variables $v(x)$, $v(y_1)$, $v(y_2)$, $v(y_3)$, and $v(y_4)$ are reduced).

In the branching $x = 0$, due to Lemma 5 there is an optimal assignment satisfying xy_3. Hence, we assign $y_3 = 1$, remove all satisfied clauses including $\overline{x}y_4$ and xy_3, and decrease k by at least 2 accordingly. After this, $v(y_4)$ is of degree 1 or 2, and application of R-Rules 3 or 4 reduces $v(y_4)$ and decreases k by at least 1. In total, k decreases by at least $2+1 = 3$ and n decreases by at least 3 (the variables $v(x)$, $v(y_3)$, and $v(y_4)$ are reduced).

In summary, in this case, we have a branching superior to a $\langle 6, 3\rangle$-k-branching (resp. $\langle 5, 3\rangle$-n-branching) as stated in the lemma.

Case 2. y_4 is a $(1, 2)$-literal.

Let $\overline{y_4}D_1$ and $\overline{y_4}D_2$ be the two clauses including the literal $\overline{y_4}$. It must be that $\min\{|D_1|, |D_2|\} \geq 1$, since otherwise R-Rule 3 is applicable to F, a contradiction. When branching $x = 1$, we remove the satisfied clauses xy_1y_2 and xy_3, and accordingly decrease k by 2. Due to Lemma 6, exhaustive applications of R-Rules 1–4 reduce the variables $v(y_1)$, $v(y_2)$, $v(y_3)$, and decrease k by at least 3. In total, k decreases by at least $2+3 = 5$ and n decreases by at least 4 (the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_3)$ are reduced). When branching $x = 0$, we first assign $y_3 = 1$ similar to Case 1. Then, all satisfied clauses including $\overline{x}y_4$ and xy_3 are removed. So, k is decreased by at least 2. Then, $\overline{y_4}$ is either a $(2, 0)$-literal or a $(1, 0)$-literal (when D_1 or D_2 includes $v(y_3)$). So, application of R-Rule 3 reduces $v(y_4)$ and decreases k by at least 1. As F is reduced by the reduction rules, D_1 and D_2 do not include $v(y_4)$. In addition, as F is linear, D_1 and D_2 do not include $v(x)$, and at most one of them includes $v(y_3)$. This implies that at least one of D_1 and D_2 includes a variable $v(y_5) \notin \{v(x), v(y_3), v(y_4)\}$. After removing $\overline{y_4}D_1$ and $\overline{y_4}D_2$, the variable $v(y_5)$ can be reduced by R-Rules 3 or 4, and k can be decreased by at least 1 accordingly. In total, k decreases by at least $2+1+1 = 4$ and n decreases by at least 4 (the variables $v(x)$, $v(y_3)$, $v(y_4)$, and $v(y_5)$ are reduced).

In summary, in this case we have a branching superior to a $\langle 5, 4\rangle$-k-branching (resp. $\langle 4, 4\rangle$-n-branching), which is superior to a $\langle 6, 3\rangle$-k-branching (resp. $\langle 5, 3\rangle$-n-branching) as stated in the lemma. □

Lemma 12 *Let $i \in \{1, 2\}$. If $|C_i| \geq 3$, $|C_{3-i}| \geq 2$, and $|C_3| = 1$, branching on x leads to a branching superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching).*

Proof Let y_1, y_2, and y_3 be any three arbitrary literals in C_i, y_4 and y_5 be any two arbitrary literals in C_{3-i}, and y_6 be the literal in C_3. So, $\overline{x}C_3 = \overline{x}y_6$. We distinguish between the following two cases.

Case 1. y_6 is a $(2, 1)$-literal.

