



# Clause tableaux for maximum and minimum satisfiability

Josep Argelich, Chu Min Li, Felip Manyà, Joan Ramon Soler

## ► To cite this version:

Josep Argelich, Chu Min Li, Felip Manyà, Joan Ramon Soler. Clause tableaux for maximum and minimum satisfiability. *Logic Journal of the IGPL*, 2019, 29, pp.7 - 27. 10.1093/jigpal/jzz025 . hal-04320635

**HAL Id: hal-04320635**

**<https://hal.science/hal-04320635v1>**

Submitted on 5 Dec 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/344737078>

# Clause tableaux for maximum and minimum satisfiability

Article in *Logic Journal of IGPL* · August 2019

DOI: 10.1093/jigpal/jzz025

CITATIONS

10

READS

67

4 authors:



**Josep Argelich**

Universitat de Lleida

50 PUBLICATIONS 539 CITATIONS

[SEE PROFILE](#)



**Chu-Min Li**

Université de Picardie Jules Verne

165 PUBLICATIONS 3,626 CITATIONS

[SEE PROFILE](#)



**Felip Manyà**

Spanish National Research Council

130 PUBLICATIONS 2,470 CITATIONS

[SEE PROFILE](#)



**Joan Ramon Soler**

Spanish National Research Council

9 PUBLICATIONS 72 CITATIONS

[SEE PROFILE](#)

# Clause tableaux for maximum and minimum satisfiability

JOSEP ARGELICH, *Departament d' Informàtica i Enginyeria Industrial (DIEI), Universitat de Lleida, 25001 Lleida, Spain.*

CHU MIN LI, *MIS, Université de Picardie Jules Verne, 80039 Amiens, France.*

FELIP MANYÀ\* AND JOAN RAMON SOLER, *Artificial Intelligence Research Institute (IIIA, CSIC), 08193 Bellaterra, Spain.*

## Abstract

The inference systems proposed for solving SAT are unsound for solving MaxSAT and MinSAT, because they preserve satisfiability but not the minimum and maximum number of clauses that can be falsified, respectively. To address this problem, we first define a clause tableau calculus for MaxSAT and prove its soundness and completeness. We then define a clause tableau calculus for MinSAT and also prove its soundness and completeness. Finally, we define a complete clause tableau calculus for solving both MaxSAT and MinSAT, in that the minimum number of generated empty clauses provides an optimal MaxSAT solution and the maximum number provides an optimal MinSAT solution.

*Keywords:* Boolean optimization, MaxSAT, MinSAT, tableaux, calculus, completeness.

## 1 Introduction

MaxSAT and MinSAT are nowadays competitive generic problem solving approaches that are able to solve challenging optimization problems in different areas (see e.g. [2–7, 15, 17, 19, 21, 23, 27–34] and the references therein for related work). MaxSAT is to find a truth assignment that minimizes the number of unsatisfied clauses in a multiset of clauses, while MinSAT is to find a truth assignment that maximizes the number of unsatisfied clauses.

The inference systems proposed for solving SAT are unsound for solving MaxSAT and MinSAT, because they preserve satisfiability but not the maximum and minimum number of clauses that can be falsified. Thus, we first need to define inference rules meeting that condition and then show that their correct application allows one to derive as many empty clauses as the minimum number of clauses that can be falsified in the case of MaxSAT, and as the maximum number of clauses that can be falsified in the case of MinSAT.

Resolution-style calculus have been defined for MaxSAT [9, 10, 20] and MinSAT [22], and the proposed variable elimination algorithms provide optimal solutions. The MaxSAT and MinSAT resolution rule states that the complementary clauses  $x \vee l_1^1 \vee \dots \vee l_m^1$  and  $\neg x \vee l_1^2 \vee \dots \vee l_n^2$  can be replaced with the usual resolvent  $l_1^1 \vee \dots \vee l_m^1 \vee l_1^2 \vee \dots \vee l_n^2$  and the following  $m + n$  compensation clauses:  $x \vee l_1^1 \vee \dots \vee l_m^1 \vee \neg l_1^2, \dots, x \vee l_1^1 \vee \dots \vee l_m^1 \vee l_1^2 \vee \dots \vee l_{n-1}^2 \vee \neg l_n^2, \neg x \vee l_1^2 \vee \dots \vee l_n^2 \vee \neg l_1^1, \dots, \neg x \vee l_1^2 \vee \dots \vee l_n^2 \vee l_1^1 \vee \dots \vee l_{m-1}^1 \vee \neg l_m^1$ . The role of the compensation clauses is to ensure that the two parent clauses are replaced by a collection of clauses that preserve

\*E-mail: felip@iiia.csic.es

## 2 Clause Tableaux for Maximum and Minimum Satisfiability

the number of unsatisfied clauses. The difference between MaxSAT and MinSAT lies in the way the variable elimination algorithms saturate the variables. Refinements of the previous resolution rule have been incorporated into branch-and-bound MaxSAT and MinSAT algorithms and have produced important speedups [1, 2, 20, 23, 24, 28].

In this paper, we first extend the clause tableau calculus for SAT [12, 16] to solve both MaxSAT [26] and MinSAT [25]. An essential issue to address is how to derive simpler subproblems in such a way that the minimum/maximum number of unsatisfied clauses is preserved. The proposed clause MaxSAT tableau calculus and the clause SAT tableau calculus are quite similar, in that they use the same rules but applied differently. The clause MinSAT tableau calculus has an inference rule that does not resemble the rules of the SAT and MaxSAT calculi. It is worth noticing that a clause MinSAT tableau calculus must be able to derive contradictions from satisfiable instances because they can have interpretations that falsify some clauses.

From the insights gained after analysing clause MaxSAT and MinSAT tableaux, we then propose a clause tableau calculus that is valid for both MaxSAT and MinSAT. It preserves adequately the number of unsatisfied clauses in the generated subproblems. The leaf nodes of a completed search tree contain a number of empty clauses ranging between the minimum and the maximum number of unsatisfied clauses in the input formula, and there is at least one branch with the minimum value and there is at least one branch with the maximum value. This calculus also provides optimal MaxSAT and MinSAT assignments by inspecting the optimal branches.

This is a journal version of three conference papers. It unifies results on clause MaxSAT tableaux appeared in [26], on clause MinSAT tableaux appeared in [25] and on a tableau-based procedure for MaxSAT and MinSAT appeared in [8].

The paper is structured as follows. Section 2 defines basic concepts. Section 3 reviews how clause tableaux can be used to solve SAT. Section 4 defines a complete clause tableau calculus for MaxSAT. Section 5 defines a complete clause tableau calculus for MinSAT. Section 6 describes a complete clause tableau calculus that is valid for both MaxSAT and MinSAT. Section 7 gives the conclusions.

## 2 Preliminaries

Given a set of propositional variables  $\{x_1, \dots, x_n\}$ , a literal is a variable  $x_i$  or its negation  $\neg x_i$ . A weighted clause is a pair  $(c, w)$ , where  $c$  is a disjunction of literals and  $w$ , its weight, is a positive integer.

A truth assignment assigns to each variable either 0 (false) or 1 (true). It satisfies literal  $x_i$  ( $\neg x_i$ ) if  $x_i$  evaluates to 1 (0), it satisfies weighted clause  $(c, w)$  if it satisfies a literal of  $c$  and it satisfies a multiset of clauses if it satisfies all its clauses. The weight  $w$  is the penalty of violating clause  $c$ . When all clauses have the same weight, their weights are omitted.

The Weighted Partial MaxSAT problem, or WPMMaxSAT, for a multiset of clauses  $\phi$  is to find an assignment that satisfies all the hard clauses and minimizes the sum of the weights of the unsatisfied soft clauses. The most common subproblems of WPMMaxSAT are the following ones: Weighted MaxSAT (WMaxSAT), which is WPMMaxSAT without hard clauses; Partial MaxSAT (PMaxSAT), which is WPMMaxSAT when all the soft clauses have the same weight; and MaxSAT, which is PMaxSAT without hard clauses.

The Weighted Partial MinSAT problem, or WPMMinSAT, for a multiset of clauses  $\phi$  is to find an assignment that satisfies all the hard clauses and maximizes the sum of the weights of the unsatisfied soft clauses. The most common subproblems of WPMMinSAT are the following ones: Weighted MinSAT (WMinSAT), which is WPMMinSAT without hard clauses; Partial MinSAT (PMinSAT),

TABLE 1. Expansion rules of a complete clause tableau calculus for SAT

$\begin{array}{c} C_1 \\   \\ \vdots \\   \\ C_m \end{array}$	$\frac{l_1 \vee l_2 \vee \dots \vee l_n}{\begin{array}{ c c c c } \hline l_1 & l_2 & \dots & l_n \\ \hline \end{array}}$	$\frac{\begin{array}{c} l \\ \neg l \end{array}}{\square}$
Initial tableau rule	$n$ -ary extension rule	$\square$ -rule

which is WPMInSAT when all the soft clauses have the same weight; and MinSAT, which is PMinSAT without hard clauses. The SAT problem is PMaxSAT/PMInSAT without soft clauses.

