Information Sciences 227 (2013) 43–59

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Resolution procedures for multiple-valued optimization ☆

Carlos Ansótegui [a], María Luisa Bonet [b], Jordi Levy [c], Felip Manyà [c,*]

[a] Computer Science Department, Universitat de Lleida (UdL), Jaume II, 69, 20001 Lleida, Spain
[b] Department of Llenguatges i Sistemes Informàtics (LSI), Universitat Politècnica de Catalunya (UPC), Jordi Girona, 1-3, 08034 Barcelona, Spain
[c] Artificial Intelligence Research Institute (IIIA), Spanish Scientific Research Council (CSIC), Campus UAB, 08193 Bellaterra, Spain

ARTICLE INFO

Article history:
Received 16 May 2011
Received in revised form 19 October 2012
Accepted 2 December 2012
Available online 14 December 2012

Keywords:
Multiple-valued logics
Signed logic
Resolution
Optimization
Maximum satisfiability

ABSTRACT

Signed clausal forms offer a suitable logical framework for automated reasoning in multiple-valued logics. It turns out that the satisfiability problem of any finitely-valued propositional logic, as well as of certain infinitely-valued logics, can be easily reduced, in polynomial time, to the satisfiability problem of signed clausal forms. On the other hand, signed clausal forms are a powerful knowledge representation language for constraint programming, and have shown to be a practical and competitive approach to solving combinatorial decision problems.

Motivated by the existing theoretical and practical results for the satisfiability problem of signed clausal forms, as well as by the recent logical and algorithmic results on the Boolean maximum satisfiability problem, in this paper we investigate the maximum satisfiability problem of propositional signed clausal forms from the logical and practical points of view. From the logical perspective, our aim is to define complete inference systems taking as a starting point the resolution-style calculi defined for the Boolean CNF case. The result is the definition of two sound and complete resolution-style rules, called signed binary resolution and signed parallel resolution for maximum satisfiability. From the practical perspective, our main motivation is to use the language of signed clausal forms along with the newly defined inference systems as a generic approach to solve combinatorial optimization problems, and not just for solving decision problems as so far. The result is the establishment of a link between signed logic and constraint programming that provides a concise and elegant logical framework for weighted constraint programming.

1. Introduction

Multiple-valued logics are one of the central areas of non-classical logic. Apart from philosophical, linguistic, and mathematical motivations for their investigation, they have found many applications in computer science and artificial intelligence [19]. In particular, signed logics have shown to be a formalism particularly well-suited for automated reasoning [14,15,28], and a powerful knowledge representation language for constraint programming [3,6,21].

In this paper we focus on propositional signed clausal forms [5], which are sets of clauses that use a generalized notion of literal, called signed literal. A signed literal is an expression of the form $S\colon p$, where p is a propositional variable and S, its *sign*,

☆ Research partially supported by the Ministerio de Economía y Competividad research projects ARINF TIN2009-14704-C03-01, TASSAT TIN2010-20967-C04-01/03/04, and Newmatica INNPACTO IPT-2011-1496-310000 (funded by MICINN until 2011).
* Corresponding author.
E-mail addresses: carlos@diei.udl.cat (C. Ansótegui), bonet@lsi.upc.edu (M.L. Bonet), levy@iiia.csic.es (J. Levy), felip@iiia.csic.es (F. Manyà).
URLs: http://www.lsi.upc.edu/~bonet (M.L. Bonet), http://www.iiia.csic.es/~levy (J. Levy).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.12.004

is a subset of a domain N. Informally, S: p is true if p takes one of the values in S. As in the Boolean case, conjunction is interpreted with the minimum function and disjunction with the maximum function.

Signed clausal forms are well-known because the satisfiability of any finitely-valued propositional logic, as well as of certain infinitely-valued logics, can be easily reduced, in polynomial time, to the satisfiability problem of signed clausal forms [4,16,17]. This is important because it avoids the development of a devoted satisfiability solver for every multiple-valued logic. On the other hand, signed clausal forms are a simple and expressive language for encoding combinatorial decision problems, and count with competitive satisfiability testing algorithms that extend the techniques developed for Boolean CNF formulae with a very low overhead, but with the advantage of exploiting the structure of domains as in constraint programming [3,6,21,27].

Given the important results and applications achieved with the study of the satisfiability problem of signed clausal forms, our goal here is to investigate the corresponding maximum satisfiability problem from both a logical and a practical point of view. From the logical perspective, our main motivation is to define complete resolution-style calculi taking as a starting point the calculi defined for the maximum satisfiability problem of Boolean CNF formulas (MaxSAT) [8,26]. From the practical perspective, our main motivation is to use signed clausal forms as a language for solving combinatorial optimization problems, and not just for solving decision problems as so far. Our approach will rely on encoding combinatorial optimization problems as instances of the maximum satisfiability problem of signed clausal forms, and use the defined logical calculi to derive an optimal solution of the original problem. Interestingly, we will uncover the link between our approach and the weighted constraint programming approach developed in the constraint programming community to deal with soft constraints [29]. As we will see, signed clausal forms equipped with inference rules for the maximum satisfiability problem provide a concise and elegant logical framework for weighted constraint programming. They allow to formally define and analyze both global and local consistency properties in a natural way.

The contributions of the paper may be summarized as follows: In Section 2 we define signed clausal forms, and the corresponding notions of satisfiability and maximum satisfiability. In Section 3 we define two resolution-style rules, called signed MaxSAT binary resolution and signed MaxSAT parallel resolution. In Section 4 we prove that these rules are sound and complete for the maximum satisfiability problem of signed clausal forms. In Section 5 we establish a link between signed logic and weigthed constraint programming: Section 5.1 defines the weighted constraint satisfaction problem (WCSP) and its encoding as an instance of the signed maximum satisfiability problem. Section 5.2 describes a new complete algorithm for solving WCSP, that we can extract from the completeness proof of the new signed inference systems. Section 5.3 defines a restriction and a generalization of the signed MaxSAT parallel rule called, respectively, signed MaxSAT i-consistency resolution and signed MaxSAT (i,j)-consistency resolution. These rules have the following property: if a WCSP signed encoding is *closed* under signed MaxSAT i-consistency, then the corresponding WCSP is i-consistent, and if it is *closed* under signed MaxSAT (i,j)-consistency, then the WCSP is (i,j)-consistent. Section 5.4 describes an algorithm that applies directional soft consistency with the previous rules, and enforces directional i-consistency. Finally, we give some concluding remarks.

The present paper extend the results of two conference papers [1,2]. The additional results included here are: (a) many more examples to understand the complete inference rules; (b) a characterization of a collection of complete rules of inference of WCSP (see Corollary 22); (c) the analysis of the algorithm to enforce directional i-consistency; (d) introduction of weights in the inference rules to be able to work with the weighted format (see Definitions 10 and 11).

2. Signed clausal forms

Next we will give the basic definitions of a signed CNF formula and the corresponding notions of satisfiability and maximum satisfiability.

Definition 1. Given a propositional variable p, its *domain* $d(p)$ is a non-empty finite set $d(p) = \{i_1, i_2, \ldots, i_n\}$ of truth values.

A sign of a variable is a subset $S \subseteq d(p)$ of its truth values.

A signed literal is an expression of the form S: p, where p is a propositional variable and S is a sign of p.

The complement of a signed literal S: p is defined as $\overline{S} : p = (d(p) \setminus S) : p$.

A signed clause is a disjunction of signed literals. The empty clause, denoted by $\Box$, is a disjunction of zero literals.

A signed CNF formula is a multiset of signed clauses. The empty multiset of clauses is denoted by $\emptyset$.

A weighted signed clause is a pair C, w where C is a signed clause and $w \in \mathbb{N}$. A *weighted signed CNF formula* is a multiset of weighted signed clauses.

Definition 2. An assignment for a signed CNF formula is a mapping that assigns to every propositional variable an element of the truth value set.

An assignment I satisfies a signed literal S: p iff $I(p) \in S$, satisfies a signed clause C iff it satisfies at least one of the signed literals in C, and satisfies a signed CNF formula Γ iff it satisfies all clauses in Γ.