Consider first the branching $x = 1$. The clauses xC_1 and xC_2 are satisfied and removed. Hence, we decrease k by 2. Then, $v(y_1)$, …, $v(y_5)$ are of degree 2. Then, exhaustive applications of R-Rules 1–4 reduce the variables $v(y_1)$, …, $v(y_5)$ and decrease k by at least 5, due to Lemma 6. Moreover, due to Lemma 5, we assign to y_6 the value 1, leading at least one more clause (i.e., $\overline{x}y_6$) to be satisfied and k being decreased by at least 1. In total, k decreases by at least $2 + 5 + 1 = 8$ and n decreases by at least 7 (the variables $v(x)$, $v(y_1)$, …, $v(y_6)$ are reduced). Consider now the branching $x = 0$. The clause $\overline{x}y_6$ is satisfied and removed, and k is decreased by 1 accordingly. Then, applying R-Rule 3 or R-Rule 4 reduces $v(y_6)$ and decreases k by at least 1. In total, k decreases by at least $1 + 1 = 2$ and n decreases by at least 2 (the variables $v(x)$ and $v(y_6)$ are reduced).

In summary, in this case we have a branching superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching) as stated in the lemma.

Case 2. y_6 is a $(1, 2)$-literal.

Let $\overline{y_6}D_1$ and $\overline{y_6}D_2$ be the two clauses including $\overline{y_6}$. Due to R-Rule 3, we have that $\min\{|D_1|, |D_2|\} \geq 1$. For the branching $x = 1$, clauses xC_1 and xC_2 are satisfied. Hence, we remove them and decrease k by 2. Due to Lemma 6, applying R-Rules 1–4 exhaustively reduces the variables $v(y_1)$, $v(y_2)$, ..., $v(y_5)$ and decreases k by at least 5. In total, k decreases by at least $2 + 5 = 7$ and n decreases by at least 6 (the variables $v(x)$, $v(y_1)$, ..., $v(y_5)$ are reduced). For the branching $x = 0$, we remove the satisfied clause $\overline{x}y_6$ and decrease k by 1. Then, application of R-Rule 3 reduces the two clauses $\overline{y}_6D_1$ and $\overline{y}_6D_2$, and decreases k by 2. Let z and z' be any arbitrary literals in D_1 and D_2, respectively. Both $v(z)$ and $v(z')$ are of degree 2. Moreover, due to Lemma 3, z and z' cannot be the literals of the same variable, and none of them can be a literal of the variable $v(x)$. As F is reduced by the reduction rules, z and z' cannot be the literals of $v(y_6)$ too. Therefore, exhaustive applications of R-Rules 1–4 reduce $v(z)$, $v(z')$ and decrease k by at least 2. In total, k decreases by at least $1 + 2 + 2 = 5$ and n decreases by at least 4 (the variables $v(x)$, $v(y_6)$, $v(z)$, and $v(z')$ are reduced).

In summary, in this case we have a branching superior to a $\langle 7, 5\rangle$-k-branching (resp. $\langle 6, 4\rangle$-n-branching), which is superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching) as stated in the lemma. □

Now we consider the last branching case, i.e., the two clauses including the literal x are of size 3, and the clause including the literal $\overline{x}$ is of size 2. We note that we use this branching case only when none of the above branching cases is applicable. Hence, we assume now that there are no $(2, 1)$-literals satisfying conditions in Lemmas 7–12.

Lemma 13 *If $|C_1| = |C_2| = 2$ and $|C_3| = 1$, branching on x leads to a branching superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching).*

Proof Let y_1 and y_2 be the two literals in C_1, y_3 and y_4 be the two literals in C_2, and y_5 be the literal in C_3. We distinguish between the following cases.
Case 1. y_5 is a $(1, 2)$-literal.

Let $\overline{y_5}C$ and $\overline{y_5}C'$ be the two clauses including $\overline{y_5}$. Due to R-Rule 3, it must be that $\min\{|C|, |C'|\} \geq 1$. Let y and y' be any arbitrary literals from C and C', respectively. When branching $x = 1$, clauses xC_1 and xC_2 are satisfied. Hence, we remove them and accordingly decrease k by 2. Then, $v(y_1)$, $v(y_2)$, $v(y_3)$, and $v(y_4)$ are of degree 2. Due to Lemmas 3 and 6, exhaustive applications of R-Rules 1–4 reduce these variables and decrease k by at least 4. In total, k decreases by at least $2+4 = 6$ and n decreases by at least 5 (the variables $v(x)$, $v(y_1)$, $v(y_2)$, $v(y_3)$, and $v(y_4)$ are reduced). Consider the branching $x = 0$. Clearly, we can remove the satisfied clause $\overline{x}y_5$ and decrease k by 1. Then, we apply R-Rule 3 to reduce $\overline{y_5}C$ and $\overline{y_5}C'$, and decrease k by 2. After this, $v(y)$ and $v(y')$ are of degree 2. Due to Lemma 3, it holds that $v(y), v(y') \notin \{v(x), v(y_5)\}$ and $v(y) \neq v(y')$. Then, exhaustive applications of R-Rules 1–4 reduce these variables and decrease k by at least 2. As a result, k decreases by at least $1 + 2 + 2 = 5$ and n decreases by at least 4 in total (variables $v(x)$, $v(y_5)$, $v(y)$, and $v(y')$ are reduced).