We represent MaxSAT and MinSAT instances as multisets of clauses. Repeated clauses cannot be collapsed into one clause as in SAT, because then the maximum/minimum number of unsatisfied clauses might not be preserved. For example, the multiset of unit clauses  $\{x_1, \neg x_1, x_1, \neg x_1\}$  has a minimum of two unsatisfied clauses while  $\{x_1, \neg x_1\}$  has just one unsatisfied clause. In fact,  $\{x_1, \neg x_1, x_1, \neg x_1\}$  is equivalent to the multiset of weighted clauses  $\{(x_1, 2), (\neg x_1, 2)\}$ . Clauses can be represented by the set of its literals as in SAT because repeated literals can be collapsed into one literal without affecting the preservation of the number of unsatisfied clauses.

Let  $\phi$  be a multiset of clauses and let  $l_1, \dots, l_r$  be literals that occur in  $\phi$ . The instantiation of  $l_1, \dots, l_r$  in  $\phi$ , denoted by  $\phi_{l_1|\dots|l_r}$ , is the multiset of clauses resulting of eliminating from  $\phi$  all the occurrences of  $\neg l_1, \dots, \neg l_r$  and removing all the clauses with occurrences of  $l_1, \dots, l_r$ .

### 3 Clause tableaux for SAT

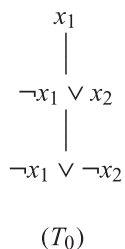
It is common to view the tableau method for solving SAT as a proof by case distinction that allows one to systematically generate subcases until elementary contradictions are reached [14, 35]. In the context of SAT, a clause tableau is a tree with a finite number of branches whose nodes are labelled 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  $\phi = \{C_1, \dots, C_m\}$ , we start by creating an initial tableau that has a single branch with  $m$  nodes, where each node is labelled with a clause of  $\phi$ . This process is known as the application of the initial tableau rule. Then, we select an open branch  $B$  and a clause  $l_1 \vee \dots \vee l_r$  of  $\phi$  with  $r \geq 2$  that has not yet been expanded in  $B$  and append  $r$  sibling nodes below  $B$ , labelling each node with a different unit clause from  $\{l_1, \dots, l_r\}$ . 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 by applying the contradiction rule. In this paper, closing a branch amounts to deriving an empty clause. 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. The set of clauses  $\phi$  is declared to be unsatisfiable in the first case and satisfiable in the second case. Table 1 shows the expansion rules of a complete clause tableau calculus for SAT.

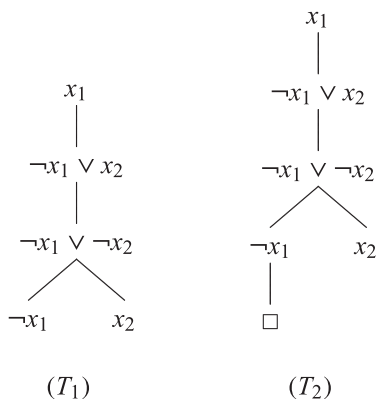
#### EXAMPLE 3.1

To determine the satisfiability of  $\phi = \{x_1, \neg x_1 \vee x_2, \neg x_1 \vee \neg x_2\}$  with clause tableaux we start by creating the initial tableau ( $T_0$ ),

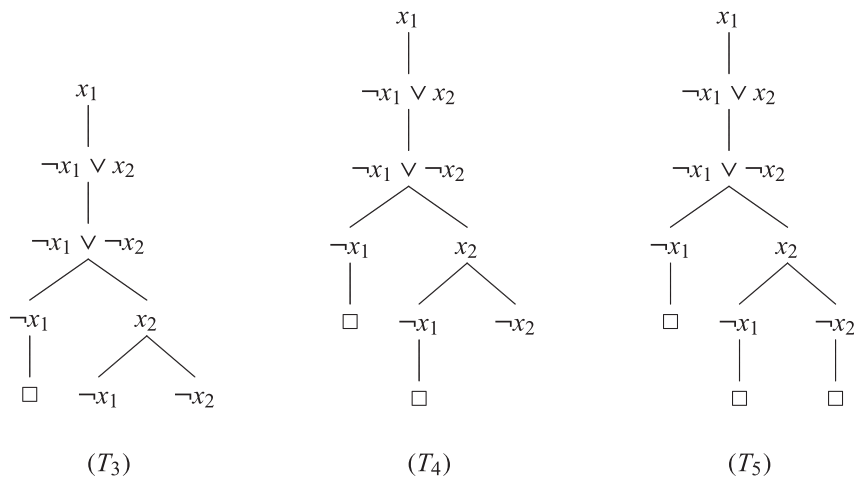
#### 4 Clause Tableaux for Maximum and Minimum Satisfiability



We then expand the second node ( $T_1$ ) and close the leftmost branch by applying the contradiction rule to  $x_1$  and  $\neg x_1$ , obtaining another clause tableau ( $T_2$ ):



Finally, we expand the third node on the rightmost branch ( $T_3$ ) and close the new leftmost ( $T_4$ ) and rightmost branches ( $T_5$ ), obtaining a clause tableau proof of the unsatisfiability of  $\phi$ ,



Formally, a clause tableau proof of the unsatisfiability of a set of clauses  $\phi$  is a sequence of clause tableaux  $T_0, \dots, T_n$  such that  $T_0$  is an initial tableau of  $\phi$ ,  $T_n$  is a closed tableau and  $T_i$  has been obtained by a single application of the extension or contradiction rule on an open

branch of  $T_{i-1}$  for  $i = 1, \dots, n$ . 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  $\phi$ . It is common to say that  $T_n$  is a clause tableau proof because it collapses all the sequence of tableaux. In Example 3.1, the sequence  $T_0, T_1, T_2, T_3, T_4, T_5$  is a clause tableau proof, although  $T_5$  alone is also considered to be a proof.

We say that a clause tableau is completed when all its branches are closed or it contains an open branch in which it was not possible to detect a contradiction with the expansion rules of Table 1.

From a semantic perspective, given a set of clauses  $\phi$  and a completed clause tableau  $T$  for  $\phi$ , we have that  $\phi$  is satisfiable iff there is a branch of  $T$  such that the conjunction of all its literals is satisfiable. Alternatively,  $\phi$  is unsatisfiable iff all the branches of  $T$  are unsatisfiable.

## 4 Clause tableaux for MaxSAT

The clause SAT tableau calculus is not valid for solving MaxSAT, but it can become sound and complete by applying differently the expansion rules of Table 1.

Firstly, the application of expansion rules in a branch cannot stop once a contradiction is detected. Since the aim of MaxSAT is to derive all the possible contradictions, the application of rules in a branch should continue until no more expansion rules can be applied. Thus, a different notion of completed tableau is needed.

Secondly, the application of rules in SAT leads to accumulate the newly added unit clauses in the branch in such a way that satisfiability is preserved in at least one branch when the input set of clauses is satisfiable. However, the addition of redundant clauses can lead to wrong MaxSAT solutions. In clause MaxSAT tableaux, the goal should be to keep the minimum number of unsatisfied clauses in at least one branch and not to decrease that number in the rest of branches. As we show below, the rules of Table 1 satisfy that condition provided that we maintain active and inactive clauses. In other words, once a clause has been used as a premise of a rule in a branch, it cannot be used again in that branch and becomes inactive. For example, thanks to distinguishing between active and inactive clauses, we will detect one contradiction in the multiset of unit clauses  $\{x_1, \neg x_1, \neg x_1\}$  and two in  $\{x_1, x_1, \neg x_1, \neg x_1\}$ . Without that, we could detect two contradictions in the first case, obtaining a wrong answer. In fact, the inference rules of MaxSAT can be seen as rewriting rules.

### DEFINITION 4.1

A clause MaxSAT tableau is a tree with a finite number of branches whose nodes are labelled with clauses. A branch is a maximal path in a tree, and we assume that branches have a finite number of nodes.

### DEFINITION 4.2

Let  $\phi = \{C_1, \dots, C_m\}$  be a multiset of clauses. A clause MaxSAT tableau for  $\phi$  is constructed by a sequence of applications of the following expansion rules:

- Initialize A tree with a single branch with  $m$  nodes such that each node is labelled with a clause of  $\phi$  is a clause MaxSAT tableau for  $\phi$ . Such a tableau is called initial tableau, and its clauses are declared to be active.

## 6 Clause Tableaux for Maximum and Minimum Satisfiability

- Extension** Given a clause MaxSAT tableau  $T$  for  $\phi$ , a branch  $B$  of  $T$  and a node of  $B$  labelled with an active clause  $l_1 \vee \dots \vee l_r$  with  $r \geq 2$ , the tableau obtained by creating  $r$  sibling nodes below  $B$  and labelling each node with a different unit clause from  $\{l_1, \dots, l_r\}$  is a clause MaxSAT tableau for  $\phi$ . The clause  $l_1 \vee \dots \vee l_r$  becomes inactive in the new branches, and the unit clauses  $l_1, \dots, l_r$  are declared to be active.
- Contradiction** Given a clause MaxSAT tableau  $T$  for  $\phi$ , a branch  $B$  of  $T$  and two nodes of  $B$  labelled with two active unit clauses  $l$  and  $\neg l$ , the tableau obtained by appending a node labelled with an empty clause ( $\square$ ) below  $B$  is a clause MaxSAT tableau for  $\phi$ . The unit clauses  $l$  and  $\neg l$  become inactive in  $B$ , and the added empty clause ( $\square$ ) is considered active.