A signed CNF formula is satisfiable iff it is satisfied by at least one assignment; otherwise it is unsatisfiable.

Definition 3. The cost of an assignment for a signed CNF formula is the number of clauses not satisfied by the assignment. The cost of an assignment for a weighted formula is the sum of the weights of the falsified clauses.

The Signed MaxSAT Problem for a signed CNF formula consists in finding an assignment that minimizes the number of falsified signed clauses, i.e. an assignment with minimal cost. Such an assignment will be called *optimal assignment* and its cost, the optimal cost.

Definition 4. A minimal unsatisfiable core of a signed CNF formula Γ is any unsatisfiable subset Γ' of Γ such that, if we remove any clause $C \in \Gamma'$, then $\Gamma' \setminus \{C\}$ is satisfiable.

3. Inference rules for signed MaxSAT

We start by recalling two complete inference rules for solving the satisfiability problem of signed CNF formulae: the first, called signed binary resolution, is a straightforward generalization of the SAT resolution rule. The second is called signed parallel resolution [15].

Second, we give two complete rules for solving the signed MaxSAT problem. The first one, called signed MaxSAT binary resolution, is the natural extension to signed MaxSAT of the signed binary resolution rule. The second one, called signed MaxSAT parallel resolution, is the extension of the signed parallel resolution rule.

Signed binary resolution and signed parallel resolution for the signed SAT problem are defined as follows [15,18]:

Signed binary resolution:

$$\frac{\begin{array}{c} S{:}x \vee A \\ S'{:}x \vee B \end{array}}{S \cap S'{:}x \vee A \vee B} \qquad \frac{\emptyset{:}x \vee D}{D}$$

Signed parallel resolution:

$$\frac{\begin{array}{c} S_1{:}x \vee A_1 \\ \vdots \\ S_k{:}x \vee A_k \end{array}}{A_1 \vee \cdots \vee A_k} \qquad \text{whenever } \bigcap_{i=1}^{k} S_i = \emptyset$$

In the above inference systems we assume w.l.o.g. that every variable in a clause appears only once, collapsing different occurrences of a literal with the same variable by computing the union of the signs.

Example 5. Given the set of variables $X = \{x_1, x_2, x_3\}$ with domains $d(x_1) = d(x_2) = d(x_3) = \{a, b, c\}$, the following are examples of the application of the signed binary and parallel resolution rule (with x_1 as the resolving variable):

Signed binary resolution:

$$\frac{\begin{array}{c} \{a\}{:}x_1 \\ \overline{\{a\}}{:}x_1 \end{array}}{\square} \qquad \frac{\begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_2 \\ \overline{\{a\}}{:}x_1 \end{array}}{\{b\}{:}x_2} \qquad \frac{\begin{array}{c} \overline{\{c\}}{:}x_1 \vee \{b\}{:}x_2 \\ \{b\}{:}x_1 \vee \{a\}{:}x_2 \end{array}}{\{b\}{:}x_1 \vee \overline{\{c\}}{:}x_2}$$

Signed parallel resolution:

$$\frac{\begin{array}{c} \overline{\{a\}}{:}x_1 \\ \overline{\{b\}}{:}x_1 \\ \overline{\{c\}}{:}x_1 \end{array}}{\square} \qquad \frac{\begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_2 \\ \{b\}{:}x_1 \vee \{b\}{:}x_2 \\ \{c\}{:}x_1 \vee \{b\}{:}x_2 \end{array}}{\{b\}{:}x_2} \qquad \frac{\begin{array}{c} \overline{\{a\}}{:}x_1 \vee \{b\}{:}x_2 \\ \overline{\{b\}}{:}x_1 \vee \{b\}{:}x_2 \\ \overline{\{c\}}{:}x_1 \vee \{c\}{:}x_3 \end{array}}{\{b\}{:}x_2 \vee \{c\}{:}x_3}$$

Next we define the signed MaxSAT counterparts of the previous rules. The first thing to notice about the new rules—which can be seen as extension of the Boolean MaxSAT resolution rule [8,23] to the signed framework—is that we substitute premises by conclusions rather than just adding the conclusions. This is necessary to make the rule sound in the optimization context. We will show that the formula formed by the conclusions of the rules have the same optimal cost as the formula formed by the premises. If we *replace* the premises by the conclusions, then the optimal cost of the formula is preserved, but this does not hold if we simply *add* the conclusions without erasing the premises.

The first rule was defined and proved sound and complete in [2].

Definition 6. The **signed MaxSAT binary resolution** rule is defined as follows:

$$\begin{array}{l} S{:}x \vee a_1 \vee \ldots \vee a_s \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \\ \hline\hline S \cap S'{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_t \\ S \cup S'{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_t \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee \overline{b_1} \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \overline{b_2} \\ \vdots \\ S{:}x \vee a_1 \vee \ldots \vee a_s \vee b_1 \vee \ldots \vee b_{t-1} \vee \overline{b_t} \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee \overline{a_1} \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \overline{a_2} \\ \vdots \\ S'{:}x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \ldots \vee a_{s-1} \vee \overline{a_s} \end{array}$$

This inference rule is applied to multisets of clauses, and *replaces* the premises of the rule by its conclusions.

We say that the rule *resolves* the variable x, and x is called the *resolving* variable.

The tautologies concluded by the rule like $N : x \vee A$ are removed from the resulting multiset. Also we can substitute clauses like $S : x \vee S' : x \vee A$ by $(S \cup S') : x \vee A$, and clauses like $\emptyset : x \vee A$ by A. Next we see an example of the use of the rule.

For the sake of space, we can use the following more compact representation:

$$\begin{array}{c} S{:}x \vee A \\ S'{:}x \vee B \\ \hline\hline S \cap S'{:}x \vee A \vee B \\ S \cup S'{:}x \vee A \vee B \\ S{:}x \vee A \vee \overline{B} \\ S'{:}x \vee \overline{A} \vee B \end{array}$$

Notice that $\overline{B}$ where B is a disjunction of signed literals is not in CNF. Thus, an expression of the form $D \vee \overline{E}$, where D is a disjunction of signed literals and E is the disjunction of signed literals $e_1 \vee \ldots \vee e_t$, should be replaced by the following equivalent set of clauses:

$$\begin{array}{l} D \vee \overline{e_1} \\ D \vee e_1 \vee \overline{e_2} \\ \vdots \\ D \vee e_1 \vee \ldots \vee e_{t-1} \vee \overline{e_t} \end{array}$$

The second inference rule is the one that corresponds to the signed parallel resolution rule in the signed MaxSAT framework. It was first defined in [1].

Definition 7. The **signed MaxSAT parallel resolution** rule is defined as follows:

$$\begin{array}{c} S_1{:}x \vee D_1 \\ \vdots \\ S_k{:}x \vee D_k \\ \hline\hline D_1 \vee \ldots \vee D_k \\ \left\{ \begin{array}{r} (S_1 \cap \ldots \cap S_{t-1}) \cup S_t{:}x \vee D_1 \vee \ldots \vee D_t \\ (S_1 \cap \ldots \cap S_{t-1}){:}x \vee D_1 \vee \ldots \vee D_{t-1} \vee \overline{D_t} \\ S_t{:}x \vee \overline{D_1 \vee \ldots \vee D_{t-1}} \vee D_t \end{array} \right\}_{t=2\ldots k} \end{array}$$

where D_i is a disjunction of signed literals, and $\bigcap_{i=1}^{k} S_i = \emptyset$. We call x the *resolving* variable.

As we will see later, the application of this rule corresponds to $k-1$ applications of the signed MaxSAT binary resolution rule, which is the argument for the soundness of the rule. However, in this case we require the intersection of the signs of the resolved variable to be empty. As proved in Theorem 21, this additional restriction does not invalidates completeness. The restriction leads the resolution process to the elimination of the resolved variable.

We could use an alternative definition of this rule. The only difference would be that instead of requiring that $\bigcap_{i=1}^{k} S_i = \emptyset$, we would require that $\{S_1 : x, \ldots, S_k : x\}$ should be a minimal unsatisfiable core. This restriction ensures that the set of clauses used in the premise is minimal. Next we will prove that all these versions of the rule are sound and complete.