In summary, in this case we have a branching superior to a $\langle 6, 5\rangle$-k-branching (resp. $\langle 5, 4\rangle$-n-branching), which is superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching) as stated in the lemma.

Case 2. y_5 is a $(2, 1)$-literal.

Let $y_5 D_1$ and $\overline{y_5} D_2$ be the other two clauses including $v(y_5)$. Observe that D_2 must be empty, since otherwise there is a $(2, 1)$-literal (e.g., y_5) satisfying the conditions of one of Lemmas 8–11, a contradiction. Moreover, D_1 is not empty, since otherwise R-Rule 3 is applicable, a contradiction too. Furthermore, y_i for each $1 \le i \le 4$ is a $(2, 1)$-literal, since otherwise there is a $(2, 1)$-literal (e.g., $\overline{y_i}$) satisfying the conditions of one of Lemmas 7–9, a contradiction. Finally, as F is linear and reduced by the reduction rules, D_1 does not include $v(x)$ and $v(y_5)$. We proceed the proof by distinguishing between the following cases.

Case 2.1. D_1 includes a variable $v(y) \notin \{v(y_1), \ldots, v(y_4)\}$.

In the branching $x = 1$, according to Lemma 5 there is an optimal assignment satisfying the two clauses including the literal y_5, implying that $y_5 = 1$ in this assignment. Hence, we assign $y_5 = 1$. As a result, the clauses xy_1y_2, xy_3y_4, $\overline{x}y_5$, and y_5D_1 are satisfied and accordingly we decrease k by 4. Then, the variables $v(y_1)$, $v(y_2)$, $v(y_3)$, and $v(y_4)$ are of degree 1 or 2. We can apply R-Rules 1–4 to reduce these variables and decrease k by at least 4. In total, k decreases by at least $4 + 4 = 8$. Note that after removing y_5D_1, the variable $v(y)$ can be also reduced by R-Rules 1–4. Therefore, n decreases by at least 7 (the variables $v(x)$, $v(y_1)$, ..., $v(y_5)$, and $v(y)$ are all reduced). When branching $x = 0$, $\overline{x}y_5$ is satisfied, y_5 is further reduced by R-Rules 3–4, leading to both n and k being decreased by at least 2.

Case 2.2. D_1 includes exactly one of $v(y_1)$, ..., $v(y_4)$ and $|D_1| = 1$.

Without loss of generality, assume that D_1 includes $v(y_1)$ (other cases can be dealt with similarly). If the literal y_1 is in D_1, then the clause including the literal $\overline{y_1}$ must be a unit clause, since otherwise there is a $(2, 1)$-literal (e.g., y_1) satisfying the conditions of one of Lemmas 8 and 11, a contradiction. However, in this case R-Rule 5 applies to y_5y_1, $\overline{y_5}$, and $\overline{y_1}$ in F, contradicting that F is reduced by the reduction rules. So it must be that D_1 includes the literal $\overline{y_1}$. Let y_1C' be the other clause including the literal y_1. It must be that $|C'| = 2$, since otherwise there is a $(2, 1)$-literal (e.g., y_1) satisfying the conditions of one of Lemmas 11 and 12, a contradiction. Due to Lemma 3 and the fact that F is reduced by the reduction rules, the variables $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_5)$ cannot be included in C'. Moreover, at most one of $v(y_3)$ and $v(y_4)$ can be included in C'. Therefore, C' includes a literal z such that $v(z) \notin \{v(x), v(y_1), \ldots, v(y_5)\}$. Let z' be the other literal in C'. So, we have the following 6 clauses:

$$xy_1y_2,\ xy_3y_4,\ \overline{x}y_5,\ y_5\overline{y_1},\ \overline{y_5},\ y_1zz'.$$