### DEFINITION 4.3

Let  $T$  be a clause MaxSAT tableau for a multiset of clauses  $\phi$ , and let  $B$  be a branch of  $T$ . Branch  $B$  is saturated when all its active clauses are unit or empty, and the contradiction rule cannot be further applied on  $B$ . Tableau  $T$  is completed iff all its branches are saturated. The cost of a saturated branch is the number of empty clauses in it. The cost of a completed clause MaxSAT tableau is the minimum cost among all its branches.

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

### EXAMPLE 4.4

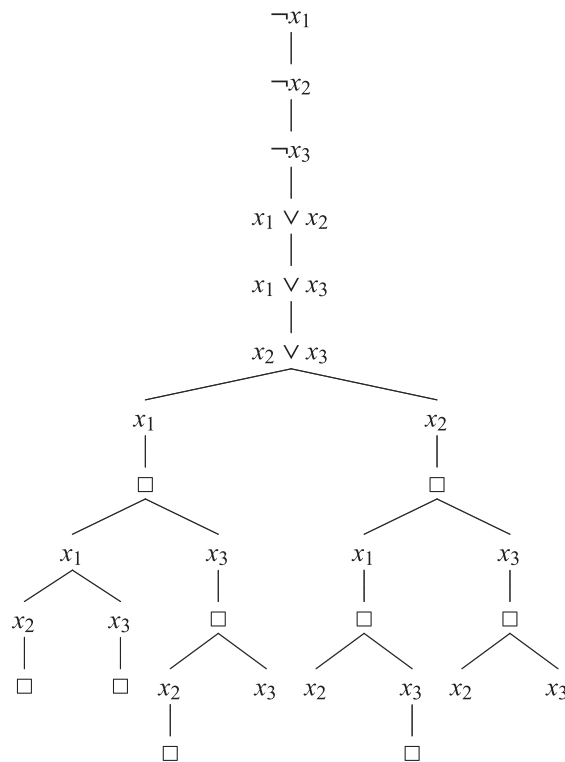
Let  $\phi = \{\neg x_1, \neg x_2, \neg x_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$  be a multiset of clauses. Figure 1 shows a completed clause MaxSAT tableau  $T$  for  $\phi$ , and Figure 2 shows the steps performed for saturating the leftmost branch of  $T$ .

In Figure 2, we first create an initial tableau. Secondly, we apply the extension rule to clause  $x_1 \vee x_2$  and declare it inactive in the newly created branches (in the figure we write in bold the inactive clauses in the leftmost branch, which is the branch on which we concentrate in this example). Thirdly, we apply the contradiction rule to  $\neg x_1$  and  $x_1$  and declare these clauses inactive in the leftmost branch. Fourthly, we apply the extension rule to  $x_1 \vee x_3$  and declare it inactive in the newly created branches. Fifthly, we apply the extension rule to  $x_2 \vee x_3$  and declare it inactive. Sixthly, we apply the contradiction rule to  $\neg x_2$  and  $x_2$  and declare these clauses inactive in the leftmost branch. No more inference rules can be applied on the leftmost branch, and therefore, the branch is saturated, having as active clauses  $\{\square, \square, x_1, \neg x_3\}$ . 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 active clauses in each branch are  $\{\square, \square, x_1, \neg x_3\}$  (branch 1),  $\{\square, \square, x_1, \neg x_2\}$  (branch 2),  $\{\square, \square, \square\}$  (branch 3),  $\{\square, \square, \neg x_2, x_3\}$  (branch 4),  $\{\square, \square, x_2, \neg x_3\}$  (branch 5),  $\{\square, \square, \square\}$  (branch 6),  $\{\square, \square, \neg x_1, x_2\}$  (branch 7) and  $\{\square, \square, \neg x_1, x_3\}$  (branch 8). Therefore, the minimum number of unsatisfied clauses in  $\phi$  is 2.

### 4.1 Soundness and completeness of clause MaxSAT tableaux

We prove that the minimum number of clauses that can be falsified in a multiset of clauses  $\phi$  is  $m$  iff the cost of each completed clause MaxSAT tableau for  $\phi$  is  $m$ .



### THEOREM 4.5

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

Inductive step: Assume that the minimum number of active clauses that  $I$  falsifies among the branches of  $T_{i-1}$  is  $k$ . We prove that the minimum number of active 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 active clauses in  $B$  and in at least one of the newly created

## 8 Clause Tableaux for Maximum and Minimum Satisfiability

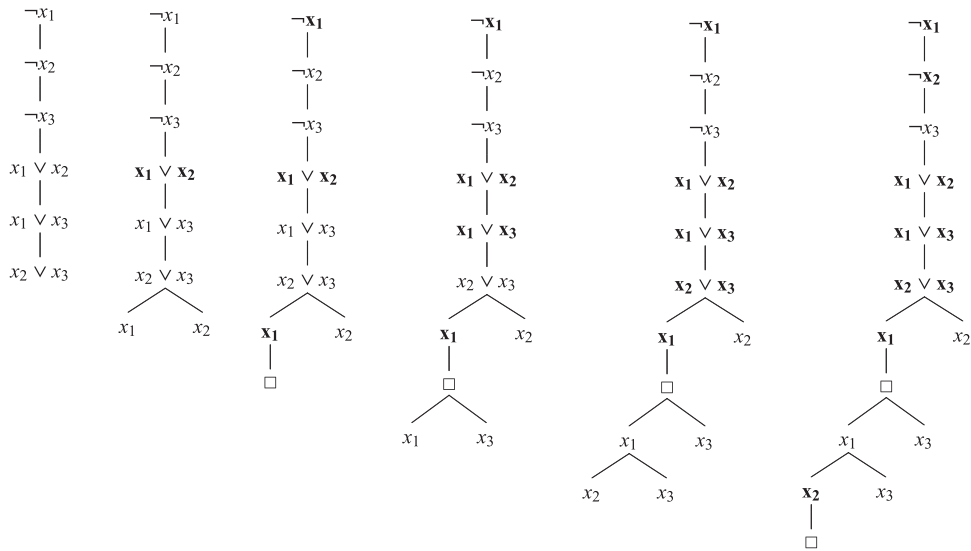


FIGURE 2. Steps performed for saturating the leftmost branch of the completed clause MaxSAT tableau for  $\phi = \{\neg x_1, \neg x_2, \neg x_3, x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$  from Example 4.7.

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 inactive and an empty clause is added to the new branch  $B'$ . Since exactly one of the newly inactive unit clauses was falsified by  $I$  and we added one empty clause,  $I$  falsifies the same number of active 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 unsatisfied active 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 unsatisfied active clauses is preserved in all these branches because  $C$  becomes inactive after the extension.

We proved that the minimum number of active clauses that  $I$  falsifies among the branches of  $T_0, \dots, T_n$ —and in particular of  $T$ —is  $k$ , but this is in contradiction with  $T$  being a completed MaxSAT tableau for  $\phi$  that has cost  $m$ ; since  $T$  is completed, the active 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 active 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  $\phi$  is  $m$ .  $\square$

### THEOREM 4.6

**Completeness.** Let  $\phi$  be a multiset of clauses whose minimum number of clauses that can be falsified is  $m$ . Then, each completed clause MaxSAT tableau for  $\phi$  has cost  $m$ .

**PROOF.** Each clause MaxSAT tableau for  $\phi$  can be completed after a finite number of steps. This follows from the fact that the number of applications of extension rules in a branch is bound by the

number of clauses in the input multiset and the number of applications of the contradiction rules is bounded by the number of literals occurring in the input multiset.

Assume that there is a completed MaxSAT tableau  $T$  for  $\phi$  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 active 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  $\phi$  as follows:  $I(x) = 1$  ( $I(x) = 0$ ) if  $x$  ( $\neg x$ ) is an active clause of  $B$ , and  $I(x') = 0$  if variable  $x'$  does not occur in any active clause of  $B$ . We next prove that  $I$  satisfies at least  $|\phi| - k$  clauses of  $\phi$ , or equivalently,  $I$  falsifies at most  $k$  clauses of  $\phi$ .  $I$  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 active clauses form a multiset of unit clauses  $\phi'$  that contains as many unit clauses as clauses are in  $\phi$ , and each literal of each unit clause of  $\phi'$  was derived from a different clause of  $\phi$ . Since the clause of  $\phi'$  are unit and there are  $k$  complementary pairs of unit clauses,  $I$  satisfies  $|\phi| - k$  clauses of  $\phi'$ , and at least  $|\phi| - k$  clauses of  $\phi$ . We have therefore an assignment of  $\phi$  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  $\phi$  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  $\phi$ . Since the tableau rules preserve the minimum number of unsatisfied clauses,  $T$  must have a saturated branch of cost  $m$ .