Example 8. For three premises, the signed MaxSAT parallel rule has the following form:

$$\begin{array}{c}
S_1{:}x \vee D_1 \\
S_2{:}x \vee D_2 \\
S_3{:}x \vee D_3 \\
\hline\hline
D_1 \vee D_2 \vee D_3 \\
\begin{array}{rl}
S_1 \cup S_2{:}x & \vee D_1 \vee D_2 \\
S_1{:}x & \vee D_1 \vee \overline{D_2} \\
S_2{:}x & \vee \overline{D_2} \vee D_1 \\
(S_1 \cap S_2) \cup S_3{:}x & \vee D_1 \vee D_2 \vee D_3 \\
(S_1 \cap S_2){:}x & \vee D_1 \vee D_2 \vee \overline{D_3} \\
S_3{:}x & \vee \overline{D_1 \vee D_2} \vee D_3
\end{array}
\end{array}$$

where $S_1 \cap S_2 \cap S_3 = \emptyset$

Example 9. Given the set of variables $X = \{x_1, x_2, x_3\}$ with domains $d(x_1) = d(x_2) = d(x_3) = \{a, b, c\}$, the following are examples of the application of the two previous signed MaxSAT resolution rules (with x_1 as the resolving variable):

Signed MaxSAT binary resolution:

$$\begin{array}{c}
\{a\}{:}x_1 \\
\overline{\{a\}}{:}x_1 \\
\hline\hline
\Box
\end{array}
\qquad
\begin{array}{c}
\{a\}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{a\}}{:}x_1 \\
\hline\hline
\{b\}{:}x_2 \\
\overline{\{a\}}{:}x_1 \vee \overline{\{b\}}{:}x_2
\end{array}
\qquad
\begin{array}{c}
\overline{\{c\}}{:}x_1 \vee \{b\}{:}x_2 \\
\{b\}{:}x_1 \vee \{a\}{:}x_2 \\
\hline\hline
\{b\}{:}x_1 \vee \overline{\{c\}}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \overline{\{c\}}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \\
\{b\}{:}x_1 \vee \overline{\{b\}}{:}x_2
\end{array}
\qquad
\begin{array}{c}
\{a\}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{a\}}{:}x_1 \vee \{c\}{:}x_3 \\
\hline\hline
\{b\}{:}x_2 \vee \{c\}{:}x_3 \\
\{a\}{:}x_1 \vee \{b\}{:}x_2 \vee \overline{\{c\}}{:}x_3 \\
\overline{\{a\}}{:}x_1 \vee \{c\}{:}x_3 \vee \overline{\{b\}}{:}x_2
\end{array}$$

Signed MaxSAT parallel resolution:

$$\begin{array}{c}
\overline{\{a\}}{:}x_1 \\
\overline{\{b\}}{:}x_1 \\
\overline{\{c\}}{:}x_1 \\
\hline\hline
\Box
\end{array}
\qquad
\begin{array}{c}
\{a\}{:}x_1 \vee \{b\}{:}x_2 \\
\{b\}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \{b\}{:}x_2 \\
\hline\hline
\{b\}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \{b\}{:}x_2
\end{array}
\qquad
\begin{array}{c}
\overline{\{a\}}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{b\}}{:}x_1 \vee \{b\}{:}x_2 \\
\overline{\{c\}}{:}x_1 \vee \{c\}{:}x_3 \\
\hline\hline
\{b\}{:}x_2 \vee \{c\}{:}x_3 \\
\{c\}{:}x_1 \vee \{b\}{:}x_2 \vee \overline{\{c\}}{:}x_3 \\
\overline{\{c\}}{:}x_1 \vee \overline{\{b\}}{:}x_2 \vee \{c\}{:}x_3
\end{array}$$

Notice that many of the conclusions are tautologies, and according to the definition of the rules we can remove them. This means that a potentially significant number of the conclusions will not appear.

Now, we introduce the weighted versions of the signed MaxSAT binary and parallel resolution rules.

Definition 10. The **weighted signed MaxSAT binary resolution** rule is defined as follows:

$$\begin{array}{l}
S{:}x \vee A, w_1 \\
S'{:}x \vee B, w_2 \\
\hline\hline
S \cap S'{:}x \vee A \vee B, w_{min} \\
S \cup S'{:}x \vee A \vee B, w_{min} \\
S{:}x \vee A \vee \overline{B}, w_{min} \\
S'{:}x \vee \overline{A} \vee B, w_{min} \\
S{:}x \vee A, w_1 - w_{min} \\
S'{:}x \vee B, w_2 - w_{min}
\end{array}$$

where A and B are disjunctions of signed literals, w_1 (w_2) denotes the weight of the first (second) clause, and $w_{min} = min(w_1, w_2)$.

We call x the *resolving variable*.

Definition 11. The **weighted signed MaxSAT parallel resolution** rule is defined as follows:

$$\begin{array}{l}
S_1{:}x \vee D_1, w_1 \\
\vdots \\
S_k{:}x \vee D_k, w_k \\
\hline\hline
D_1 \vee \ldots \vee D_k, w_{min} \\
\left\{ \begin{array}{r}
(S_1 \cap \ldots \cap S_{t-1}) \cup S_t{:}x \vee D_1 \vee \ldots \vee D_t, w_{min} \\
(S_1 \cap \ldots \cap S_{t-1}){:}x \vee D_1 \vee \ldots \vee D_{t-1} \vee \overline{D_t}, w_{min} \\
S_t{:}x \vee \overline{D_1 \vee \ldots \vee D_{t-1}} \vee D_t, w_{min}
\end{array} \right\}_{t=2\ldots k} \\
S_1{:}x \vee D_1, w_1 - w_{min} \\
\vdots \\
S_k{:}x \vee D_k, w_k - w_{min}
\end{array}$$

where D_i is a disjunction of signed literals, $\bigcap_{i=1}^{k} S_i = \emptyset$, w_i denotes the weight of the i^{th} clause, and $w_{min} = min(w_1, \ldots, w_k)$. We call x the *resolving variable*.

4. Soundness and completeness

In the context of MaxSAT rules, a rule is sound if the number of unsatisfied clauses in the premises coincides with the number of unsatisfied clauses in the conclusions for every truth assignment. Recall that applying a rule amounts to replacing the premises by the conclusions.

Theorem 12. *The signed MaxSAT binary resolution rule is sound.*

Proof. Let I be an arbitrary assignment. There are four cases:

1. If I falsifies the two premises, then I also falsifies the first two conclusions, and only them.
2. If I satisfies the two premises, then it also trivially satisfies the last $s + t$ clauses of the conclusion, because they are either implied by one or the other premise. The second clause of the conclusion is implied by each one of the premises. Therefore, it is also satisfied by I.
   The first clause of the conclusion is not implied by the premises. However, if both premises are satisfied then we have two cases. If $S : x$ and $S' : x$ are both satisfied, then so it is $(S \cap S') : x$. Otherwise, either some a_i's or some b_j's is satisfied, thus also the first clause of the conclusion.
3. If I satisfies the first premise, but not the second one, then the second clause of the conclusion as well as the t following clauses are satisfied, because all them are implied by the first premise.
   For the rest of conclusions, there are two cases: If some of the a_i's is satisfied, then let i be the index of such a. The assignment will satisfy the first clause of the conclusion and the last s conclusions, except $S' : x \vee b_1 \vee \ldots \vee b_t \vee a_1 \vee \ldots \vee a_{i-1} \vee \overline{a_i}$ that is falsified. Otherwise none of the a_i's is satisfied, and therefore, $S : x$ is satisfied. Hence, the first conclusion is falsified, and the last s conclusions are satisfied.
4. If I satisfies the second premise, but not the first one, the situation is analogous to previous case. □

We next prove that signed MaxSAT parallel resolution is a sound inference rule.

Definition 13. A rule R is a derived rule from a inference system, if the conclusion of R can be obtained from the premises of R using several steps of the inference system.