Similar to the above argument, z cannot be a literal of $v(x)$, $v(y_1)$, $v(y_2)$, and $v(y_5)$. If $x = 1$, then xy_1y_2 and xy_3y_4 are satisfied and removed. Accordingly, we decrease k by 2. Then, we apply R-Rule 3 which assigns $y_5 = 1$, leading to the two clauses including y_5 being satisfied, y_5 being reduced, and k being decreased by 2. Then, y_1zz' is the only clause including y_1. Hence, we assign $y_1 = 1$ to make y_1zz' satisfied.

Meanwhile, we decrease k by 1. After this, $v(z)$ and $v(z')$ are of degree 1 or 2. As F is linear, except the clause y_1zz' there is no other clause including both $v(z)$ and $v(z')$. As a result, applying R-Rules 3 or 4 to clauses including $v(z)$ or $v(z')$ reduces these two variables and decreases k by at least 2. Now, observe that $v(y_2)$ is of degree 1 or 2. Hence, we can apply R-Rules 3 or 4 to reduce $v(y_2)$ and decrease k by 1. In summary, k decreases by at least $2+2+1+2+1 = 8$. We further apply R-Rules 1–4 to reduce $v(y_3)$ and $v(y_4)$. Therefore, the variables $v(x)$, $v(y_1)$, ..., $v(y_5)$, $v(z)$, and $v(z')$ are reduced. Though it may be that $v(z') \in \{v(y_3), v(y_4)\}$, n decreases by at least 7. When branching with $x = 0$, $\overline{x}y_5$ is satisfied and $v(x)$ is reduced. Then, R-Rule 4 can be applied to reduce $v(y_5)$. In summary, in this case both k and n decrease by at least 2.

Case 2.3. D_1 includes exactly two of $\{v(y_1), \dots, v(y_4)\}$ and $|D_1| = 2$.

Due to Lemma 3, D_1 can include at most one of $v(y_1)$, $v(y_2)$, and at most one of $\{v(y_3), v(y_4)\}$. Without loss of generality, assume that $v(y_1), v(y_3) \in D_1$ (other cases are dealt with symmetrically). Notice that none of $\overline{y_1}$ and $\overline{y_3}$ can be in D_1, since otherwise there is a literal (y_1 or y_3) satisfying the conditions in one of Lemmas 7–8, a contradiction. As F is linear, no clause except $y_5y_1y_3$ includes both $v(y_1)$ and $v(y_3)$. Let $\overline{y_1}C'$ and $\overline{y_3}C''$ be the two clauses including the literals $\overline{y_1}$ and $\overline{y_3}$, respectively. So, we have the following 7 clauses:

$$xy_1y_2,\ xy_3y_4,\ \overline{x}y_5,\ y_5y_1y_3,\ \overline{y_5},\ \overline{y_1}C',\ \overline{y_3}C''.$$

If $\max\{|C'|, |C''|\} \geq 2$, then there are literals (e.g., y_1 or y_3) satisfying the conditions of Lemma 7, a contradiction. Therefore, assume that $|C'| \leq 1$ and $|C''| \leq 1$. We proceed the proof by distinguishing between the following cases.

Case 2.3.1. $C' = C'' = \emptyset$.

