A Clause Tableau Calculus for MaxSAT *

Chu-Min Li
Huazhong Univ. of Science and Technology
MIS, Université de Picardie Jules Verne

Felip Manyà and **Joan Ramon Soler**
Artificial Intelligence Research Inst. (IIIA)
Spanish National Research Council (CSIC)

Abstract

We define a clause tableau calculus for MaxSAT, prove its soundness and completeness, and describe a tableau-based algorithm for MaxSAT. Given a multiset of clauses ϕ, the algorithm computes both the minimum number of clauses that can be falsified in ϕ, and an optimal assignment. We also describe how the algorithm can be extended to solve weighted MaxSAT and weighted partial MaxSAT.

1 Introduction

There has been tremendous progress in theoretical and applied aspects of the MaxSAT problem over the last decade. As a result, there are now a number of competitive solvers that are able to solve challenging optimization problems in different areas (see e.g. [Abramé and Habet, 2015; Ansótegui *et al.*, 2013; 2016a; 2016b; Li and Manyà, 2009; Martins *et al.*, 2014; Morgado *et al.*, 2013; Narodytska and Bacchus, 2014] and the references therein for previous and related work).

One lesson learned during that time is that the inference rules applied in SAT solving usually cannot be applied in MaxSAT, because such rules preserve satisfiability but do not preserve the number of falsified clauses. This implies that successful SAT solving techniques such as unit propagation are unsound in MaxSAT.

The most relevant studies conducted to date about inference rules in MaxSAT analyze how an extended resolution rule, or refinements of this rule, can be applied to solve MaxSAT [Abramé and Habet, 2014a; Bonet *et al.*, 2007; Heras and Larrosa, 2006; Larrosa *et al.*, 2008; Li *et al.*, 2007].

MaxSAT resolution replaces two parent clauses ($x \vee a_1 \vee \ldots \vee a_s$ and $\overline{x} \vee b_1 \vee \ldots \vee b_t$) with their resolvent ($a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_t$) plus $s + t$ compensation clauses that are not needed in SAT ($x \vee a_1 \vee \ldots \vee a_s \vee \overline{b}_1, \ldots, x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_{t-1} \vee \overline{b}_t, \overline{x} \vee b_1 \vee \ldots \vee b_t \vee \overline{a}_1, \ldots, \overline{x} \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \ldots \vee a_{s-1} \vee \overline{a}_s$). Compensation clauses ensure that the number of falsified clauses is preserved but complicate the application of the rule. A nice feature of MaxSAT resolution is its completeness [Bonet *et al.*, 2006; 2007]: given a multiset of clauses whose minimum number of falsified clauses is m, the application of MaxSAT resolution, finitely many times and following a certain strategy, derives a multiset formed by exactly m empty clauses.

The present paper investigates how to extend the clause tableau method for SAT [D'Agostino, 1999; Hähnle, 2001] to solve MaxSAT. The contributions of our work can be summarized as follows:

- A clause tableau calculus for MaxSAT that is sound and complete in the following sense: the minimum number of clauses that can be falsified in a multiset of clauses ϕ is m iff the minimum number of empty clauses, among the branches of a clause tableau for ϕ, that can be derived by applying the tableau rules of the calculus until saturation is m.[1]
- A tableau-based MaxSAT algorithm that computes both the minimum number of clauses that can be falsified in ϕ, and an optimal assignment.
- Description of how the algorithm can be extended to solve weighted MaxSAT and weighted partial MaxSAT.

It is the first time, to the best of our knowledge, that a tableau calculus has been employed in combinatorial optimization and, in particular, to solve Boolean optimization problems. Moreover, the proposed calculus offers an inference system that is simpler and more intuitive than MaxSAT resolution: no compensation clauses are needed, and all the derived clauses are either empty or unit.

An advantage of the tableau method for SAT is that it can deal directly with formulas that are not in conjunctive normal form (CNF). Tableaux do not need to apply CNF conversion, avoiding so the space complexity of some conversion CNF algorithms. It remains an open question how to extend the contributions of the present paper to solve MaxSAT for non-CNF formulas.

The paper is organized as follows: Section 2 gives basic concepts about SAT and MaxSAT. Section 3 describes how clause tableaux can be used to solve SAT. Section 4 defines

*Research partially supported by the National Natural Science Foundation of China (Grant no. 61472147), the Generalitat de Catalunya grant AGAUR 2014-SGR-118, and the Ministerio de Economía y Competividad project RASO TIN2015-71799-C2-1-P. The second author was supported by Mobility Grant PRX16/00215 of the Ministerio de Educación, Cultura y Deporte.

[1] By saturation we mean that all the possible applications of inference rules were performed.

an original clause tableau calculus for MaxSAT, proves its soundness and completeness, and describes an exact MaxSAT algorithm. Section 5 shows how the proposed algorithm can be extended to solve weighted MaxSAT and weighted partial MaxSAT. Section 6 contains the concluding remarks, and points out future research directions.