Theorem 14. *The signed MaxSAT parallel resolution rule is a derived rule from the signed MaxSAT binary resolution rule.*

Proof. We will show that given the initial k premises, the multiset of conclusions can be obtained by exactly $k - 1$ applications of signed MaxSAT binary resolution. The first application is on the first two premises. Then, for the l-th application, $2 \leqslant l \leqslant k - 1$, one of the premises is the $l + 1$st premise, $S_{l+1} : x \vee D_{l+1}$, and the second premise is the first conclusion of the previous step, $S_1 \cap \ldots \cap S_l : x \vee D_1 \vee \ldots \vee D_l$. Then we replace these two premises by the following conclusions:

$$\begin{array}{l}
S_1 \cap \ldots \cap S_{l+1} : x \vee D_1 \vee \ldots \vee D_{l+1} \\
(S_1 \cap \ldots \cap S_l) \cup S_{l+1} : x \vee D_1 \vee \ldots \vee D_{l+1} \\
(S_1 \cap \ldots \cap S_l) : x \vee D_1 \vee \ldots \vee D_l \vee \overline{D_{l+1}} \\
S_{l+1} : x \vee \overline{D_1 \vee \ldots \vee D_l} \vee D_{l+1}
\end{array}$$

Notice that in the next step the second premise will be the first clause of the above conclusions. The first conclusion of the last step will be $S_1 \cap \ldots \cap S_k : x \vee D_1 \vee \ldots \vee D_k$. Since $S_1 \cap \ldots \cap S_k = \emptyset$, the last application of the rule produces as a first conclusion $D_1 \vee \ldots \vee D_k$. □

Theorem 15. *The signed MaxSAT parallel resolution is sound.*

Proof. We obtain the soundness of the rule noting that it is a derived rule of signed MaxSAT binary resolution. □

We will prove first the completeness of the parallel signed MaxSAT rule. The completeness of the signed MaxSAT rule will follow from Theorem 14 and the completeness of the parallel rule.

The completeness of the rule is similar to the arguments of [1,2,7]. In order to prove the result, we need some definitions and lemmas.

The first definition is the notion of saturation. This notion captures the idea that if a multiset of clauses $\mathcal{C}$ is saturated w.r.t. a variable x, then it does not make sense to keep applying the resolution rule with x as the resolving variable, either because the supports of x do not have an empty intersection (therefore the rule could not be applied) or the first clause of the conclusion is a tautology.

Definition 16. A multiset of clauses $\mathcal{C}$ is said to be saturated with respect to x if for every subset of clauses $\{S_1 : x \vee D_1, \ldots, S_m : x \vee D_m\} \subseteq \mathcal{C}$, it holds that either $S_1 \cap \cdots \cap S_m \neq \emptyset$ or there exist literals $S_{i_1} : y \in D_{i_1}, \ldots, S_{i_l} : y \in D_{i_l}, l \leqslant m$, such that $S_{i_1} \cup \cdots \cup S_{i_l} = N$.

A multiset of clauses $\mathcal{C}'$ is a *saturation of* $\mathcal{C}$ w.r.t. x if $\mathcal{C}'$ is saturated w.r.t. x and $\mathcal{C} \vdash_x \mathcal{C}'$, i.e. $\mathcal{C}'$ can be obtained from $\mathcal{C}$ applying the inference rule resolving x finitely many times.

Lemma 17. *Let $\mathcal{E}$ be a saturated multiset of clauses w.r.t. x. Let $\mathcal{E}'$ be the subset of clauses of $\mathcal{E}$ not containing x. Then, any assignment I satisfying $\mathcal{E}'$ (and not assigning x) can be extended to an assignment satisfying $\mathcal{E}$.*

Proof. We have to extend I to satisfy the whole $\mathcal{E}$. In fact we only need to set the value of x. Let us partition the multiset $(\mathcal{E} - \mathcal{E}')$ (multiset of clauses that contain the variable x) into two multisets: $(\mathcal{E} - \mathcal{E}')_T$ the multiset already satisfied by I, and $(\mathcal{E} - \mathcal{E}')_F$ the multiset such that the partial assignment I does not satisfy any of the clauses.

Let $(\mathcal{E} - \mathcal{E}')_F = \{S_1 : x \vee D_1, \ldots, S_k : x \vee D_k\}$. Since $\mathcal{E}$ is saturated, either $S_1 \cap \cdots \cap S_k \neq \emptyset$ or $D_1 \cup \cdots \cup D_k$ is a tautology. If $S_1 \cap \cdots \cap S_k \neq \emptyset$, we extend I with a value inside the intersection. If $S_1 \cap \cdots \cap S_k = \emptyset$, then $D_1 \cup \cdots \cup D_k$ is a tautology. Then, there exist $S_{i_1} : y \in D_{i_1}, \ldots, S_{i_l} : y \in D_{i_l}, l \leqslant m$, such that $S_{i_1} \cup \cdots \cup S_{i_l} = N$. Then, I satisfies one of these literals and the corresponding clause. So, some of the clauses are not in $(\mathcal{E} - \mathcal{E}')_F$. Contradiction. $\square$

Another ingredient of the proof is showing that the procedure of applying inference until saturation terminates. For that we need to define the notion of characteristic function of a multiset of clauses.

We assign a function $P : \{0,1\} \to \{0,1\}$ to every signed literal, a function $P : \{0,1\}^n \to \{0,1\}$ to every clause, and a function $P : \{0,1\}^n \to \mathbb{N}$ to every multiset of clauses as follows.

Definition 18. The characteristic function of a signed literal $L = \{i_1, \ldots, i_m\} : x$ is $P_L(x) = (1 - \{i_1\} : x) \cdots (1 - \{i_m\} : x)$.

Definition 19. For every clause $C = L_1 \vee \ldots \vee L_s$ we define its characteristic function as $P_C(\vec{x}) = P_{L_1}(x_1) \cdots P_{L_s}(x_s)$.

For every multiset of clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$, we define its *characteristic function* as $P_{\mathcal{C}} = \Sigma_{i=1}^{m} P_{C_i}(\vec{x})$.

Notice that for every assignment $\mathcal{I}, P_{\mathcal{C}}(\mathcal{I})$ is the number of clauses of $\mathcal{C}$ falsified by $\mathcal{I}$.

Also notice that the set of functions $\{0,1\}^n \to \mathbb{N}$, with the order relation: $f \leqslant g$ if for all $\vec{x}, f(\vec{x}) \leqslant g(\vec{x})$, defines a partial order between functions. The strict part of this relation, i.e. $f < g$ if for all $\vec{x}, f(\vec{x}) \leqslant g(\vec{x})$ and for some $\vec{x}, f(\vec{x}) < g(\vec{x})$, defines a well-founded order.

Lemma 20. *For every multiset of clauses $\mathcal{C}$ and every variable x, there exists a multiset $\mathcal{C}'$ such that $\mathcal{C}'$ is a saturation of $\mathcal{C}$ w.r.t. x.*

Proof. By the soundness of the signed MaxSAT parallel resolution rule, every application of the rule replaces a multiset of clauses by another with the same characteristic function. But if we only look at the multisets containing the variable x, the characteristic function strictly decreases. This is because the first clause of the conclusion of the application of the rule does not contain the variable x and since it is not a tautology, its characteristic function is strictly greater than zero. Therefore when we apply the rule the characteristic function of the multiset of conclusions eliminating the first clause is strictly smaller than the characteristic function of the premises. $\square$

Now we are ready to state and prove the proof of completeness.

Theorem 21. *Signed MaxSAT parallel resolution is complete; i.e., for any multiset of clauses $\mathcal{C}$, we have*

$$\mathcal{C} \vdash \underbrace{\Box, \ldots, \Box}_{m}, \mathcal{D}$$

where $\mathcal{D}$ is a satisfiable multiset of clauses, and m is the minimum number of unsatisfied clauses of $\mathcal{C}$.

Proof. This part of the completeness proof is identical to the corresponding proof for the MaxSAT rule [7,8]. Let $x_1, \ldots, x_n$ be any list of the variables of $\mathcal{C}$. We construct two sequences of multisets $C_0, \ldots, C_n$ and $D_1, \ldots, D_n$ such that

1. $\mathcal{C} = \mathcal{C}_0$,
2. for $i = 1, \ldots, n, \mathcal{C}_i \cup \mathcal{D}_i$ is a saturation of $\mathcal{C}_{i-1}$ w.r.t. x_i, and
3. for $i = 1, \ldots, n, \mathcal{C}_i$ is a multiset of clauses not containing $x_1, \ldots, x_i$, and $\mathcal{D}_i$ is a multiset of clauses containing the variable x_i.