Hence, each completed clause MaxSAT tableau  $T$  for a multiset of clauses  $\phi$  has cost  $m$  if the minimum number of clauses that can be falsified in  $\phi$  is  $m$ .  $\square$

From the proof of Theorem 4.6, it follows that we can derive an optimal MaxSAT assignment  $I$  from a saturated branch  $B$  of minimum cost. The optimal assignment  $I$  sets a variable  $x$  to 1 (0) if  $B$  has a node labelled with the active clause  $x$  ( $\neg x$ ); the rest of variables can be set to either 0 or 1.

#### 4.2 Clause tableaux for WMaxSAT and WPMMaxSAT

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 clause tableaux for unweighted MaxSAT, but the proposed calculus can be extended to solve both WMaxSAT and WPMMaxSAT.

In the case of WMaxSAT, we should keep in mind that a weighted clause  $(c, w)$  is equivalent to having  $w$  copies of the unweighted clause  $c$ . So, the application of the contradiction rule to two unit clauses  $(l, w_1), (\neg l, w_2)$  amounts to adding an active empty clause with weight  $w = \min(w_1, w_2)$  (i.e.;  $(\square, w)$ ), declare the clauses  $(l, w_1), (\neg l, w_2)$  to be inactive and add the active clauses  $(l, w_1 - w)$  and  $(\neg l, w_2 - w)$  in the newly created branch. Clauses with weight 0 are not added. The application of the extension rule to a weighted clause  $(l_1 \vee \dots \vee l_r, w)$  amounts to appending  $r$  nodes below the current branch, labelling each node with a different unit weighted clause from  $\{(l_1, w), \dots, (l_r, w)\}$ . Finally, to get a complete calculus, we need to define a contraction rule: if a branch contains two active clauses  $(C, w_1)$  and  $(C, w_2)$ , inactivate these clauses and add the active clause  $(C, w_1 + w_2)$  in the newly created branch.

##### EXAMPLE 4.7

Let  $\phi = \{(\neg x_1, 3), (\neg x_2, 2), (x_1 \vee x_2, 2)\}$  be a multiset of weighted clauses. Figure 3 shows a completed clause WMaxSAT tableau  $T$  for  $\phi$ . We first apply the extension rule to  $(x_1 \vee x_2, 2)$  and

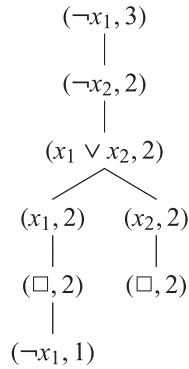


FIGURE 3. A completed clause WMaxSAT tableau for the multiset of weighted clauses  $\phi = \{(\neg x_1, 3), (\neg x_2, 2), (x_1 \vee x_2, 2)\}$  that proves that the minimum sum of weights of unsatisfied clauses in  $\phi$  is 2.

derive two new branches. In the leftmost branch, the application of the contradiction rule to  $(\neg x_1, 3)$  and  $(x_1, 2)$  yields  $(\square, 2)$  and  $(\neg x_1, 1)$ . In the rightmost branch, the application of the contradiction rule to  $(\neg x_2, 2)$  and  $(x_2, 2)$  yields  $(\square, 2)$ . The two saturated branches of the tableau have cost 2. The active clauses in each branch are  $\{(\neg x_2, 2), (\square, 2), (\neg x_1, 1)\}$  (left branch) and  $\{(\neg x_1, 3), (\square, 2)\}$  (right branch). Therefore, the minimum sum of weights of unsatisfied clauses in  $\phi$  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. Moreover, we could prune those branches in which a contradiction is detected between hard clauses or clauses derived from hard clauses.

## 5 Clause tableaux for MinSAT

We showed in the previous section that the minimum number of empty clauses among the branches of a completed clause MaxSAT tableau for a multiset of clauses  $\phi$  is equal to the number of clauses falsified by an optimal MaxSAT assignment of  $\phi$ . Nevertheless, the maximum number of empty clauses among the branches of a completed clause MaxSAT tableau for  $\phi$  is not the maximum number of clauses that can be falsified in  $\phi$ , i.e. clause MaxSAT tableaux cannot solve MinSAT. This is so because the extension rule is unsound for MinSAT in the sense that it does not preserve the maximum number of clauses that can be falsified.

In the rest of the section, we first define a clause MinSAT tableau calculus that incorporates a sound extension rule. We then prove the soundness and completeness of the proposed calculus. Finally, we present how our results can be extended to deal with WMinSAT and WPMInSAT instances. Note that, in MinSAT, we also need to derive contradictions from satisfiable instances because the maximum number of clauses that can be falsified in a satisfiable instance other than the empty multiset is always greater than or equal to one.

### DEFINITION 5.1

A clause MinSAT tableau is a finite tree whose nodes are labelled with multisets of clauses. A branch is a maximal path in a tree.

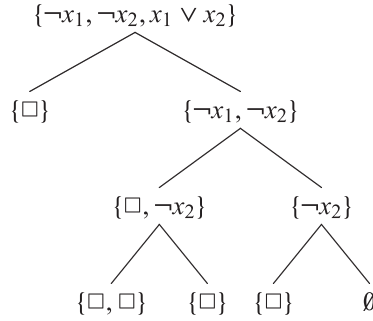


FIGURE 4. Completed clause MinSAT tableau for  $\phi_1 = \{\neg x_1, \neg x_2, x_1 \vee x_2\}$ .

Note that we now label the nodes of a tableau with multisets of clauses instead of clauses. Also note that clause MinSAT tableaux do not need to declare clauses either as active or inactive.

#### DEFINITION 5.2

Let  $\phi$  be a multiset of clauses. A clause MinSAT tableau for  $\phi$  is constructed by a sequence of applications of the following rules:

- Initialize** A tree with a single branch with a single node labelled with the multiset of clauses  $\phi$  is a clause MinSAT tableau for  $\phi$ . Such a tableau is called initial tableau.
- Extension** Given a clause MinSAT tableau  $T$  for  $\phi$ , and a branch  $B$  of  $T$  whose leaf node is labelled with a multiset  $\phi' = \phi'' \cup \{l_1 \vee \dots \vee l_r\}$ , the tableau obtained by appending a new left node below  $B$  labelled with the multiset  $\phi'_{\neg l_1 | \dots | \neg l_r}$  and a new right node below  $B$  labelled with the multiset  $\phi''$  is a clause MinSAT tableau for  $\phi$ .

In the definition of the extension rule, note that  $\phi'_{\neg l_1 | \dots | \neg l_r} = \{\square\} \cup \phi''_{\neg l_1 | \dots | \neg l_r}$ .

#### DEFINITION 5.3

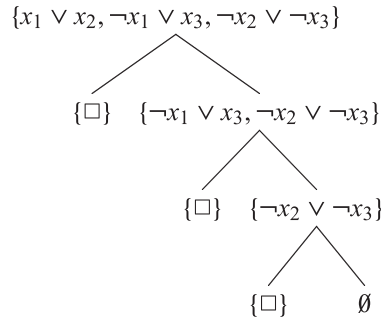
Let  $T$  be a clause MinSAT tableau for a multiset of clauses  $\phi$ , and let  $B$  be a branch of  $T$ . Branch  $B$  is saturated iff its leaf node is labelled with the empty multiset or with a multiset of empty clauses. Tableau  $T$  is completed iff all its branches are saturated. The cost of a saturated branch is the number of empty clauses in its leaf node. The cost of a completed clause MinSAT tableau is the maximum cost among all its branches.

#### EXAMPLE 5.4

Let  $\phi_1 = \{\neg x_1, \neg x_2, x_1 \vee x_2\}$  and  $\phi_2 = \{x_1 \vee x_2, \neg x_1 \vee x_3, \neg x_2 \vee \neg x_3\}$  be multisets of clauses. Figures 4 and 5 show completed clause MinSAT tableaux for  $\phi_1$  and  $\phi_2$ , respectively. The leaf nodes of the branches of the tableau for  $\phi_1$  have at most cost 2 and of the tableau for  $\phi_2$  have at most cost 1. Therefore, the maximum number of clauses that can be falsified in  $\phi_1$  is 2 and in  $\phi_2$  is 1. Note that  $\phi_2$  is satisfiable.

#### 5.1 Soundness and completeness of clause MinSAT tableaux

We first prove that the extension rule preserves the maximum number of unsatisfied clauses among the branches of a clause MinSAT tableau and then the soundness and completeness of the clause MinSAT tableau calculus.

FIGURE 5. Completed clause MinSAT tableau for  $\phi_2 = \{x_1 \vee x_2, \neg x_1 \vee x_3, \neg x_2 \vee \neg x_3\}$ .

## LEMMA 5.5

Let  $\phi = \phi' \cup \{l_1 \vee \dots \vee l_r\}$  be a multiset of clauses, and let  $\text{minsat}(\psi)$  denote the maximum number of clauses that can be falsified in the multiset of clauses  $\psi$ . It holds that  $\text{minsat}(\phi) = \max(\text{minsat}(\phi'), \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r}))$ .

PROOF. We first prove the following inequalities:  $\text{minsat}(\phi) \geq \text{minsat}(\phi')$  and  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r})$ .