2 Preliminaries

A literal is a propositional variable or a negated propositional variable. A clause is a disjunction of literals. A weighted clause is a pair (c, w), where c is a disjunction of literals and w, its weight, is a natural number or infinity. A clause is hard if its weight is infinity; otherwise it is soft. A weighted partial MaxSAT instance is a multiset of weighted clauses $\phi = \{(h_1, \infty), \ldots, (h_k, \infty), (c_1, w_1), \ldots, (c_m, w_m)\}$, where the first k clauses are hard and the last m clauses are soft. The total number of weighted clauses in ϕ is denoted by $|\phi|$. For simplicity, in what follows, we omit infinity weights, and write $\phi = \{h_1, \ldots, h_k, (c_1, w_1), \ldots, (c_m, w_m)\}$. A soft clause (c, w) is equivalent to having w copies of the clause $(c, 1)$, and $\{(c, w_1), (c, w_2)\}$ is equivalent to $(c, w_1 + w_2)$.

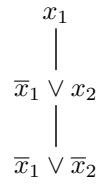
A truth assignment assigns to each propositional variable either 0 (false) or 1 (true). Weighted Partial MaxSAT, or WPMaxSAT, for an instance ϕ is the problem of finding an assignment that satisfies all the hard clauses and minimizes the sum of the weights of the falsified soft clauses; such an assignment is called optimal assignment. The Weighted MaxSAT problem, or WMaxSAT, is WPMaxSAT when there are no hard clauses. The Partial MaxSAT problem, or PMaxSAT, is WPMaxSAT when all the soft clauses have the same weight. The (Unweighted) MaxSAT problem is PMaxSAT when there are no hard clauses. The SAT problem, or SAT, is PMaxSAT when there are no soft clauses.

3 Clause Tableaux for SAT

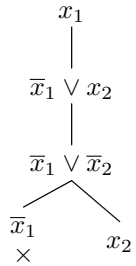
It is common to view the tableau method for solving SAT as a proof by case distinction that allows one to systematically generate subcases until elementary contradictions are reached. In the context of SAT, a clause tableau is a tree with a finite number of branches whose nodes are labelled with clauses, and a branch is a maximal path in a tree with a finite number of nodes. A branch is closed if there are two nodes labelled with complementary unit clauses; otherwise, it is open. A clause tableau is closed iff all its branches are closed.

Given a set of clauses $C = \{C_1, \ldots, C_m\}$, we create an initial tableau that has a single branch with m nodes, where each node is labelled with a clause of C. Then, we select an open branch B and a non-unit clause $l_1 \vee \ldots \vee l_r$ of C that has not yet been expanded in B, and append r nodes below B, labelling each node with a different unit clause from $\{l_1, \ldots, l_r\}$. This process of creating r new branches from B is known as the application of the *extension rule*. If there are two complementary unit clauses in a branch, we close it. This process continues until either all the branches are closed, or the application of the extension rule on a branch until saturation leaves it open. C is declared to be unsatisfiable in the first case, and satisfiable in the second case.

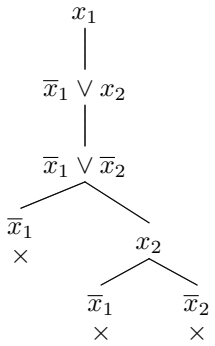
Example 1 *To determine the satisfiability of* $C = \{x_1, \overline{x}_1 \vee x_2, \overline{x}_1 \vee \overline{x}_2\}$ *with clause tableaux we start by creating the initial tableau* (T_0)*:*



We then expand the second node, and close he leftmost branch (it has the occurrence of both x_1 *and* $\overline{x}_1$*, and the closing is denoted by* $\times$ *in the tableau), obtaining the following clause tableau* (T_1)*:*



Finally, we expand the third node on the rightmost branch, and get a closed tableau (T_2) *that provides a clause tableau proof of the unsatisfiability of* C*:*



Formally, a clause tableau proof of the unsatisfiability of a set of clauses ϕ is a sequence of clause tableaux $T_0, \ldots, T_n$ such that T_0 is an initial tableau, T_n is a closed tableau, and T_i has been obtained by a single application of the extension rule on an open branch of T_{i-1} for $i = 1, \ldots, n$. In a proof of satisfiability, T_n must have some open branch after applying the extension rule on it until saturation. Besides, the literals occurring in the unit clauses of the open branch provide a satisfying assignment of ϕ.

It is common to say that T_n is a clause tableau proof because it collapses all the sequence of tableaux. In Example 1, T_0, T_1, T_2 is a clause tableau proof, although T_2 alone is also considered to be a clause tableau proof.

From a semantic perspective, given a set of clauses ϕ and a clause tableau T for ϕ, we have that ϕ is satisfiable iff there is a branch in T such that the conjunction of all its clauses is satisfiable (or alternatively, ϕ is unsatisfiable iff all the branches of T are unsatisfiable).