By Lemma 20, this sequences can effectively be computed: for $i = 1, \ldots, n$, we saturate $\mathcal{C}_{i-1}$ w.r.t. x_i, and then we partition the resulting multiset into a subset $\mathcal{D}_i$ containing x_i, and another $\mathcal{C}_i$ not containing this variable.

Notice that, since $\mathcal{C}_n$ does not contain any variable, it is either the empty multiset $\emptyset$, or it only contains (some) empty clauses $\{\Box, \ldots, \Box\}$.

Now we are going to prove that the multiset $\mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$ is satisfiable by constructing an assignment satisfying it. For $i = 1, \ldots, n$, let $\mathcal{E}_i = \mathcal{D}_i \cup \ldots \cup \mathcal{D}_n$, and let $\mathcal{E}_{n+1} = \emptyset$. Notice that, for $i = 1, \ldots, n$,

1. the multiset $\mathcal{E}_i$ only contains the variables $\{x_i, \ldots, x_n\}$,
2. $\mathcal{E}_i$ is saturated w.r.t. x_i, and
3. $\mathcal{E}_i$ decomposes as $\mathcal{E}_i = \mathcal{D}_i \cup \mathcal{E}_{i+1}$, where all the clauses of $\mathcal{D}_i$ contain x_i and none of $\mathcal{E}_{i+1}$ contains x_i.

Now, we construct a sequence of assignments $I_1, \ldots, I_{n+1}$, where I_{n+1} is the empty assignment, hence satisfies $\mathcal{E}_{n+1} = \emptyset$. Now, I_i is constructed from I_{i+1} as follows. Assume by induction hypothesis that I_{i+1} satisfies $\mathcal{E}_{i+1}$. Since $\mathcal{E}_i$ is saturated w.r.t. x_i, and decomposes into $\mathcal{D}_i$ and $\mathcal{E}_{i+1}$, by Lemma 17, we can extend I_{i+1} with an assignment for x_i to obtain I_i satisfy $\mathcal{E}_i$. Iterating, we get that I_1 satisfies $\mathcal{E}_1 = \mathcal{D} = \bigcup_{i=1}^{n} \mathcal{D}_i$.

Concluding, since by the soundness of the rule (Theorem 15) the inference preserves the number of falsified clauses for every assignment, $m = |\mathcal{C}_n|$ is the minimum number of unsatisfied clauses of $\mathcal{C}$. $\Box$

In fact, our result is stronger than the statement of the previous theorem, because it characterizes a family of complete rules:

Corollary 22. *Any sound resolution rule for signed MaxSAT, that is applicable when it is not saturated under some variable and that contains a non-tautological resolvent in which the resolving variable does not appear is complete.*

The signed MaxSAT binary resolution rule is also complete. This fact does not follow from the previous corollary since the rule does not have a resolvent in which the resolving variable does not appear. The completeness follows from the fact that the signed parallel MaxSAT resolution rule is a derived rule of the signed MaxSAT binary resolution rule.

Theorem 23. *The signed MaxSAT binary resolution rule is complete.*

5. Linking signed logic and weighted constraint programming

The Weighted Constraint Satisfaction Problem (WCSP) is a well known soft constraint framework for modeling over-constrained problems. WCSP is an optimization version of the CSP framework in which constraints are extended by associating *costs* to tuples. Solving a WCSP instance, which is NP-hard, consists in finding a complete assignment of optimal cost.

There exist two main types of complete algorithms to solve constraint problems by applying inference rules (or transformations). We can use algorithms based on applying complete inference rules, or algorithms based on search that apply incomplete inference rules. Informally, complete inference rules are those such that applying them iteratively we can solve the problem, i.e., to find an optimal solution. A typical example are WCSP algorithms such as Bucket Elimination [11] that solve WCSP instances without search. They obtain an optimal solution by applying transformations that preserve cost distributions.

On the other hand, when using just incomplete inference rules one cannot guarantee to solve the problem completely (or find the optimal solution). This is why these rules need to be incorporated into other algorithms that guarantee completeness. These rules are usually added to WCSP Branch and Bound (BnB) based search algorithms such as MAC* [25], MFDAC* [24] and MEDAC* [10]. Branch and Bound search algorithms perform a systematic search in the space of all possible assignments. They differ in the method of computing a lower bound at each node of the proof tree to prune some parts of the search space. Incomplete inference rules, like local consistency rules, can be used to compute these lower bounds. Modern algorithms such as MAC*, MFDAC* and MEDAC* enforce some extension of Arc Consistency (AC) to WCSP—Soft AC (AC*), Full Directional AC (FDAC*) or Existential Directional AC (EDAC*)—when computing that lower bound.

Given the relevance of both complete and incomplete inference in WCSP solvers, our aim in this section is to establish a link between the above complete inference systems for Signed MaxSAT and complete inference algorithms for WCSP, and to define refinements of the Signed-MaxSAT inference rules that capture the known *soft local consistency* properties defined in the literature and other stronger forms of inference. Therefore, we propose a unified approach for solving WCSP using inference techniques in the case that costs are integers.

The remaininig of the present section is organized as follows. Section 5.1 defines the weighted constraint satisfaction problem (WCSP) and its encoding as an instance of the signed maximum satisfiability problem. Section 5.2 describes a new complete algorithm for solving WCSP, which is the derived from the completeness proof of the new signed inference systems. Section 5.3 defines a restriction and a generalization of the signed MaxSAT parallel rule called, respectively, signed MaxSAT *i*-consistency resolution and signed MaxSAT (*i*,*j*)-consistency resolution. These rules have the following property: if a WCSP signed encoding is *closed* under signed MaxSAT *i*-consistency, then the corresponding WCSP is *i*-consistent, and if it is *closed* under signed MaxSAT (*i*,*j*)-consistency, then the WCSP is (*i*,*j*)-consistent. Section 5.4 describes an algorithm that applies directional soft consistency with the previous rules, and enforces directional *i*-consistency.

5.1. Weighted constraint satisfaction problems

Next we will will define constraint satisfaction and weighted constraint satisfaction problems, and explain why the language of signed CNF formulas is appropriate to talk about constraint and weighted constraint satisfaction problems.

Definition 24. A *constraint satisfaction problem (CSP)* instance is defined as a triple $\langle X, D, C\rangle$, where $X = \langle x_1, \ldots, x_n\rangle$ is a set of variables, $D = \langle d(x_1), \ldots, d(x_n)\rangle$ is a set of domains containing the values the variables may take, and $C = \{C_1, \ldots, C_m\}$ is a set of constraints. Each constraint $C_i = \langle S_i, R_i\rangle$ is defined as a relation R_i over a subset of variables $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$, called the *constraint scope*. The relation R_i may be represented extensionally as a subset of the Cartesian product $d(x_{i_1}) \times \cdots \times d(x_{i_k})$.

Definition 25. An assignment v for a CSP instance $\langle X, D, C\rangle$ is a mapping that assigns to every variable $x_i \in X$ an element $v(x_i) \in d(x_i)$.

A partial assignment v for a CSP instance $\langle X, D, C\rangle$ is a mapping that assigns to every variable $x_i \in Y$ an element $v(x_i) \in d(x_i)$, where Y is a subset of X.

An assignment or partial assignment v satisfies a constraint $\langle\{x_{i_1}, \ldots, x_{i_k}\}, R_i\rangle \in C$ iff $\langle v(x_{i_1}), \ldots, v(x_{i_k})\rangle \in R_i$.

Definition 26. A Weighted CSP (WCSP) instance is defined as a triple $\langle X, D, C\rangle$, where X and D are variables and domains as in CSP. A constraint C_i is now defined as a pair $\langle S_i, f_i\rangle$, where $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$ is the constraint scope and $f_i : d(x_{i_1}) \times \cdots \times d(x_{i_k}) \to \mathbb{N}$ is a *cost function*. The cost of a constraint C_i induced by an assignment v in which the variables of $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$ take values $b_{i_1}, \ldots, b_{i_k}$ is $f_i(b_{i_1}, \ldots, b_{i_k})$.