Since  $\phi' \subset \phi$  and the maximum number of clauses that can be falsified in every subset of  $\phi$  cannot be greater than the maximum number of clauses that can be falsified in  $\phi$ , it holds that  $\text{minsat}(\phi) \geq \text{minsat}(\phi')$ .

Assume that there is an optimal assignment  $I'$  of  $\phi_{\neg l_1 | \dots | \neg l_r}$  that falsifies more clauses than an optimal assignment  $I$  of  $\phi$ . Then, we could extend  $I'$  by setting  $I'(l_i) = 0$  for  $i = 1, \dots, r$  and get an optimal assignment of  $\phi$  that falsifies more clauses than  $I$ ; if we restore the occurrences of the literals  $l_1, \dots, l_r$  in the clauses of  $\phi_{\neg l_1 | \dots | \neg l_r}$  in which such literals were eliminated when  $\neg l_1, \dots, \neg l_r$  were instantiated in  $\phi$ , we get a multiset  $\phi''$  such that  $\phi'' \subseteq \phi$ . It holds that the number of clauses that  $I'$  falsifies in  $\phi_{\neg l_1 | \dots | \neg l_r}$  and  $\phi''$  is the same because  $I'$  falsifies the added literals, but this is in contradiction with  $I$  being optimal. Therefore,  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r})$ .

Taking into account the previous inequalities, we prove that  $\text{minsat}(\phi) = \max(\text{minsat}(\phi'), \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r}))$ . Let  $I$  be an optimal assignment of  $\phi$ . We distinguish two cases:

- i)  $I$  satisfies  $l_1 \vee \dots \vee l_r$ . Then,  $I$  falsifies the same clauses in  $\phi$  and  $\phi'$  and is also an optimal assignment of  $\phi'$  because  $\text{minsat}(\phi) \geq \text{minsat}(\phi')$ . Since  $\text{minsat}(\phi) = \text{minsat}(\phi')$  and  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r})$ , it follows that  $\text{minsat}(\phi) = \max(\text{minsat}(\phi'), \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r}))$ .
- ii)  $I$  falsifies  $l_1 \vee \dots \vee l_r$ . Then,  $I$  sets  $l_1, \dots, l_r$  to 0, and  $I$  falsifies the same number of clauses in  $\phi$  and  $\phi_{\neg l_1 | \dots | \neg l_r}$ . Since  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r})$ , it follows that  $I$  is also an optimal assignment of  $\phi_{\neg l_1 | \dots | \neg l_r}$ . Since  $\text{minsat}(\phi) = \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r})$  and  $\text{minsat}(\phi) \geq \text{minsat}(\phi')$ , it follows that  $\text{minsat}(\phi) = \max(\text{minsat}(\phi'), \text{minsat}(\phi_{\neg l_1 | \dots | \neg l_r}))$ .  $\square$

## THEOREM 5.6

**Soundness.** Let  $\phi$  be a multiset of clauses, and let  $T$  be a completed clause MinSAT tableau for  $\phi$  that has cost  $m$ . Then, the maximum number of clauses that can be falsified in  $\phi$  is  $m$ .

PROOF. The clause MinSAT tableau  $T$  was obtained by creating a sequence of clause MinSAT tableaux  $T_0, \dots, T_n$  ( $n \geq 0$ ) such that  $T_0$  is an initial tableau for  $\phi$ ,  $T_n = T$ , and  $T_i$  was obtained by a single application of the extension rule on a leaf node of a branch of  $T_{i-1}$  for  $i = 1, \dots, n$ . Assume that  $I$  is an optimal assignment of  $\phi$  that falsifies  $k$  clauses, where  $k \neq m$ . By induction on  $n$ , we

prove that the maximum number of clauses that  $I$  falsifies among the leaf nodes of the branches of  $T_0, \dots, T_n$  (and in particular of  $T$ ) is  $k$ :

Basis:  $T_0$  has a single branch with one node labelled with the clauses of  $\phi$ . So,  $I$  falsifies  $k$  clauses in  $T_0$ , and  $k$  is the maximum number of clauses that can be falsified in  $T_0$ .

Inductive step: assume that the maximum number of clauses that  $I$  falsifies among the leaf nodes of the branches of  $T_{i-1}$  is  $k$ . We prove that the maximum number of clauses that  $I$  falsifies among the leaf nodes of the branches of  $T_i$  is also  $k$ .

$T_i$  was constructed from  $T_{i-1}$  by applying the extension rule on a branch  $B$  of  $T_{i-1}$ . If  $I$  falsifies  $k$  clauses of the leaf node of  $B$ , by Lemma 1, the maximum number of clauses that  $I$  falsifies among the branches of  $T_i$  remains  $k$ . If  $I$  falsifies  $r$  clauses of the leaf node of  $B$ , where  $r < k$ , then  $I$  falsifies  $k$  clauses of the leaf node of a branch  $B'$  of  $T_i$  ( $B' \neq B$  and  $B'$  is also a branch of  $T_{i-1}$ ), and  $I$  cannot falsify more than  $k$  clauses in the leaf nodes of any of the two branches derived from  $B$  because otherwise we could define an assignment that falsifies more than  $k$  clauses of the leaf node of  $B$ .

We proved that the maximum number of clauses that  $I$  falsifies among the leaf nodes of the branches of  $T_0, \dots, T_n$ —and in particular of  $T$ —is  $k$ , but this is in contradiction with  $T$  being a completed clause MinSAT tableau for  $\phi$  that has cost  $m$ . Since  $T$  is completed and has cost  $m$ , the leaf nodes are labelled with either a multiset of empty clauses or the empty formula, and there is at least a branch  $B$  whose leaf node is a multiset with  $m$  empty clauses. So, the maximum number of clauses that can be falsified in the leaf node of  $B$  is  $m$  (and not  $k$ ) and is, at most,  $m$  in the rest of leaf nodes of branches of  $T$ . Hence, the maximum number of clauses that can be falsified in  $\phi$  is  $m$ .  $\square$

#### THEOREM 5.7

**Completeness.** Let  $\phi$  be a multiset of clauses whose maximum number of clauses that can be falsified in  $\phi$  is  $m$ . Then, any completed clause MinSAT tableau for  $\phi$  has cost  $m$ .

PROOF. Each clause MinSAT tableau  $T$  for  $\phi$  can be completed after a finite number of steps. This follows from the fact that the extension rule either eliminates one clause or replaces one clause with an empty clause at each application of the rule. Moreover, the instantiation of literals neither increases the number of clauses nor increases the number of literals per clause. Thus, after a finite number of applications of the extension rule,  $T$  is transformed into a completed clause MinSAT tableau.

Assume that there is a completed clause MinSAT tableau  $T$  for  $\phi$  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 leaf node of  $B$  has  $k$  empty clauses, and each empty clause is derived from a clause of  $\phi$ ; let  $C_1, \dots, C_k$  be such clauses. We define an assignment  $I$  of  $\phi$  as follows:  $I(x) = 1$  ( $I(x) = 0$ ) if  $\neg x$  ( $x$ ) occurs in  $\{C_1, \dots, C_k\}$ , and  $I(x) = 0$  if variable  $x$  does not occur in  $\{C_1, \dots, C_k\}$ . Note that  $\{C_1, \dots, C_k\}$  only contain literals with both the same variable and polarity because the corresponding literals with opposite polarity occur in clauses that were eliminated. Assignment  $I$  falsifies at least  $k$  clauses of  $\phi$  because each literal occurring in  $\{C_1, \dots, C_k\}$  is unsatisfied by  $I$ . Since  $k > m$ , this is in contradiction with  $m$  being the maximum number of clauses that can be falsified in  $\phi$ .

(ii)  $T$  has no branch of cost  $m$ . This is in contradiction with  $m$  being the maximum number of clauses that can be falsified in  $\phi$ . Since an optimal assignment falsifies  $m$  clauses of the initial tableau and the leaf nodes of a completed clause MinSAT tableau are labelled with either a multiset of empty clauses or the empty formula, by Lemma 5.5,  $T$  must have a branch of cost  $m$ .  $\square$

## 14 Clause Tableaux for Maximum and Minimum Satisfiability

From the proof of Theorem 5.7 it follows that, for building an optimal assignment  $I$  from a completed tableau, we have to consider a branch with a maximum number of empty clauses in its leaf node and identify the input clauses that became empty. Then, for each one of such clauses, say  $l_1 \vee \dots \vee l_m$ , we define  $I(l_i) = 0$  for  $i = 1, \dots, m$ , and the variables that do not appear in such clauses can be set to an arbitrary value. For example, in the tableau for  $\phi_1 = \{\neg x_1, \neg x_2, x_1 \vee x_2\}$  of Figure 4, the input clauses that became empty in the branch with maximum cost are  $\{\neg x_1, \neg x_2\}$ , and therefore,  $I(x_1) = I(x_2) = 1$  is an optimal assignment of  $\phi_1$ .

### 5.2 Clause tableaux for WMinSAT and WPMInSAT

We presented clause tableaux for unweighted MinSAT to keep the description as simple as possible. We now describe how the clause MinSAT tableau calculus can be extended to deal with WMinSAT and WPMInSAT instances.