4 Clause Tableaux for MaxSAT

The tableau method for SAT is not valid for solving MaxSAT, among other reasons, because it stops the search in a branch once a contradiction is detected. Moreover, in the case of MaxSAT, we should pay attention to not use the same literal to detect different contradictions; for example, an exact MaxSAT method detects one contradiction in $\{x_1, \overline{x}_1, \overline{x}_1\}$, whereas it detects two contradictions in $\{x_1, x_1, \overline{x}_1, \overline{x}_1\}$ because x_1 can be used only once in the first case.

We define below a tableau method for MaxSAT that contemplates to find more than one contradiction in each branch, and that marks as used both the clauses that have already been expanded and the complementary unit clauses involved in a contradiction; in this way, the tableau method avoids to use more than once the same clause in a branch. Recall that we are now working with multisets of clauses. In the rest of the paper, unless otherwise stated, when we say clause tableau we mean clause tableau for MaxSAT.

Definition 1 *A clause tableau is a tree with a finite number of branches whose nodes are labelled with clauses that are declared to be either marked or unmarked in each branch. A branch is a maximal path in a tree, and we assume that branches have a finite number of nodes.*

Definition 2 *Let $\phi = \{C_1, \ldots, C_m\}$ be a multiset of clauses. A clause tableau for ϕ is constructed by a sequence of applications of the following rules:*

Initialize *A tree with a single branch with m nodes such that each node is labelled with a clause of ϕ is a clause tableau for ϕ. Such a tableau is called initial tableau and its clauses are declared to be unmarked.*

Extension *Given a clause tableau T for ϕ, a branch B of T, and a node of B labelled with an unmarked non-unit clause $l_1 \vee \cdots \vee l_r$, the tableau obtained by appending r nodes below B, labelling each node with a different unit clause from $\{l_1, \ldots, l_r\}$, is a clause tableau for ϕ. The clause $l_1 \vee \cdots \vee l_r$ is declared to be marked in the newly created branches, and the unit clause l_i, for $i = 1, \ldots, r$, is declared to be unmarked in the newly created branch in which it occurs.*

Contradiction *Given a clause tableau T for ϕ, a branch B of T, and two nodes of B labelled with two unmarked unit clauses l and $\bar{l}$, the tableau obtained by appending a node labelled with an empty clause below B is a clause tableau for ϕ. The empty clause is declared to be unmarked in the newly created branch, and the complementary unit clauses, l and $\bar{l}$, are declared to be marked.*

Definition 3 *Let T be a clause tableau for a multiset of clauses ϕ, and let B be a branch of T. The branch B is saturated when all its unmarked clauses are empty and unit clauses, and the contradiction rule cannot be further applied on B. The tableau T is saturated iff all its branches are saturated. The cost of a saturated branch is the number of empty clauses in it. The cost of a saturated tableau is the minimum cost among all its branches.*

The notion of saturation is crucial in MaxSAT because it indicates that the application of inference rules has been completed. As we show below, the minimum number of clauses that can be falsified in a multiset of clauses ϕ is k iff the cost of a saturated tableau for ϕ is k. So, the systematic construction of a saturated clause tableau for ϕ provides an exact method for MaxSAT, and each saturated tableau is a proof.

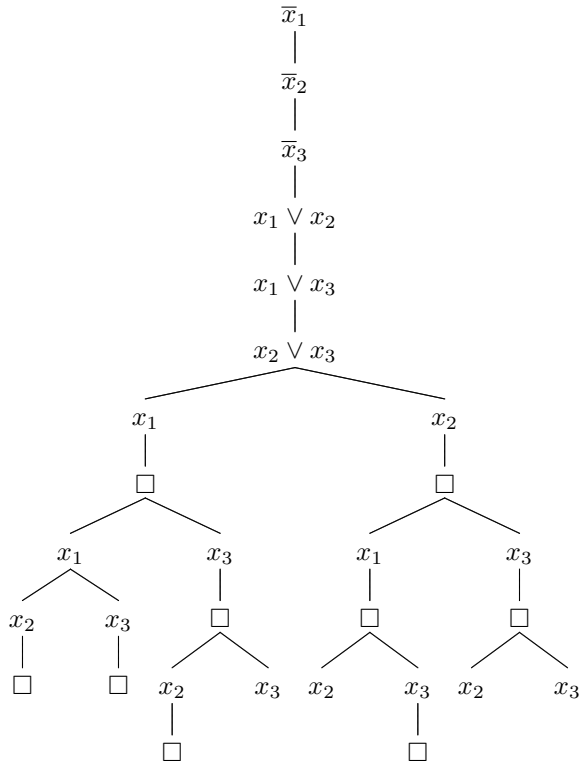


Figure 1: A saturated clause tableau for $\phi = \{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$ that proves that the minimum number of falsified clauses in ϕ is 2.