An optimal solution to a WCSP instance is a complete assignment in which the sum of the costs of the constraints is minimal.

Definition 27. The Weighted Constraint Satisfaction Problem (WCSP) for a WCSP instance consists in finding an optimal solution for that instance.

Now we present an encoding that shows that solving WCSPs is equivalent to finding optimal assignments to signed CNF formulas.

Definition 28. The signed encoding of a WCSP instance $\langle X, D, C\rangle$ is the signed CNF formula that contains, for every variable $x_i \in X$ with domain $d(x_i)$, a propositional variable x_i with the same domain $d(x_i)$, and for every possible tuple $\langle b_{i_1}, \ldots, b_{i_k}\rangle \in d(x_{i_1}) \times \ldots \times d(x_{i_k})$ of every constraint $\langle\{x_{i_1}, \ldots, x_{i_k}\}, f_i\rangle \in C, f_i(b_{i_1}, \ldots, b_{i_k})$ copies of the signed clause:

$$\overline{\{b_{i_1}\}} : x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}} : x_{i_k}$$

Alternatively, we can consider just one weighted signed clause

$$\overline{\{b_{i_1}\}} : x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}} : x_{i_k}, \quad f_i(b_{i_1}, \ldots, b_{i_k})$$

For the sake of clarity we will use unweighted clauses. The extension of our theoretical results to weighted clauses is straightforward. In Section 3 we give the sound and complete inference rule for signed MaxSAT using weights.

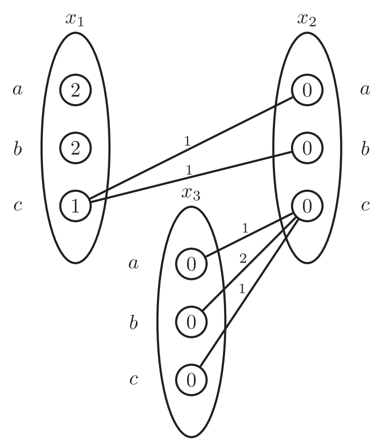
Next we provide an example of the encoding of a WCSP as a signed formula.

Example 29. Fig. 1 shows a WCSP instance $\langle X, D, C\rangle$ and its signed encoding. The WCSP has the set of variables $X = \{x_1, x_2, x_3\}$ with domains $d(x_1) = d(x_2) = d(x_3) = \{a, b, c\}$. There is a binary constraint between variables x_1 and x_2, a binary constraint between variables x_2 and x_3, and a unary constraint for every variable. Unary costs are depicted inside small circles. Binary costs are depicted as labeled edges connecting the corresponding pair of values. The label of each edge is the corresponding cost. If two values are not connected, the binary cost between them is 0. In this instance, the optimal cost is 2.

Notice that the graphical representation of constraint satisfaction problems is limited to unary and binary constraints. Bigger constraints would require a representation in terms of hypergraphs which would be very difficult to visualize. Instead the language of signed clauses does not have this restriction.

Proposition 30. *Solving a WCSP instance P is equivalent to solving the signed MaxSAT problem of its signed encoding; i.e., the optimal cost of P coincides with the minimal number of unsatisfied signed clauses of the signed encoding of P.*

Proof. For every combination of values to the variables of the scope of a constraint $C_i = \langle S_i, f_i\rangle$, the signed encoding contains as many clauses as the cost associated with that combination. If an assignment of the signed encoding restricted to the variables of S_i coincides with a combination of C_i with cost 0, then all the clauses of the signed encoding introduced by C_i are satisfied because there is no clause forbidding that combination. If an assignment of the signed encoding restricted to the variables of S_i coincides with a combination $\langle b_{i_1}, \ldots, b_{i_k}\rangle$ of C_i with cost u, where $u > 0$, then, by construction of the signed encoding, only the u clauses of the form $\overline{\{b_{i_1}\}} : x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}} : x_{i_k}$ are falsified among the clauses introduced by C_i. □



$$
\begin{array}{rl}
1: & \overline{\{a\}}{:}x_1 \\
2: & \overline{\{a\}}{:}x_1 \\
3: & \overline{\{b\}}{:}x_1 \\
4: & \overline{\{b\}}{:}x_1 \\
5: & \overline{\{c\}}{:}x_1 \\
6: & \overline{\{c\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \\
7: & \overline{\{c\}}{:}x_1 \vee \overline{\{b\}}{:}x_2 \\
8: & \overline{\{a\}}{:}x_3 \vee \overline{\{c\}}{:}x_2 \\
9: & \overline{\{b\}}{:}x_3 \vee \overline{\{c\}}{:}x_2 \\
10: & \overline{\{b\}}{:}x_3 \vee \overline{\{c\}}{:}x_2 \\
11: & \overline{\{c\}}{:}x_3 \vee \overline{\{c\}}{:}x_2
\end{array}
$$

Fig. 1. A WCSP instance and its signed encoding (see Example 29).

In our work we use signed formulas instead of Boolean formulas for the following reasons: (a) we can define resolution-like inference rules for constraint satisfaction (weighted or not) problems extending the corresponding rules for Boolean logic; and (b) the language of signed logic corresponds directly to the language of constraints and therefore complicated encodings are not necessary.

5.2. A complete algorithm for WCSP

As a first result of the link between signed logic and WCSP, we describe an exact algorithm for solving WCSP, called Signed MaxSAT DP, which is derived from the proof of Theorem 21.

Signed MaxSAT DP algorithm:

```
input: A WCSP instance $P$
$\mathcal{C}_0 := signed\_encoding(P)$
for $i := 1$ to $k$
  $\mathcal{C} := saturation(\mathcal{C}_{i-1}, x_i)$
  $\langle \mathcal{C}_i, \mathcal{D}_i \rangle := partition(\mathcal{C}, x_i)$
endfor
$m := |\mathcal{C}_k|$
$I := \emptyset$
for $i := k$ downto 1
  $I := I \cup [x_i \mapsto extension(x_i, I, \mathcal{D}_i)]$
output: $m, I$
```

Given an initial WCSP instance P with k variables, this algorithm returns the optimal cost m of P and an optimal solution I.

The function $saturation(\mathcal{C}_{i-1}, x_i)$ computes a saturation of $\mathcal{C}_{i-1}$ w.r.t. x_i applying the resolution rule resolving x until it gets a saturated set. Lemma 17 ensures that this process terminates, in particular that it does not cycle. As we have already said, the saturation of a multiset is not unique, but the proof of Theorem 21 does not depend on which particular saturation we take.

The function $partition(\mathcal{C}, x_i)$ computes a partition of $\mathcal{C}$, already saturated, into the subset of clauses containing x_i and the subset of clauses not containing x_i.

The function $extension(x_i, I, \mathcal{D}_i)$ computes an assignment for x_i extending the assignment I, to satisfy the clauses of $\mathcal{D}_i$ according to Lemma 17. The function filters all clauses of $\mathcal{D}_i$ that are not satisfied by I. Then it computes the intersection of the supports for x_i of all of them, and returns one of the values of such a intersection, i.e. returns a value from

$$\cap\{S \mid S : x_i \vee A \ \in \ \mathcal{D}_i \text{ and } I \text{ falsifies } A\}$$

The argumentation of the proof of Lemma 17 ensures that this intersection is not empty.

Algorithm Signed MaxSAT DP is the equivalent version of the Davis–Putnam (DP) algorithm [9] in the context of the Signed MaxSAT problem. The algorithm also follows the general bucket elimination schema [11].

In the basic Davis–Putnam algorithm, variables are eliminated one by one following some ordering. To eliminate a variable, say x, we perform all possible resolution steps with clauses containing x, obtaining new clauses without the variable x. After each variable elimination step, we can show that the new set of clauses is satisfiable iff the previous set was satisfiable. The new algorithm, Signed MaxSAT DP, is a little different. Some of the clauses we obtain saturating a variable, say x, still contain that variable. Therefore when the resolution process finishes because we have a saturated multiset of clauses, some clauses containing variable x are still in the multiset. Lemma 17 ensures that this clauses are not necessary to continue the resolution process. They will only be important to obtain an assignment that satisfies the maximum number of clauses.