In the case of WMinSAT, we use the same tableau rules but keeping the weights of the clauses. Note that the extension rule either removes clauses or eliminates literals. When a literal is eliminated from a clause, the shortened clause maintains the same weight. In addition, we also could need to collapse several weighted clauses of the form  $(C, w_1), \dots, (C, w_k)$  into a single weighted clause  $(C, w_1 + \dots + w_k)$ .

#### EXAMPLE 5.8

Let  $\phi = \{(x_1, 1), (\neg x_2, 3), (x_1 \vee \neg x_2, 5), (x_1 \vee \neg x_3, 2), (x_2 \vee x_3, 1)\}$  be a multiset of weighted clauses. Figure 6 shows a completed clause WMinSAT tableau  $T$  for  $\phi$ . The leaf nodes of the branches of  $T$  have at most cost 11. Therefore, the maximum sum of the weights of the clauses that can be falsified in  $\phi$  is 11.

In the case of WPMInSAT, we must first derive an equivalent WMinSAT instance and then solve the derived instance as explained above. We will assume that there is an assignment that satisfies all the hard clauses, since otherwise no feasible solution exists.

Given a WPMInSAT instance  $\phi$  whose number of hard clauses is  $\#hard$  and whose sum of the weights of all its soft clauses is  $w$ , we derive a WMinSAT instance  $\phi'$  by adding (i) all the soft clauses in  $\phi$  and (ii) the soft clauses  $(\neg l_1, w + 1), (l_1 \vee \neg l_2, w + 1), \dots, (l_1 \vee l_2 \vee \dots \vee \neg l_k, w + 1)$  for each hard clause  $h = l_1 \vee l_2 \vee \dots \vee l_k$  in  $\phi$ .

Observe that an assignment  $I$  satisfies  $h$  iff  $I$  falsifies exactly one clause among  $\neg l_1, l_1 \vee \neg l_2, \dots, l_1 \vee l_2 \vee \dots \vee \neg l_k$ , or equivalently,  $I$  falsifies  $h$  iff  $I$  satisfies all these clauses. Since the clauses derived from hard clauses have weight  $w + 1$  and we assumed that the hard part of  $\phi$  is satisfiable, every optimal solution of  $\phi'$  falsifies exactly one clause derived from a hard clause and is also an optimal solution of  $\phi$ . Besides, if the maximum sum of the weights of the unsatisfied clauses in  $\phi'$  is  $m$ , then the maximum sum of the weights of the unsatisfied clauses in  $\phi$  is  $m - \#hard \times (w + 1)$ . The treatment of hard clauses in MinSAT tableaux is not as in MaxSAT tableaux, where it is enough to add the weight  $w + 1$  to each hard clause and solve the resulting WMaxSAT instance.

## 6 Clause tableaux for MaxSAT and MinSAT

After defining a clause tableau calculus for MaxSAT and another for MinSAT, a natural question to ask is whether there exists a calculus that is valid for both MaxSAT and MinSAT. In this section, we propose a tableau calculus for MaxSAT and MinSAT that preserves adequately the number of unsatisfied clauses in the generated subproblems. The leaf nodes of a completed tableau contain a

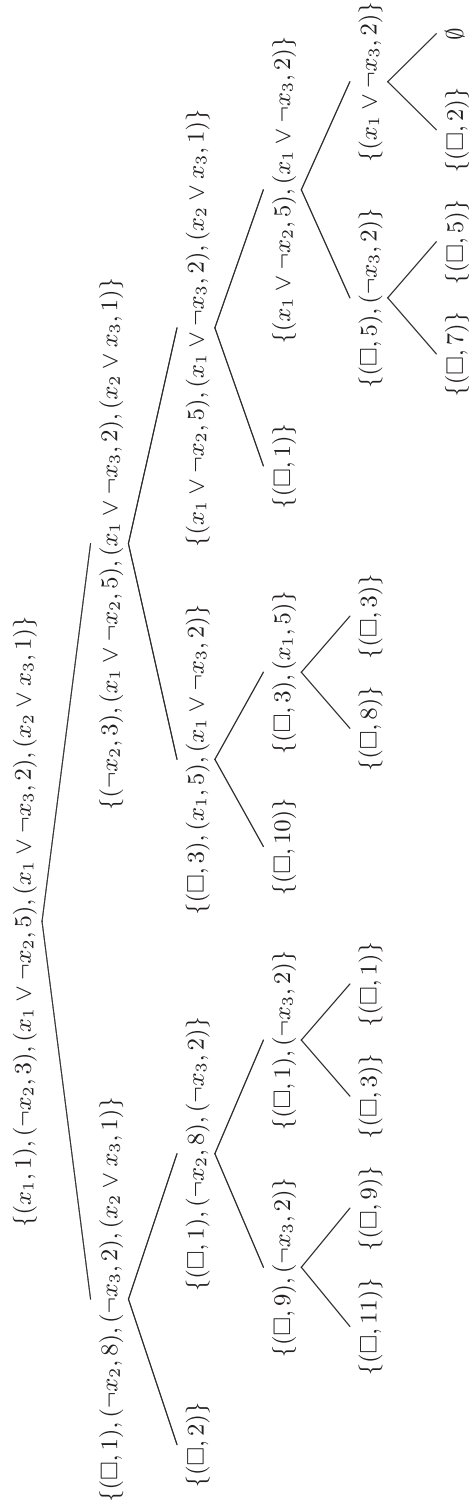
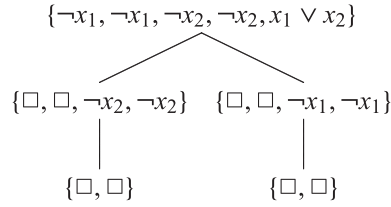


FIGURE 6. Completed clause WMinSAT tableau for  $\phi = \{(x_1, 1), (\neg x_2, 3), (x_1 \vee \neg x_2, 5), (x_1 \vee \neg x_3, 2), (x_2 \vee x_3, 1)\}$ .

FIGURE 7. Proof tree for  $\{\neg x_1, \neg x_1, \neg x_2, \neg x_2, x_1 \vee x_2\}$  using SAT clause branching.

number of empty clauses ranging between the minimum and the maximum number of unsatisfied clauses in the input formula, and there is at least one branch with the minimum value and at least one branch with the maximum value. This scheme also generates optimal MaxSAT and MinSAT assignments by inspecting the optimal branches.

A natural extension of clause tableaux is to perform some kind of local inference at each node, as in fact MinSAT tableaux do in one of the branches when assigning the variables occurring in a select clause to force its violation. In SAT, assuming that the initial tableau contains a single node with the input set of clauses, we can introduce local inference by applying the following extension rule: if the leaf node of a branch  $B$  is labelled with the set of clauses  $\phi = \phi' \cup \{l_1 \vee \dots \vee l_k\}$ , then append  $k$  sibling nodes below  $B$  labelled with  $\phi_{l_1}, \dots, \phi_{l_k}$ . Then, the input set of clauses is unsatisfiable iff the empty clause has been derived in each branch. Actually, if the local inference applied is unit propagation instead of the instantiation of literal  $l_i$  (i.e.,  $\phi_{l_i}$ ), also known as unit clause rule, we get the DPLL procedure [13] with clause branching [18].

Unfortunately, the outlined SAT approach is unsound for MaxSAT and MinSAT because it does not preserve neither the maximum nor the minimum number of unsatisfied clause. For example, if we consider the multiset of clauses  $\phi = \{\neg x_1, \neg x_1, \neg x_2, \neg x_2, x_1 \vee x_2\}$ , the generated tableau has two leaf nodes with two empty clauses, as Figure 7 shows. Note that we first branch on  $x_1$  (labelling the node with  $\phi_{x_1}$ ) and  $x_2$  (labelling the node with  $\phi_{x_2}$ ) and then instantiate  $\neg x_2$  in the left branch and  $\neg x_1$  in the right branch. However, the MaxSAT solution of  $\phi$  is one and the MinSAT solution is four. Also note that branching on  $l_1, \dots, l_k$  is sound in MaxSAT tableaux but becomes unsound when local inference is added to each node.

The key point to integrate MaxSAT and MinSAT is to define an extension rule that preserves both the maximum and the minimum number of unsatisfied clause. This is what we do in the clause MaxMinSAT tableau defined below.

#### DEFINITION 6.1

A clause MaxMinSAT tableau is a finite tree whose nodes are labelled with multisets of clauses. A branch is a maximal path in a tree.

#### DEFINITION 6.2

Let  $\phi$  be a multiset of clauses. A clause MaxMinSAT tableau for  $\phi$  is constructed by a sequence of applications of the following rules:

- Initialize** A tree with a single branch with a single node labelled with the multiset of clauses  $\phi$  is a clause MaxMinSAT tableau for  $\phi$ . Such a tableau is called initial tableau.
- Extension** Given a clause tableau  $T$  for  $\phi$ , and a branch  $B$  of  $T$  whose leaf node is labelled with a multiset  $\phi = \phi' \cup \{l_1 \vee \dots \vee l_k\}$ , the tableau obtained by appending  $k + 1$  new left nodes below  $B$  labelled with the multisets  $\phi_{l_1}, \dots, \phi_{l_k}, \phi_{\neg l_1, \dots, \neg l_k}$  is a clause MaxMinSAT tableau for  $\phi$ .