Example 2 *Let ϕ be the multiset of clauses $\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$. Figure 1 shows a saturated clause tableau T for ϕ, and Figure 2 shows the steps performed for saturating the leftmost branch of T.*

In Figure 2, we first create an initial tableau. Secondly, we apply the extension rule to clause $x_1 \vee x_2$, and mark it in the newly created branches (in the figure we write in bold the marked clauses in the leftmost branch, which is the branch on which we concentrate in this example). Thirdly, we apply the contradiction rule to $\overline{x}_1$ and x_1, and mark these clauses in the leftmost branch. Fourthly, we apply the extension rule to $x_1 \vee x_3$, and mark the clause in the newly created branches. Fifthly, we apply the extension rule to $x_2 \vee x_3$, and mark the clause in the newly created branches. Sixthly, we apply the contradiction rule to $\overline{x}_2$ and x_2, and mark these clauses in the leftmost branch. No more inference rules can be applied on the leftmost branch and therefore the branch is saturated, having as unmarked clauses $\{\Box, \Box, x_1, \overline{x}_3\}$. A similar process is repeated to create the rest of branches in Figure 1.

The saturated branches of the tableau of Figure 1 have cost 2 except for branches 3 and 6 (counting from left to right) that have cost 3. The unmarked clauses in each

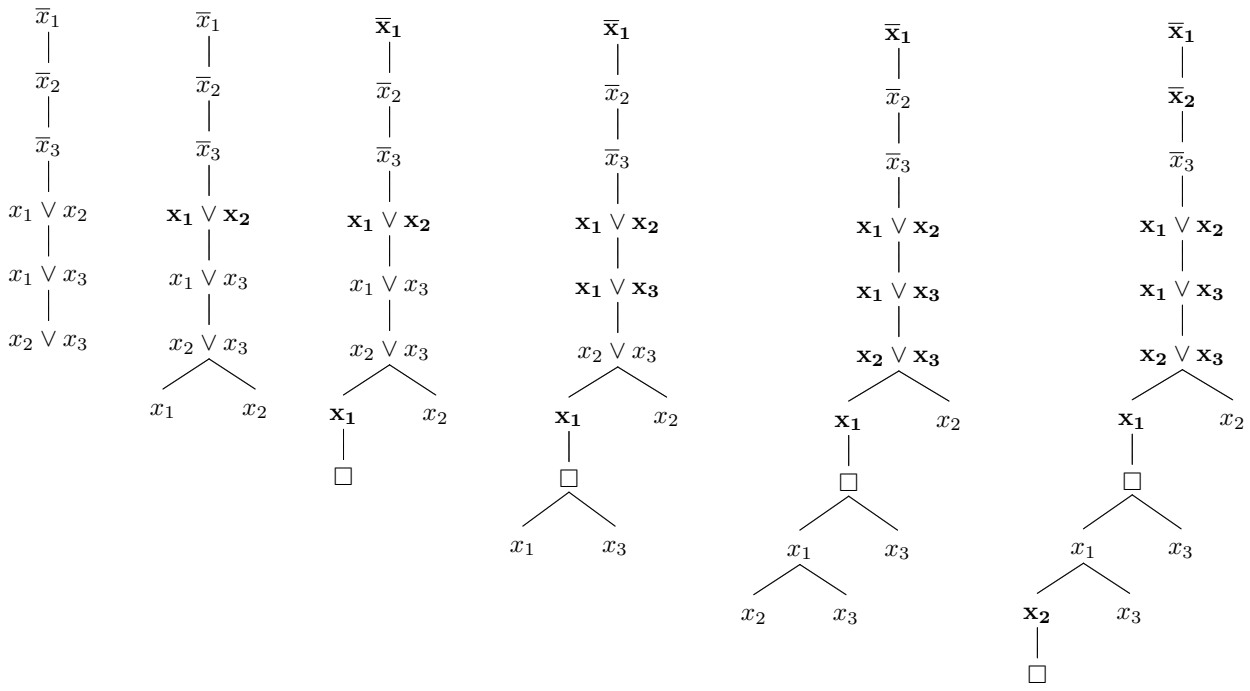


Figure 2: Steps performed for saturating the leftmost branch of the saturated clause tableau for $\phi = \{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$ from Example 4.

branch are: $\{\Box, \Box, x_1, \overline{x}_3\}$ *(branch 1),* $\{\Box, \Box, x_1, \overline{x}_2\}$ *(branch 2),* $\{\Box, \Box, \Box\}$ *(branch 3),* $\{\Box, \Box, \overline{x}_2, x_3\}$ *(branch 4),* $\{\Box, \Box, x_2, \overline{x}_3\}$ *(branch 5),* $\{\Box, \Box, \Box\}$ *(branch 6),* $\{\Box, \Box, \overline{x}_1, x_2\}$ *(branch 7), and* $\{\Box, \Box, \overline{x}_1, x_3\}$ *(branch 8). Therefore, the minimum number of falsified clauses in* ϕ *is 2.*

Lemma 1 *There exists a saturated tableau for each multiset of clauses* ϕ*.*

Proof We prove the lemma by induction on the number m of clauses in $\phi = \{C_1, \ldots, C_m\}$.