In the general bucket elimination schema [11,12], constraints are separated into *buckets*, one for each variable, according to some ordering on the variables. Every constraint is put into the bucket corresponding to the biggest variable of its domain. Then, we simplify buckets sequentially, following the order of the variables (starting with the biggest one and finishing with the smallest one). All constraints of a bucket are combined using some sort of *join operator*, $R_A \bowtie R_B$ and then, the variable X defining the bucket is removed using a *projection operator* Π_X. In the context of CSP, the domain of $R_A \bowtie R_B$ is the union $A \cup B$ of the domains of R_A and R_B, and contains all tuples satisfying both R_A and R_B. Therefore, if constraints are represented extensively as sets of goods or no-goods (the characteristic functions $2^{|A|} \to \{0,1\}$ and $2^{|B|} \to \{0,1\}$), the representation of $R_A \bowtie R_B$ (the function $2^{|A\cup B|} \to \{0,1\}$) can have size equal to the product of the sizes of the representations of R_A and R_B. Moreover, this is not only in the worst case, but also in the typical case. The DP algorithm also follows a bucket elimination schema. The size representation explosion is avoided (in the typical case) using clauses, that can be seen as compact or intensive representations of characteristic functions or sets of goods. Notice that resolution of two clauses results into a new clause with size bounded by the sum of both sizes. In this sense, Signed MaxSAT DP can be seen as the natural extension of DP to (W) CSP where constraints are compactly represented as (weighted) signed clauses.

Bucket elimination based algorithms are not practical unless the interaction graph of variables (see Definition 31) is sparse. In [22], it is described an algorithm that combines branch and bound with bucket elimination. Some variables are eliminated by bucket elimination if the resulting new constrains have low arity, and the others are eliminated using branching. This algorithm outperforms branch and bound and bucket elimination on some problems. The algorithm of [22] can be improved using signed clauses to compactly represent constraints.

Like in the DP and bucket elimination algorithms, in our algorithm the order of the saturation of the variables can be freely chosen, i.e. the sequence $x_1, \ldots x_n$ can be any enumeration of the variables. Nevertheless, different orders may have different performance. Next we present some definitions that will help us analyze the time and space used in our algorithm. For the definitions we will follow closely the presentation of [12,13], extending the definitions to the signed formula context.

Definition 31. The iteration graph of a signed CNF formula $\mathcal{C}$, denoted $G(\mathcal{C})$, is an undirected graph that contains a node for every variable in $\mathcal{C}$ and an edge for every pair of variables appearing in the same clause.

In [13] we find the following running example. $\phi = \overline{C}, A \vee B \vee C, \overline{A} \vee B \vee E, \overline{B} \vee C \vee D\}$. We can think of ϕ as a signed formula where the domain N is binary. The set of edges of $G(\phi)$ is $\{(A,B),(A,C),(B,C),(A,E),(B,E),(B,D),(C,D)\}$. Note that the edges pair up variables, not literals. Therefore, we do not include the edge $(\overline{A},B)$.

Definition 32. Let $\mathcal{C}$ be a signed CNF formula, $G(\mathcal{C})$ its iteration graph, and $\mathcal{O}$ an ordering of de nodes of $G(\mathcal{C})$ (i.e. an ordering of the variables of $\mathcal{C}$). The *width* of a variable x is the number of variables connected to x in $G(\mathcal{C})$ that precede x in the order $\mathcal{O}$. The width of a graph along $\mathcal{O}$ is the maximum width over all the variables. The induced graph of $G(\mathcal{C})$ along $\mathcal{O}$, denoted $G^*_{\mathcal{O}}(\mathcal{C})$, is obtained by connecting all neighbors of any variable x that precede x in the ordering. The induced widthof $G(\mathcal{C})$ along $\mathcal{O}$, denoted $w^*_{\mathcal{O}}$, is the maximal width of a variable in the induced graph $G^*_{\mathcal{O}}(\mathcal{C})$. The induced width w^* of $G(\mathcal{C})$ is the minimum induced width along any ordering.

Different orderings of variables can generate different induced widths of the induced graph. Going back to the running example of [13] we can analyze the induced width depending on different orderings:

1. $\mathcal{O}_1 = (E,D,C,A,B)$. The induced graph adds the following edges: $\{(A,D),(E,C),(E,D)\}$. The width of B in the induced graph is 4.
2. $\mathcal{O}_2 = (A,B,C,D,E)$. The induced graph along $\mathcal{O}_2$ does not introduce any new edges. The induced width is 2.

The notion of induced width is related to the maximal size (number of literals) of clauses obtained by performing resolution inference in a Davis–Putnam algorithm. In fact the induced width along an ordering $\mathcal{O}$ of variables, coincides with the maximal number of literals a clause can have if we perform a Davis–Putnam algorithm on the initial clauses using the reverse ordering of $\mathcal{O}$. For instance in the ordering $\mathcal{O}_1$, after resolving with the variables E,D,C, we resolve $\overline{A} \vee B \vee E$ with $\overline{B} \vee C \vee D$, obtaining among others the clause $\overline{A} \vee E \vee C \vee D$. This clause has 4 literals as the induced width of the variable B. Instead, if we perform a Davis–Putnam type of algorithm using $\mathcal{O}_2$, we only obtain as new clause $A \vee B$. Since our algorithm for solving signed MaxSAT or WCSP problems is based in the same type of algorithms, the previous definitions will help us analyze the performance of the algorithm.

Theorem 33. *Let $\mathcal{C}$ be an arbitrary signed CNF instance with n variables and sign of size N. Let $\mathcal{O}$ be an ordering of the variables of $\mathcal{C}$ and let the Signed MaxSAT DP algorithm eliminate the variables in the reverse ordering of $\mathcal{O}$. The time and space complexity of solving the signed MaxSAT problem for $\mathcal{C}$ using Signed MaxSAT DP is $O(n2^{2Nw^*_{\mathcal{O}}})$ and $O(n2^{Nw^*_{\mathcal{O}}})$ respectively, where $w^*_{\mathcal{O}}$ is the induced width of $G(\mathcal{C})$ along $\mathcal{O}$.*

Proof. Lemmas 17 and 20 and Theorem 21 ensure that the algorithm solves the signed MaxSAT problem. Now we will show the time and space bounds. In each variable elimination step of the algorithm, we can generate a maximum of $(2^N)^{w^*_{\mathcal{O}}}$ signed clauses. This is because, for every variable we eliminate, the induced width of that variable is at most $w^*_{\mathcal{O}}$. So performing resolution on that variable we can generate in the worst case all clauses with $w^*_{\mathcal{O}}$ variables. Since each variable has 2^N possible literals, we obtain the corresponding space bound. We obtain the time bound by noting that the maximal number of steps in each variable elimination is $2^{N\,w^{*2}_{\mathcal{O}}}$. This is because in the worst case we can try to apply resolution among each pair of clauses that contain the variable. This will give us a quadratic bound. Finally, we multiply the time and space bounds by n, since n is the number of variables and therefore the number of applications of the saturation procedure. □

5.3. Local consistency via resolution

In WCSP a number of local consistency properties have been proposed. These local properties do not ensure the global consistency of a set of constraints. However, they can be enforced very efficiently and used to find a lower bound of the cost.

We are going to prove that the signed MaxSAT parallel resolution rule actually enforces i-consistency in WCSP. In order to do that, we first define *(i,j)-consistency* and *i-consistency* in WCSP. We will define in parallel the notions of partial consistency for CSPs and WCSPs. Both will be useful later on.

Definition 34. Given a CSP (a WCSP) and a partial assignment v, v is *consistent* if it satisfies all of the constraints of the CSP (or WCSP) whose scopes have all their variables assigned by v.

Definition 35. A CSP is (i,j)-consistent, for $i \geqslant 0$ and $j \geqslant 1$, iff any consistent instantiation of i variables can be extended to a consistent instantiation of any j additional variables.