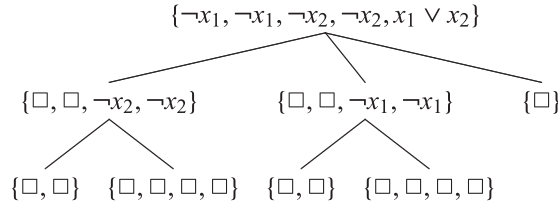


FIGURE 8. A completed clause MaxMinSAT tableau for  $\{\neg x_1, \neg x_1, \neg x_2, \neg x_2, x_1 \vee x_2\}$ .

### DEFINITION 6.3

Let  $T$  be a clause MaxMinSAT tableau for a multiset of clauses  $\phi$ , and let  $B$  be a branch of  $T$ . Branch  $B$  is saturated iff its leaf node is labelled with the empty multiset or with a multiset of empty clauses. Tableau  $T$  is completed iff all its branches are saturated. The cost of a saturated branch is the number of empty clauses in its leaf node. The MaxSAT cost of a completed clause MaxMinSAT tableau is the minimum cost among all its branches, and the MinSAT cost of a completed clause MaxMinSAT tableau is the maximum cost among all its branches.

We prove below that the branches with MaxSAT and MinSAT costs provide optimal MaxSAT and MinSAT solutions, respectively.

### EXAMPLE 6.4

We consider again the multiset of clauses  $\phi = \{\neg x_1, \neg x_1, \neg x_2, \neg x_2, x_1 \vee x_2\}$  of Figure 7. Figure 8 displays a completed clause MaxMinSAT tableau for  $\phi$ . The MaxSAT cost of the tableau is 1 and the MinSAT cost is 4. Hence, the minimum number of clauses that can be falsified in  $\phi$  is 1, and the maximum number is 4.

The next two lemmas prove that the MaxMinSAT extension rule preserves both the maximum and the minimum number of unsatisfied clauses. In other words, we prove its soundness. Based on these results, we then prove the completeness of the clause MaxMinSAT tableau calculus.

### LEMMA 6.5

**MaxSAT clause branching.** Let  $\phi$  be a multiset of clauses, let  $l_1 \vee \dots \vee l_k$  be a clause of  $\phi$  and let  $\text{maxsat}(\phi)$  be the minimum number of unsatisfied clauses in  $\phi$ . Then,

$$\text{maxsat}(\phi) = \min(\text{maxsat}(\phi_{l_1}), \dots, \text{maxsat}(\phi_{l_k}), \text{maxsat}(\phi_{\neg l_1, \dots, \neg l_k})). \quad (1)$$

**PROOF.** Let  $I$  be an optimal MaxSAT assignment. We prove that the minimum number of unsatisfied clauses in  $\phi$  is the same as the minimum number of unsatisfied clauses in at least one of the multisets  $\phi_{l_1}, \dots, \phi_{l_k}, \phi_{\neg l_1, \dots, \neg l_k}$  and is not smaller in the rest of multisets. We distinguish two cases:

1.  $I$  satisfies  $l_1 \vee \dots \vee l_k$ : if  $I$  satisfies  $l_i$ ,  $1 \leq i \leq k$ , then  $\text{maxsat}(\phi) = \text{maxsat}(\phi_{l_i})$  because deleting the clauses containing  $l_i$  and removing the occurrences of  $\neg l_i$  preserve the number of unsatisfied clauses between  $\phi$  and  $\phi_{l_i}$  for every assignment. Since at least one literal  $l_i$  is satisfied by  $I$ , the minimum number of unsatisfied clauses is preserved in one of the derived multisets.
2. If  $I$  does not satisfy  $l_j$ ,  $1 \leq j \leq k$ , then  $\text{maxsat}(\phi) \leq \text{maxsat}(\phi_{l_j})$ . Assume that there exists an assignment  $I'$  of  $\phi_{l_j}$  that falsifies less than  $\text{maxsat}(\phi)$  clauses. If we extend  $I'$  by assigning  $I'(l_j) = \text{true}$ , we get an assignment of  $\phi$  that falsifies less than  $\text{maxsat}(\phi)$  clauses. But this is in contradiction with  $I$  being optimal. So, it holds that  $\text{maxsat}(\phi) \leq \text{maxsat}(\phi_{l_j})$ .

18 *Clause Tableaux for Maximum and Minimum Satisfiability*

3. Finally, we have to prove that  $\text{maxsat}(\phi) \leq \text{maxsat}(\phi_{\neg l_1, \dots, \neg l_k})$ . Observe that  $\text{maxsat}(\phi_{\neg l_1, \dots, \neg l_k}) = \text{maxsat}(\dots(\text{maxsat}(\text{maxsat}(\phi_{\neg l_1})_{\neg l_2}) \dots)_{\neg l_k})$ . The literals  $\neg l_i$ ,  $1 \leq i \leq k$ , which are satisfied by  $I$  preserve the number of unsatisfied clauses, and the literals  $\neg l_j$ ,  $1 \leq j \leq k$ , which are not satisfied by  $I$ , can increase the number of unsatisfied clauses. Since  $I$  satisfies  $l_1 \vee \dots \vee l_k$ ,  $I$  does not satisfy at least one literal in  $\{\neg l_1, \dots, \neg l_k\}$ , and therefore,  $\text{maxsat}(\phi) \leq \text{maxsat}(\phi_{\neg l_1, \dots, \neg l_k})$ .
4. The last two cases guarantee that the number of unsatisfied clauses does not decrease in any case.
5.  $I$  does not satisfy  $l_1 \vee \dots \vee l_k$ : in this case,  $I$  does not satisfy any literal in the clause. As shown above,  $\text{maxsat}(\phi) \leq \text{maxsat}(\phi_{l_i})$  for every  $i$ ,  $1 \leq i \leq k$ . On the other hand,  $\text{maxsat}(\phi) = \text{maxsat}(\phi_{\neg l_1, \dots, \neg l_k})$  because  $I$  satisfies  $\neg l_1, \dots, \neg l_k$ , and deleting the clauses containing  $\neg l_j$  and removing the occurrences of  $l_j$  preserve the number of unsatisfied clauses in this case.  $\square$

## LEMMA 6.6

**MinSAT clause branching.** Let  $\phi$  be a multiset of clauses, let  $l_1 \vee \dots \vee l_k$  be a clause of  $\phi$  and let  $\text{minsat}(\phi)$  be the maximum number of unsatisfied clauses in  $\phi$ . Then,

$$\text{minsat}(\phi) = \max(\text{minsat}(\phi_{l_1}), \dots, \text{minsat}(\phi_{l_k}), \text{minsat}(\phi_{\neg l_1, \dots, \neg l_k})). \quad (2)$$

PROOF. Let  $I$  be an optimal MinSAT assignment. We can prove, using the arguments of Lemma 6.5, that the maximum number of unsatisfied clauses in  $\phi$  is the same as the maximum number of unsatisfied clauses in at least one of the multisets  $\phi_{l_1}, \dots, \phi_{l_k}, \phi_{\neg l_1, \dots, \neg l_k}$  and is not greater in the rest of multisets. We have to take into account the following facts: (i) if  $I$  satisfies a literal  $l_i$ ,  $1 \leq i \leq k$ , then  $\phi_{l_i}$  clearly preserves the number of unsatisfied clauses and so  $\text{minsat}(\phi) = \text{minsat}(\phi_{l_i})$ . (ii) If  $I$  does not satisfy  $l_j$ ,  $1 \leq j \leq k$ , then  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{l_j})$ . Assume that there exists an assignment  $I'$  of  $\phi_{l_j}$  that falsifies more than  $\text{minsat}(\phi)$  clauses. If we extend  $I'$  by assigning  $I'(l_j) = \text{true}$ , we get an assignment of  $\phi$  that falsifies more than  $\text{minsat}(\phi)$  clauses. But this is in contradiction with  $I$  being optimal. So, it holds that  $\text{minsat}(\phi) \geq \text{minsat}(\phi_{l_j})$ .  $\square$

## THEOREM 6.7

A completed clause MaxMinSAT tableau for a multiset of clauses  $\phi$  provides an optimal MaxSAT assignment and an optimal MinSAT assignment.

PROOF. The root node of a completed MaxMinSAT tableau is labelled with the input formula. Such a tableau is finite because each descendant has at least one variable less than its parents: at least one less variable in the descendants labelled with  $\phi_{l_1}, \dots, \phi_{l_k}$  and at least  $k$  less variables in the descendants labelled with  $\phi_{\neg l_1, \dots, \neg l_k}$ . Thus, after a finite number of steps, the leaves of a tableau contain either the empty formula or a multiset of empty clauses. By Lemma 6.5, all the branches with the minimum number of empty clauses correspond to optimal MaxSAT solutions. By Lemma 6.6, all the branches with the maximum number of empty clauses correspond to optimal MinSAT solutions. Besides, Lemmas 6.5 and 6.6 guarantee that the cost of each branch ranges between the maximum and minimum number of unsatisfied clauses in  $\phi$ .