Basis: for $m = 1$, it is trivial. If the unique clause in ϕ has more than one literal, we apply the extension rule on the single branch of the initial tableau and get a saturated tableau; otherwise, the initial tableau is already saturated.

Inductive step: assume that T is a saturated tableau for $\{C_1, \ldots, C_{k-1}\}$. We prove that there exists a saturated tableau T'' for $\{C_1, \ldots, C_k\}$: Let T' be the tableau obtained from T by expanding C_k in each branch of T, and let T'' be the tableau obtained from T' by applying the contradiction rule on each branch of T'. The extension and the contradiction rule can be applied at most once in each branch, and T'' is saturated because no tableau rules can be applied on the branches of T''. □

4.1 Soundness and Completeness

We now prove that the minimum number of clauses that can be falsified in a multiset of clauses ϕ is m iff the cost of each saturated tableau for ϕ is m.

Theorem 1 Soundness. *Let* ϕ *be a multiset of clauses, and let* T *be a saturated clause tableau for* ϕ *of cost* m*. Then, the minimum number of clauses that can be falsified in* ϕ *is* m*.*

Proof T was obtained by creating a sequence of clause tableaux $T_0, \ldots, T_n$ ($n \geq 0$) such that T_0 is an initial tableau for ϕ, $T_n = T$, and T_i was obtained by a single application of the extension or the contradiction rule on an branch of T_{i-1} for $i = 1, \ldots, n$. Assume that I is an optimal assignment of ϕ that falsifies k clauses, where $k \neq m$. By induction on n, we prove that the minimum number of unmarked clauses that I falsifies among the branches of $T_0, \ldots, T_n$ (and in particular of T) is k:

Basis: T_0 has a single branch whose nodes are labelled with the clauses of ϕ, and such clauses are declared to be unmarked in that branch. So, I falsifies k unmarked clauses in T_0, and k is the minimum number of unmarked clauses that can be falsified in T_0.

Inductive step: Assume that the minimum number of unmarked clauses that I falsifies among the branches of T_{i-1} is k. We prove that the minimum number of unmarked clauses that I falsifies among the branches of T_i is also k.

Since T_i was constructed from T_{i-1} by applying either the contradiction rule or the extension rule on a branch B of T_{i-1} and the rest of branches of T_{i-1} remain unchanged in T_i, we just need to prove that I satisfies the same number of unmarked clauses in B and in at least one of the newly created branches, and does not decrease that number in the rest of newly created branches. We distinguish two cases:

- The contradiction rule was applied on B: two complementary unit clauses in B become marked and an empty clause is added in the new branch B'. Since exactly one

of the newly marked unit clauses was falsified by I and we added one empty clause, I falsifies the same number of unmarked clauses in B and B'.

- The extension rule was applied on B: If I satisfies the extended clause C of B, then I satisfies the leaf node of at least one of the newly created branches, say B'. The number of falsified unmarked clauses is preserved in B' and does not decrease in the rest of branches. If I falsifies C, then I falsifies the leaf nodes of all the newly created branches, and the number of falsified unmarked clauses is preserved in all these branches because C becomes marked after the extension.

We proved that the minimum number of unmarked clauses that I falsifies among the branches of $T_0, \ldots, T_n$ —and in particular of T— is k but this is in contradiction with T being a saturated tableau for ϕ that has cost m: Since T is saturated, the unmarked clauses of any branch B of T with minimum cost is the union of a multiset with m empty clauses and a multiset of unit clauses whose complementary unit clauses do not occur in it. The multiset of unit clauses is clearly satisfiable, and so the minimum number of unmarked clauses that can be falsified in B is m (not k), and is at least m in the rest of branches of T. Hence, the minimum number of clauses that can be falsified in ϕ is m.

□

Theorem 2 Completeness. *Let ϕ be a multiset of clauses whose minimum number of clauses that can be falsified is m. Then, each saturated clause tableau T for ϕ has cost m.*

Proof Assume that there is a saturated tableau T for ϕ that does not have cost m. We distinguish two cases:

(i) T has a branch B that has cost k, where $k < m$. Then, the unmarked clauses of B are the union of a multiset with k empty clauses and a satisfiable multiset of unit clauses (otherwise, B could not be saturated because the contradiction rule could be applied). We define an assignment I of ϕ as follows: $I(x) = 1$ ($I(x) = 0$) if x ($\overline{x}$) is an unmarked clause of B, and $I(x') = 0$ if variable x' does not occur in any unmarked clause of B. We next prove that I satisfies at least $|\phi| - k$ clauses of ϕ, or equivalently I falsifies at most k clauses of ϕ. I is clearly an optimal assignment of B. If we undo all the applications of the contradiction rule in B, we get a branch B' whose unmarked clauses form a multiset of unit clauses ϕ' that contains as many unit clauses as clauses are in ϕ, and each literal of each unit clause of ϕ' was derived from a different clause of ϕ. Since the clause of ϕ' are unit and there are k complementary pairs of unit clauses, I satisfies $|\phi|-k$ clauses of ϕ', and at least $|\phi| - k$ clauses of ϕ. We have therefore an assignment of ϕ that cannot falsify more than k clauses, but this is in contradiction with m being the minimum number of clauses that can be falsified in ϕ because $k < m$.