When $x = 1$, the clauses xy_1y_2 and xy_3y_4 are satisfied and removed. Accordingly, k is decreased by 2. Then, R-Rule 3 assigns $y_5 = 1$, making the clauses y_5 and $y_5y_1y_3$ satisfied. Moreover, k is decreased by 2. Then, we can directly assign $y_1 = y_3 = 0$ to make the two clauses $\overline{y_1}$ and $\overline{y_3}$ satisfied, and decrease k by 2. In addition, $v(y_2)$ and $v(y_4)$ are of degree 2 now. Applying R-Rules 1–4 reduces these variables and decreases k by at least 2. In summary, k decreases by at least 8 and n decreases by at least 6 (the variables $v(x)$, $v(y_1)$, ..., $v(y_5)$ are all reduced). In the branching $x = 0$, the clause $\overline{x}y_5$ is satisfied and removed. Accordingly, k is decreased by 1. Then, R-Rule 4 applies to $y_5y_1y_3$ and $\overline{y_5}$, which reduces $v(y_5)$ and decreases k by 1. Now we have the clauses y_1y_2, y_3y_4, y_1y_3, $\overline{y_1}$, and $\overline{y_3}$. Applying R-Rule 5 replaces y_1y_3, $\overline{y_1}$, and $\overline{y_3}$ with $\overline{y_1}\,\overline{y_3}$, and decreases k by 1. Then, $v(y_1)$ and $v(y_3)$ can be reduced by R-Rule 4. In total, k decreases by at least 2 and n decreases by at least 4 (the variables $v(x)$, $v(y_1)$, $v(y_3)$, $v(y_5)$ are reduced). In summary, we have a branching superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 6, 4\rangle$ -n-branching), which is superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching).

Case 2.3.2. C' or C'' includes a variable $v(y)$ not in $\{v(y_1), \dots, v(y_5)\}$.

Assume that $y \in C'$ (the case that $y \in C''$ can be dealt with similarly). Hence, we have the clauses

$$xy_1y_2,\ xy_3y_4,\ \overline{x}y_5,\ y_5y_1y_3,\ \overline{y_5},\ \overline{y_1}y,\ \overline{y_3}C''.$$

In the branching $x = 1$, the two clauses xy_1y_2 and xy_3y_4 are satisfied and removed. In addition, k is accordingly decreased by 2. Then, due to R-Rule 3, we assign $y_5 = 1$, making the two clauses including y_5 satisfied and k decreased by 2. Then, $\overline{y_1}y$ (resp. $\overline{y_3}C'$) is the only clause including $\overline{y_1}$ (resp. $\overline{y_3}$). Due to R-Rule 3 again, we assign $y_1 = y_3 = 0$, making $\overline{y_1}y$ and $\overline{y_3}C'$ satisfied and k further decreased by 2. Then, $v(y_2)$, $v(y_4)$, and $v(y)$ become of degree 1 or 2. Exhaustively applying R-Rules 1–4 reduces these variables and decreases k by at least 2. In total, k decreases by at least 8. As $v(x)$, $v(y_1)$, ..., $v(y_5)$, $v(y)$ are reduced, n decreases by at least 7. In the branching $x = 0$, $\overline{x}y_5$ is satisfied. Then, by R-Rule 4 we assign $y_5 = 1$. Clearly, this leads to at least two clauses being satisfied and two variables being reduced.

Case 2.3.3. C' or C'' includes a negative literal of $v(\{y_1, \ldots, y_5\})$.

Assume that C' includes a negative literal $\overline{y_i}$, $1 \leq i \leq 5$ (the case that C'' includes such a literal can be dealt with similarly). As F is linear and reduced by the reduction rules, i cannot be 1, 2, 3, and 5. Hence, $i = 4$. Let y_4D be the other clause including y_4. Observe that $|D| = 2$ must hold, since otherwise there is a literal (e.g., y_4) satisfying the conditions in one of Lemmas 11 or 12, a contradiction. Moreover, D does not include any of $v(x)$, $v(y_1)$, $v(y_3)$, $v(y_4)$, and $v(y_5)$. This means that D includes a literal y such that $v(y) \notin \{v(x), v(y_1), \ldots, v(y_5)\}$. In the branching $x = 1$, xy_1y_2 and xy_3y_4 are satisfied and removed. Accordingly, k is reduced by 2. Then, due to R-Rule 3, we assign $y_5 = 1$, making the two clauses including y_5 satisfied and removed, and k decreased by 2. Then, $v(y_1)$ and $v(y_3)$ become of degree 1, and can be reduced by R-Rule 3. Accordingly, k decreases by 2. Then, $v(y_4)$ is of degree 1, and hence R-Rule 3 can be applied to reduce it and decrease k by 1. Finally, $v(y_2)$ and $v(y)$ become of degree 1 or 2 and can be reduced by the reduction rules, leading to k being decreased by at least 2. In total, k decreases by at least 8 and n decreases by at least 7. The branching for $x = 0$ is analyzed similarly to Case 2.3.2.