The clauses became empty because of the literals that were instantiated at each node that allowed to remove the complementary literals. Thus, in the optimal branches, the assignments that set those literals to true and the rest of literals appearing in the input formula to an arbitrary value are optimal assignments.  $\square$

All the results of this section also hold if we replace the proposed extension rule with the following rule: *if the leaf node of a branch  $B$  is labelled with the multiset  $\phi = \phi' \cup \{l_1 \vee \dots \vee l_k\}$ , then append*

$k + 1$  sibling nodes below  $B$  labelled with  $\phi_{l_1}, \phi_{\neg l_1, l_2}, \dots, \phi_{\neg l_1, \dots, \neg l_{k-1}, l_k}$  and  $\phi_{\neg l_1, \dots, \neg l_k}$ . This result can be proved with the same arguments of Lemmas 6.5 and 6.6.

The extension of our approach to weighted MaxSAT/MinSAT and weighted partial MaxSAT/MinSAT is as in branch-and-bound MaxSAT and MinSAT algorithms [2, 24, 28]. Roughly speaking, in weighted MaxSAT/MinSAT, we just need to propagate weights, and given a multiset of empty weighted clauses  $\{(\square, w_1), \dots, (\square, w_k)\}$ , we replace it with  $\{(\square, w_1 + \dots + w_k)\}$ . In weighted partial MaxSAT/MinSAT, we can apply the same inference as in SAT in the hard part.

## 7 Conclusions

We have defined three complete logical calculi for MaxSAT and MinSAT. The first calculus is valid for MaxSAT, the second is valid for MinSAT and the third is valid for both MaxSAT and MinSAT. One interesting aspect of these calculi is that they allow one to solve optimization instead of decision problems using logical tools and look at MaxSAT and MinSAT from a different angle.

As future work we plan to analyse how the proposed tableau calculi can be extended to non-clausal MaxSAT and MinSAT, as well as to deal with first-order logic formulas. We also plan to study how the calculi can be extended to many-valued logics. The results of this paper inspired a natural deduction MaxSAT calculus [11].

## Acknowledgements

This work was supported by Project LOGISTAR from the EU H2020 Research and Innovation Programme under Grant Agreement No. 769142 and MINECO-FEDER projects RASO (TIN2015-71799-C2-1-P/2-P).

## References

- [1] A. Abramé and D. 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*, pp. 336–343. IEEE Computer Society, 2014.
- [2] A. Abramé and D. 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*, pp. 268–274. IJCAI, 2015.
- [3] T. Alsinet, Felip M. and J. Planes. A Max-SAT solver with lazy data structures. In *Proceedings of the 9th Ibero-American Conference on Artificial Intelligence, IBERAMIA 2004, Puebla, México*, pp. 334–342. Lecture Notes in Computer Science, vol. 3315. Springer, 2004.
- [4] C. Ansótegui, J. Gabàs and J. Levy. Exploiting subproblem optimization in SAT-based MaxSAT algorithms. *Journal of Heuristics*, **22**, 1–53, 2016.
- [5] C. Ansótegui, I. Izquierdo, F. Manyà and J. Torres Jiménez. A Max-SAT-based approach to constructing optimal covering arrays. In *Proceedings of the 16th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2013, Vic, Spain*. Frontiers in Artificial Intelligence and Applications, vol. 256, pp. 51–59. IOS Press, 2013.
- [6] C. Ansótegui, C. M. Li, F. Manyà and Z. Zhu. A SAT-based approach to MinSAT. In *Proceedings of the 15th International Conference of the Catalan Association for Artificial Intelligence, CCIA-2012, Alacant, Spain*, pp. 185–189. IOS Press, 2012.
- [7] J. Argelich, C. M. Li, F. Manyà and J. Planes. The first and second max-SAT evaluations. *Journal on Satisfiability, Boolean Modeling and Computation*, **4**, 251–278, 2008.

- [8] J. Argelich, C. M. Li, F. Manyà and J. R. Soler. Clause branching in MaxSAT and MinSAT. In *Proceedings of the 21st International Conference of the Catalan Association for Artificial Intelligence, CCIA 2018, Alt Empordà, Spain*, pp. 17–26. IOS Press, 2018.
- [9] M. L. Bonet, J. Levy and F. 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*. Lecture Notes in Computer Science, vol. 4121, pp. 240–251. Springer, 2006.
- [10] M. L. Bonet, J. Levy and F. Manyà. Resolution for max-SAT. *Artificial Intelligence*, **171**, 240–251, 2007.
- [11] J. Casas-Roma, A. Huertas and F. Manyà. Solving MaxSAT with natural deduction. In *Proceedings of the 20th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2017, Deltebre, Spain*, pp. 186–195. IOS Press, 2017.
- [12] M. D’Agostino. Tableaux methods for classical propositional logic. In *Handbook of Tableau Methods*, M. D’Agostino, D. Gabbay, R. Hähnle and J. Posegga, eds, pp. 45–123. Kluwer, 1999.
- [13] M. Davis, G. Logemann and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, **5**, 394–397, 1962.
- [14] M. C. Fitting. *First-Order Logic and Automated Theorem Proving*, 2nd edn. Springer, 1996.
- [15] F. Zhaohui and S. Malik. On solving the partial MAX-SAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT-2006, Seattle, USA*. Lecture Notes in Computer Science, vol. 4121, pp. 252–265. Springer, 2006.
- [16] R. Hähnle. Tableaux and related methods. In *Handbook of Automated Reasoning*, J. A. Robinson and A. Voronkov, eds, pp. 100–178. Elsevier and MIT Press, 2001.
- [17] F. Heras, J. Larrosa and A. Oliveras. MiniMaxSAT: an efficient weighted max-SAT solver. *Journal of Artificial Intelligence Research*, **31**, 1–32, 2008.
- [18] J. N. Hooker and V. Vinay. Branching rules for satisfiability. *Journal of Automated Reasoning*, **15**, 359–383, 1995.
- [19] A. Ignatiev, A. Morgado, J. Planes and J. Marques-Silva. Maximal falsifiability. *AI Communications*, **29**, 351–370, 2016.
- [20] J. Larrosa and F. Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-2005, Edinburgh, Scotland*, pp. 193–198. Morgan Kaufmann, 2005.
- [21] C. M. Li and F. Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, A. Biere, H. van Maaren and T. Walsh, eds, pp. 613–631. IOS Press, 2009.
- [22] C. M. Li and F. Manyà. An exact inference scheme for MinSAT. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI-2015, Buenos Aires, Argentina*, pp. 1959–1965. IJCAI, 2015.
- [23] C. M. Li, F. Manyà, N. O. Mohamedou and J. Planes. Resolution-based lower bounds in MaxSAT. *Constraints*, **15**, 456–484, 2010.
- [24] C. M. Li, F. Manyà and J. Planes. New inference rules for max-SAT. *Journal of Artificial Intelligence Research*, **30**, 321–359, 2007.
- [25] C. M. Li, F. Manyà and J. R. Soler. A clause tableau calculus for MinSAT. In *Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2016, Barcelona, Spain*. Frontiers in Artificial Intelligence and Applications, vol. 288, pp. 88–97. IOS Press, 2016.
- [26] C. M. Li, F. Manyà and J. R. Soler. A clause tableaux calculus for MaxSAT. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI-2016, New York, USA*, pp. 766–772. IJCAI, 2016.

- [27] C. M. Li, Z. Zhu, F. Manyà and L. Simon. Minimum satisfiability and its applications. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-2011, Barcelona, Spain*, pp. 605–610. IJCAI, 2011.
- [28] C. M. Li, Z. Zhu, F. Manyà and L. Simon. Optimizing with minimum satisfiability. *Artificial Intelligence*, **190**, 32–44, 2012.
- [29] H. Lin and S. Kaile. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India*, pp. 2334–2339. IJCAI, 2007.
- [30] I. Lynce, V. M. Manquinho and R. Martins. Parallel maximum satisfiability. In *Handbook of Parallel Constraint Reasoning*, pp. 61–99. Springer, 2018.
- [31] V. M. Manquinho, J. Marques-Silva and J. Planes. Algorithms for weighted Boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*. Lecture Notes in Computer Science, vol. 5584, pp. 495–508. Springer, 2009.
- [32] F. Manyà, S. Negrete, C. Roig and J. R. Soler. A MaxSAT-based approach to the team composition problem in a classroom. In *Autonomous Agents and Multiagent Systems—AAMAS 2017 Workshops, Visionary Papers, São Paulo, Brazil*, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10643, pp. 164–173. Springer, 2017.
- [33] R. Martins, S. Joshi, V. M. Manquinho and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Principles and Practice of Constraint Programming—20th International Conference, CP, Lyon, France*, pp. 531–548. Springer, 2014.
- [34] N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, Canada*, pp. 2717–2723. AAAI Press, 2014.
- [35] R. Smullyan. *First-Order Logic*, 2nd corrected edn. Dover Publications, 1995. First published 1968 by Springer.

Received 10 June 2019