(ii) T has no branch of cost m. This is in contradiction with m being the minimum number of clauses that can be falsified in ϕ. Since the tableau rules preserve the minimum number of falsified clauses, T must have a saturated branch of cost m.

Hence, each saturated clause tableau T for a multiset of clauses ϕ has cost m if the minimum number of clauses that can be falsified in ϕ is m. □

input: a multiset of clauses ϕ over the set of propositional variables $\{x_1, \ldots, x_n\}$
T := initial clause tableau for ϕ
min_cost := number of clauses in ϕ
while $\exists$ an unmarked non-unit clause C in a branch B of T **do**
  $T := T$ after applying the extension rule to C on B
  Let $B_1, \ldots, B_r$ be the newly created branches
  Declare C to be marked in $B_1, \ldots, B_r$
  $T := T$ after applying the contradiction rule on $B_1, \ldots, B_r$
  if $\exists$ saturated branch $B_i \in T$ & $min\_cost > \#empty(B_i)$
    then $min\_cost := \#empty(B_i)$
endwhile
B := branch of T of minimum cost (min_cost)
$I := \emptyset$
for $i := 1$ **to** n
  if x_i is an unmarked unit clause in B
    then $I := I \cup \{x_i \leftarrow 1\}$
    else $I := I \cup \{x_i \leftarrow 0\}$
endfor
output: min_cost, I

Figure 3: An exact clause tableau algorithm for MaxSAT.

4.2 A Tableau-Based Algorithm for MaxSAT

Figure 3 shows the pseudo-code of an algorithm for MaxSAT that creates a saturated tableau for the input multiset of clauses ϕ. In addition, it generates an optimal assignment for ϕ. Function $\#empty$ returns the number of empty clauses in the branch provided as parameter.

The algorithm produces an optimal assignment I of the input multiset of clauses ϕ from a branch B of minimum cost of the saturated tableau created in the first part of the algorithm. The optimal assignment I sets a variable x to 1 (0) if B has a node labelled with the unmarked clause x ($\overline{x}$); the rest of variables can be set to either 0 or 1, but we set them to 0 in the algorithm.

The argument to show that I is optimal comes from the proof of Theorem 2: Let m be the value of min_cost returned by the algorithm. It is clear that I is an optimal assignment of B. If we undo all the applications of the contradiction rule in B, we get a branch B' whose unmarked clauses form a multiset of unit clauses ϕ' that contains as many unit clauses as clauses are in ϕ, and each literal of each unit clause of ϕ' was derived from a different clause of ϕ. Since the clause of ϕ' are unit and there are m complementary pairs of unit clauses, I satisfies $|\phi| - m$ clauses of ϕ', and at least $|\phi| - m$ clauses of ϕ. Since B is a branch of minimum cost of a saturated tableau for ϕ and B has cost m, each assignment of ϕ falsifies at least m clauses. Therefore, I satisfies exactly $|\phi| - m$ clauses of ϕ and is optimal.

Example 3 *Let ϕ be the multiset from Example 4. From each saturated branch of minimum cost in the tableau of Figure 1 we derive an optimal assignment. For example, from branch 1, whose unmarked clauses are $\{\Box, \Box, x_1, \overline{x}_3\}$, we derive the assignment $\{x_1 \leftarrow 1, x_2 \leftarrow 0, x_3 \leftarrow 0\}$. By convention, we set the literals that do not appear in unmarked clauses to 0, but they could also be set to 1. Hence, $\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 0\}$ is also an optimal assignment.*

Note that the saturated branches that do not have minimum cost only contain three empty clauses. This means

that the assignments $\{x_1 \leftarrow 0, x_2 \leftarrow 0, x_3 \leftarrow 0\}$ *and* $\{x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1\}$ *are not optimal.*

An immediate application of our tableau calculus is to replace the common branching in branch-and-bound (BnB) MaxSAT solvers such as [Abramé and Habet, 2014b; Argelich and Manyà, 2006; Alsinet *et al.*, 2008; Heras and Larrosa, 2006; Li *et al.*, 2007; Kuegel, 2010] by the clause branching suggested by the extension rule: instead of branching on a variable x and its negation $\overline{x}$, select a clause $l_1 \vee \cdots \vee l_r$ in the current formula and create r branches, one branch for each literal from $\{l_1, \ldots, l_r\}$, and proceed as usual in BnB MaxSAT solvers.