Case 2.3.4. C' or C'' includes a positive literal of $v(y_1)$, ..., $v(y_5)$.

Assume that C' includes a literal y_i for some $i \in \{1, 2, 3, 4, 5\}$ (the case that C'' includes such a literal can be dealt with similarly). Similar to Case 2.3.3, it must be that $i = 4$.

If $|C''| = 1$, then $C'' = y_2$. Observe that the clause including $\overline{y_4}$ must be a unit clause, since otherwise there is a literal satisfying the conditions in one of Lemmas 8 or 11, a contradiction. The similar argument holds for $\overline{y_2}$. Hence, we have the following clauses:

$$xy_1y_2,\ xy_3y_4,\ \overline{x}y_5,\ y_5y_1y_3,\ \overline{y_5},\ \overline{y_1}y_4,\ \overline{y_3}y_2,\ \overline{y_2},\ \overline{y_4}.$$

These are all clauses including the variables $v(x)$, $v(y_1)$, ..., $v(y_5)$, and it is easy to check that there is an optimal assignment under which $x = y_5 = 1$ and $y_1 = y_2 = y_3 = y_4 = 0$. So, in this case we do not need to branch at all.

If C'' is empty, similar to the above arguments, we can conclude that there is a unit clause $\overline{y_4}$. However, R-Rule 7 applies, contradicting that F is reduced by the reduction rules.

In summary, in each case we have a branching which is superior to a $\langle 8, 2\rangle$-k-branching (resp. $\langle 7, 2\rangle$-n-branching) as stated in the lemma. $\square$

Armed with the above lemmas, we are able to present the main result of this paper.

Theorem 1 *The* $(n, 3)$- MAXSAT *problem can be solved in times* $O^*(1.175^k)$ *and* $O^*(1.194^n)$*, respectively.*

Proof Based on the reduction rules and branching rules studied in this section, we develop an algorithm for the $(n, 3)$-MaxSAT problem as follows. The algorithm first exhaustively runs the reduction rules and the polynomial-time algorithm stated in Lemma 3, until the CNF formula is linear and none of the reduction rules is applicable. This procedure terminates in polynomial time. Then, if the condition in Lemma 4 is satisfied, we solve the instance in polynomial time. Otherwise, there exist (2, 1)-literals x such that there are no unit clauses x and $\overline{x}$. We branch on such literals x. Table 3 summarizes the branching vectors and branching roots of all cases, with respect to the sizes of the clauses including $v(x)$. In each branching, we iteratively run the above procedure. Notice that we perform the branching case described in Lemma 13 only when none of the branching cases described in Lemmas 7–12 applies. From Table 3 we can conclude that the algorithm solves the $(n, 3)$-MaxSAT problem in $O^*(1.175^k)$ and $O^*(1.194^n)$ times.

The correctness of the algorithm directly follows from the soundness of the reduction rules [see Lemmas 1–2 and proofs in Bansal and Raman (1999), Bonet et al. (2007), Chen and Kanj (2004) and Niedermeier and Rossmanith (2000)], and the fact that the branchings cover all possible cases. □

5 Conclusion

In this paper, we have derived a branching algorithm for the $(n, 3)$-MaxSAT problem whose running time can be bounded by $O^*(1.175^k)$ and $O^*(1.194^n)$, where n is the number of variables in the given CNF formula and k is the lower bound of the number of clauses desired to be satisfied. Our algorithm largely improves the previous branching algorithm in Xu et al. (2016) which runs in times $O^*(1.194^k)$ and $O^*(1.237^n)$.

A direction for future research would be to further improve the running times of the algorithm. We can see from Table 3 that improving the running time in terms of n requires improvement in several cases, while improving the running time in terms of k only needs improvements over the branching cases described in Lemmas 12 and 13.