On the other hand, the algorithm of Figure 3 can be drastically improved by incorporating solving techniques that implement BnB MaxSAT solvers such as the computation of an initial upper bound with local search [Cai *et al.*, 2014], the computation at each node of a lower bound by detecting disjoint subsets of inconsistent formulas with a linear-time procedure that applies unit resolution [Li *et al.*, 2010; 2005; 2006], and heuristics for selecting the clause to be expanded.

Note that the suggested clause branching for MaxSAT is not valid for MinSAT [Ignatiev *et al.*, 2016; Li *et al.*, 2011; 2012], because the extension rule preserves the minimum number of falsified clauses among the newly created branches but does not preserve the maximum number of falsified clauses. This also implies that the proposed calculus is unsound for MinSAT. Recall that MinSAT for a multiset of clauses ϕ is to find an assignment that maximizes the number of falsified clauses in ϕ.

Finally, it is worth mentioning that the described tableau-based algorithm only needs polynomial space, while existing variable elimination algorithms based on MaxSAT resolution require exponential memory space [Bonet *et al.*, 2007].

5 Extension to WMaxSAT and WPMaxSAT

Many practical optimization problems admit more compact and natural MaxSAT encodings if they are encoded using weighted clauses instead of unweighted ones, as well as considering hard and soft clauses. To keep the description as simple as possible, we presented the tableau method for unweighted MaxSAT, but the proposed tableau calculus can be extended to solve both WMaxSAT and WPMaxSAT.

In the case of WMaxSAT, we should keep in mind that a weighted clause (c, w) is equivalent to have w copies of the unweighted clause c. So, the application of the contradiction rule to two unit clauses $(l, w_1), (\overline{l}, w_2)$ amounts to adding an unmarked empty clause with weight $w = min(w_1, w_2)$ (i.e.; $(\square, w)$), declare the clauses $(l, w_1), (\overline{l}, w_2)$ to be marked, and add the unmarked clauses $(l, w_1 - w)$ and $(\overline{l}, w_2 - w)$ in the newly created branch. The application of the extension rule to a weighted clause $(l_1 \vee \cdots \vee l_r, w)$, amounts to append r nodes below the current branch, labelling each node with a different unit weighted clause from $\{(l_1, w), \ldots, (l_r, w)\}$. Finally, to get a complete calculus, we need to define a contraction rule: if a branch contains two unmarked clauses (C, w_1) and (C, w_2), mark these clauses and add the unmarked clause $(C, w_1 + w_2)$ in the newly created branch.

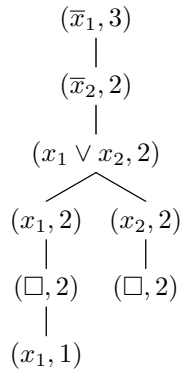


Figure 4: A saturated clause tableau for the multiset of weighted clauses $\phi = \{(\overline{x}_1, 3), (\overline{x}_2, 2), (x_1 \vee x_2, 2)\}$ that proves that the minimum sum of weights of falsified clauses in ϕ is 2.

Example 4 *Let* ϕ *be the multiset of weighted clauses* $\{(\overline{x}_1, 3), (\overline{x}_2, 2), (x_1 \vee x_2, 2)\}$. *Figure 4 shows a saturated clause tableau* T *for* ϕ. *We first apply the extension rule to* $(x_1 \vee x_2, 2)$ *and derive two new branches. In the leftmost branch, the application of the contradiction rule to* $(\overline{x}_1, 3)$ *and* $(x_1, 2)$ *yields* $(\square, 2)$ *and* $(x_1, 1)$. *In the rightmost branch, the application of the contradiction rule to* $(\overline{x}_2, 2)$ *and* $(x_2, 2)$ *yields* $(\square, 2)$. *The two saturated branches of the tableau have cost 2. The unmarked clauses in each branch are:* $\{(\overline{x}_2, 2), (\square, 2), (\overline{x}_1, 1)\}$ *(branch 1), and* $\{(\overline{x}_1, 3), (\square, 2)\}$ *(branch 2). Therefore, the minimum sum of weights of falsified clauses in* ϕ *is 2.*

In the case of WPMaxSAT, we should add, to each hard clause, a weight greater than the sum of weights of the input soft clauses, and proceed as in WMaxSAT but ignoring those branches in which a contradiction is detected in hard clauses.

6 Conclusions

We defined a sound and complete clause tableau calculus for WPMaxSAT that is simpler and more intuitive than MaxSAT resolution because it avoids the use of compensation clauses. At the same time, the contributions of the paper provide a new angle to look at MaxSAT, as well as to compare the inference in SAT with the inference in MaxSAT. Interestingly, an immediate application of our results could be the incorporation of the defined clause branching into BnB MaxSAT solvers.

As future work we plan to define a complete tableau calculus for MinSAT, and extend our results to both non-CNF formulas and first-order clauses.

References

[Abramé and Habet, 2014a] André Abramé and Djamal Habet. Efficient application of Max-SAT resolution on inconsistent subsets. In *Principles and Practice of Constraint Programming - 20th International Conference, CP, Lyon, France*, pages 92–107, 2014.

[Abramé and Habet, 2014b] André Abramé and Djamal Habet. Local Max-Resolution in branch and bound solvers

for Max-SAT. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, Limassol, Cyprus*, pages 336–343, 2014.

[Abramé and Habet, 2015] André Abramé and Djamal Habet. On the resiliency of unit propagation to Max-Resolution. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, Buenos Aires, Argentina*, pages 268–274, 2015.

[Alsinet *et al.*, 2008] Teresa Alsinet, Felip Manyà, and Jordi Planes. An efficient solver for Weighted Max-SAT. *Journal of Global Optimization*, 41:61–73, 2008.

[Ansótegui *et al.*, 2013] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Sat-based MaxSAT algorithms. *Artificial Intelligence*, 196:77–105, 2013.

[Ansótegui *et al.*, 2016a] Carlos Ansótegui, Joel Gabàs, and Jordi Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *J. Heuristics*, 22(1):1–53, 2016.

[Ansótegui *et al.*, 2016b] Carlos Ansótegui, Joel Gabàs, Yuri Malitsky, and Meinolf Sellmann. MaxSAT by improved instance-specific algorithm configuration. *Artificial Intelligence*, 235:26–39, 2016.

[Argelich and Manyà, 2006] Josep Argelich and Felip Manyà. Exact Max-SAT solvers for over-constrained problems. *Journal of Heuristics*, 12(4–5):375–392, 2006.

[Bonet *et al.*, 2006] María Luisa Bonet, Jordi Levy, and Felip Manyà. A complete calculus for Max-SAT. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT-2006, Seattle, USA*, pages 240–251. Springer LNCS 4121, 2006.

[Bonet *et al.*, 2007] María Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8–9):240–251, 2007.

[Cai *et al.*, 2014] Shaowei Cai, Chuan Luo, John Thornton, and Kaile Su. Tailoring local search for partial maxsat. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI-2014, Québec City, Québec, Canada*, pages 2623–2629, 2014.

[D'Agostino, 1999] Marcello D'Agostino. Tableaux methods for classical propositional logic. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 45–123. Kluwer, 1999.

[Hähnle, 2001] Reiner Hähnle. Tableaux and related methods. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 100–178. Elsevier and MIT Press, 2001.

[Heras and Larrosa, 2006] Federico Heras and Javier Larrosa. New inference rules for efficient Max-SAT solving. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-2006, Boston/MA, USA*, pages 68–73, 2006.

[Ignatiev *et al.*, 2016] Alexey Ignatiev, António Morgado, Jordi Planes, and Joao Marques-Silva. Maximal falsifiability. *AI Communications*, 29(2):351–370, 2016.

[Kuegel, 2010] Adrian Kuegel. Improved exact solver for the Weighted MAX-SAT problem. In *Proceedings of Workshop Pragmatics of SAT, POS-10, Edinburgh, UK*, pages 15–27, 2010.

[Larrosa *et al.*, 2008] Javier Larrosa, Federico Heras, and Simon de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2–3):204–233, 2008.

[Li and Manyà, 2009] Chu Min Li and F. Manyà. MaxSAT, hard and soft constraints. In Armin Biere, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 613–631. IOS Press, 2009.

[Li *et al.*, 2005] Chu Min Li, Felip Manyà, and Jordi Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming, CP-2005, Sitges, Spain*, pages 403–414. Springer LNCS 3709, 2005.

[Li *et al.*, 2006] Chu Min Li, Felip Manyà, and Jordi Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proceedings of the 21st National Conference on Artificial Intelligence, AAAI-2006, Boston/MA, USA*, pages 86–91, 2006.

[Li *et al.*, 2007] Chu Min Li, Felip Manyà, and Jordi Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.

[Li *et al.*, 2010] Chu Min Li, Felip Manyà, Nouredine Ould Mohamedou, and Jordi Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, 15(4):456–484, 2010.

[Li *et al.*, 2011] Chu Min Li, Zhu Zhu, Felip Manyà, and Laurent Simon. Minimum satisfiability and its applications. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, Barcelona, Spain*, pages 605–610, 2011.

[Li *et al.*, 2012] Chu Min Li, Zhu Zhu, Felip Manyà, and Laurent Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, 190:32–44, 2012.

[Martins *et al.*, 2014] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In *Principles and Practice of Constraint Programming - 20th International Conference, CP, Lyon, France*, pages 531–548, 2014.

[Morgado *et al.*, 2013] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[Narodytska and Bacchus, 2014] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, Canada*, pages 2717–2723, 2014.