References

Argelich J, Manyà F (2006) Exact Max-SAT solvers for over-constrained problems. J Heuristics 12(4–5):375–392

Aspvall B, Plass MF, Tarjan RE (1979) A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf Process Lett 8(3):121–123

Bansal N, Raman V (1999) Upper bounds for MaxSAT: further improved. In: ISAAC, pp 247–258

Berg J, Hyttinen A, Järvisalo M (2015) Applications of MaxSAT in data analysis. In: Pragmatics of SAT workshop

Bliznets I, Golovnev A (2012) A new algorithm for parameterized MAX-SAT. In: IPEC, pp 37–48

Bliznets IA (2013) A new upper bound for $(n, 3)$-MAX-SAT. J Math Sci 188(1):1–6

Bonet ML, Levy J, Manyà F (2007) Resolution for Max-SAT. Artif Intell 171(8–9):606–618

Calabro C, Impagliazzo R, Paturi R (2006) A duality between clause width and clause density for SAT. In: CCC, pp 252–260

Chen J, Kanj IA (2004) Improved exact algorithms for Max-Sat. Discrete Appl Math 142(1–3):17–27

Chen J, Kanj IA, Xia G (2010) Improved upper bounds for vertex cover. Theor Comput Sci 411(40–42):3736–3756

Cook SA (1971) The complexity of theorem-proving procedures. In: STOC, pp 151–158

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Lower bounds based on the exponential-time hypothesis. Springer, Berlin, pp 467–521

Fomin FV, Kratsch D (2010) Exact exponential algorithms, chapter 2. Texts in theoretical computer science An EATCS series. Springer, Berlin, pp 13–30

Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, New York

Goemans MX, Williamson DP (1994) New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM J Discrete Math 7(4):656–666

Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J ACM 42(6):1115–1145

Gu J (1994) Global optimization for satisfiability (SAT) problem. IEEE Trans Knowl Data Eng 6(3):361–381

Hirsch EA (2000) New worst-case upper bounds for SAT. J Autom Reason 24(4):397–420

Hirsch EA, Kojevnikov A (2005) Unitwalk: a new SAT solver that uses local search guided by unit clause elimination. Ann Math Artif Intell 43(1):91–111

Hochbaum D (1997) Approximation algorithms for NP-hard problems. PWS Publishing Company, Boston

Hutter F, Lindauer M, Balint A, Bayless S, Hoos H, Leyton-Brown K (2017) The configurable SAT solver challenge (CSSC). Artif Intell 243:1–25

Impagliazzo R, Paturi R (2001) On the complexity of k-SAT. J Comput Syst Sci 62(2):367–375

Karloff HJ, Zwick U (1997) A 7/8-approximation algorithm for MAX 3SAT? In: FOCS, pp 406–415

Kulikov AS (2005) Automated generation of simplification rules for SAT and MAXSAT. In: SAT, pp 430–436

Li W, Xu C, Wang J, Yang Y (2017) An improved branching algorithm for $(n, 3)$-MaxSAT based on refined observations. In: COCOA, pp 94–108

Lokshtanov D (2009) New methods in parameterized algorithms and complexity. Ph.D. thesis, University of Bergen

Luo C, Cai S, Su K, Huang W (2017) CCEHC: an efficient local search algorithm for weighted partial maximum satisfiability. Artif Intell 243:26–44

Niedermeier R, Rossmanith P (2000) New upper bounds for maximum satisfiability. J Algorithms 36(1):63–88

Patrascu M, Williams R (2010) On the possibility of faster SAT algorithms. In: SODA, pp 1065–1075

Poloczek M, Schnitger G, Williamson DP, van Zuylen A (2017) Greedy algorithms for the maximum satisfiability problem: simple algorithms and inapproximability bounds. SIAM J Comput 46(3):1029–1061

Raman V, Ravikumar B, Rao SS (1998) A simplified NP-complete MAXSAT problem. Inf Process Lett 65(1):1–6

Saikko P, Malone B, Järvisalo M (2015) MaxSAT-based cutting planes for learning graphical models. In: CPAIOR, pp 347–356

Selman B, Mitchell DG, Levesque HJ (1996) Generating hard satisfiability problems. Artif Intell 81(1–2):17–29

Shen H, Zhang H (2005) Improving exact algorithms for MAX-2-SAT. Ann Math Artif Intell 44(4):419–436

Xiao M, Nagamochi H (2016) An exact algorithm for maximum independent set in degree-5 graphs. Discrete Appl Math 199:137–155

Xu C, Chen J, Wang J (2016) Resolution and linear CNF formulas: improved $(n, 3)$-MaxSAT algorithms. Theor Comput Sci (to appear)