A WCSP is (i,j)-consistent, for $i \geqslant 0$ and $j \geqslant 1$, iff any instantiation of i variables V that satisfies all constraints C_D with scope D, where $\emptyset \neq D \subseteq V$, can be extended to an instantiation of j additional variables V' that satisfies all constraints $C_{D'}$ with scope D', where $\emptyset \neq D' \subseteq V \cup V'$.

Definition 36. A CSP (WCSP) is i-consistent for, $i \geqslant 1$, iff it is $(i-1,1)$-consistent.

A CSP (WCSP) is strong i-consistent, for $i \geqslant 1$, iff it is k-consistent, for every $k, 1 \leqslant k \leqslant i$.

Now we will restrict in the signed MaxSAT parallel resolution rule the number of variables that appear in $D_1 \cup \cdots \cup D_k$.

Definition 37. The signed MaxSAT $(i,1)$-consistency resolution rule is the signed MaxSAT parallel resolution where exactly i variables appear in $D_1 \cup \cdots \cup D_k$ i.e., $|var(D_1 \cup \cdots \cup D_k)| = i$. If $|var(D_1 \cup \cdots \cup D_k)| \leqslant i$ we call the resolution rule the strong signed MaxSAT $(i,1)$-consistency resolution rule.

In the previous definitions and in the following lemma we use the words satisfies/falsifies a constraint in the context of WCSP. This is strictly speaking incorrect, but makes the definitions easier to understand. What we mean is: given constraint $C_i = \langle S_i, f_i \rangle$, where $S_i = \{x_{i_1}, \ldots, x_{i_k}\}$, and given assignment $\{a_{i_1}, \ldots, a_{i_k}\}, \{a_{i_1}, \ldots, a_{i_k}\}$ satisfies C_i if $f_i(a_{i_1}, \ldots, a_{i_k}) = 0$. $\{a_{i_1}, \ldots, a_{i_k}\}$ falsifies C_i if $f_i(a_{i_1}, \ldots, a_{i_k}) > 0$.

Lemma 38. *If a set of clauses is closed under the signed MaxSAT $(i-1,1)$-consistency resolution rule, then its corresponding WCSP is i-consistent.*

Proof. Suppose that a set of clauses is closed by the signed MaxSAT $(i-1,1)$-consistency resolution rule, but its corresponding constraint network is not i-consistent. We have some tuple of $i-1$ variables $x_1, \ldots, x_{i-1}$ and $i-1$ consistent values $a_1, \ldots, a_{i-1}$ of their domains, and there exists also a variable x such that $a_1, \ldots, a_{i-1}$ cannot be extended to x consistently. I.e. for any value b of the domain of x, the tuple of i values $a_1, \ldots, a_{i-1}, b$ for the variables $x_1, \ldots, x_{i-1}, x$ falsifies some constraint about a subset of such variables (where at least the variable x is present). Therefore, for any b, the tuple $a_1, \ldots, a_{i-1}, b$ is a nogood, and therefore for any b there is a clause whose literals are a subset of $S_1 : x_1 \vee \ldots \vee S_{i-1} : x_{i-1} \vee S : x$ where $\forall l\ 1 \leqslant l \leqslant i-1,\ a_l \notin S_l$ and $b \notin S$. Since the set of clauses is closed under the rule, and the intersection of the supports of x is empty, our set of clauses also contains a subclause of $S'_1 : x_1 \vee \ldots \vee S'_{i-1} : x_{i-1}$ where $\forall l\ 1 \leqslant l \leqslant i-1,\ a_l \notin S'_l$. So the tuple $a_1, \ldots, a_{i-1}$ is a no good for $\langle x_1, \ldots, x_{i-1} \rangle$ and this contradicts the assumption. □

5.3.1. Signed MaxSAT (i,j)-consistency resolution rule

Once we have introduced the signed MaxSAT $(i,1)$-consistency resolution rule, we are ready to introduce a more general rule, the signed MaxSAT (i,j)-consistency resolution rule. In this case, instead of just having one resolving variable we have exactly j resolving variables. For the sake of clarity one way of describing this rule is to *collapse* the j resolving variables, say $x_1, \ldots, x_j$, into one single variable x' whose domain is equal to the Cartesian product of the domains of the j original variables, $d(x') = d(x_1) \times \cdots \times d(x_j)$, where $d(x)$ is the domain of x. As a consequence, a disjunction of signed literals on the j variables,

$S_1 : x_1 \vee \cdots \vee S_j : x_j$ is replaced by the signed literal, $S' : x'$, where $S' = \overline{\overline{S_1} \times \cdots \times \overline{S_j}}$. Then, we can simply apply the (i,1)-signed MaxSAT resolution rule taking x' as the new resolving variable.

Definition 39. The **signed MaxSAT (*i*,*j*)-consistency resolution** rule is defined as follows:

$$\begin{array}{l}
S_{1,1}{:}x_1 \vee \cdots \vee S_{1,j}{:}x_j \vee D_1 \\
\qquad\qquad\vdots \\
S_{k,1}{:}x_1 \vee \cdots \vee S_{k,j}{:}x_j \vee D_k \\
\hline\hline
D_1 \vee \ldots \vee D_k \\
\left\{\begin{array}{r}
(S'_1 \cap \ldots \cap S'_{t-1}) \cup S'_t{:}x' \vee D_1 \vee \ldots \vee D_t \qquad\qquad \\
(S'_1 \cap \ldots \cap S'_{t-1}){:}x' \vee \underline{D_1 \vee \ldots \vee D_{t-1}} \vee \overline{D_t} \\
S'_t{:}x' \vee \overline{D_1 \vee \ldots \vee D_{t-1}} \vee D_t
\end{array}\right\}_{t=2\ldots k}
\end{array}$$

where $d(x') = d(x_1) \times \cdots \times d(x_j), S'_r = \overline{\overline{S_{r,1}} \times \cdots \times \overline{S_{r,j}}}, k \geqslant r \geqslant 1, |var(D_1 \cup \cdots \cup D_k)| = i$, and $\bigcap_{i=1}^{k} S'_i = \emptyset$.

Regarding the relation with the local consistencies in WCSP, we also say that if a set of clauses is closed under the signed MaxSAT (*i*,*j*)-consistency resolution rule, then its corresponding WCSP is (*i*,*j*)-consistent.

Finally, notice that we get different instantiations of this rule by fixing the *i* and *j* parameters. In particular, if we set the *i* parameter to 0 we get an interesting instantiation where the rule has the empty clause as the first of its conclusions.

An alternative definition of the signed MaxSAT (*i*,*j*)-consistency resolution rule which does not introduce new variables is the following:

Definition 40. The **signed MaxSAT (*i*,*j*)-consistency resolution** rule is defined as follows (alternative version):

$$\begin{array}{l}
S_{1,1}{:}x_1 \vee \cdots \vee S_{1,j}{:}x_j \vee D_1 \\
\qquad\qquad\vdots \\
S_{k,1}{:}x_1 \vee \cdots \vee S_{k,j}{:}x_j \vee D_k \\
\hline\hline
D_1 \vee \ldots \vee D_k \\
\left\{\begin{array}{r}
\forall\ (a_1, \ldots, a_t) \notin (S'_1 \cap \ldots \cap S'_{t-1}) \cup S'_t : \qquad\qquad\qquad\qquad \\
\overline{\{a_1\}}{:}x_1 \vee \cdots \vee \overline{\{a_t\}}{:}x_t \vee D_1 \vee \ldots \vee D_t \qquad \\
\forall\ (a_1, \ldots, a_{t-1}) \notin (S'_1 \cap \ldots \cap S'_{t-1}) : \qquad\qquad\qquad\qquad \\
\overline{\{a_1\}}{:}x_1 \vee \cdots \vee \overline{\{a_{t-1}\}}{:}x_t \vee D_1 \vee \ldots \vee D_{t-1} \vee \overline{D_t} \\
\forall\ (a_1, \ldots, a_t) \notin S'_t : \qquad\qquad\qquad\qquad \\
\overline{\{a_1\}}{:}x_1 \vee \cdots \vee \overline{\{a_t\}}{:}x_t \vee \overline{D_1 \vee \ldots \vee D_{t-1}} \vee D_t
\end{array}\right\}_{t=2\ldots k}
\end{array}$$

where $S'_r = \overline{\overline{S_{r,1}} \times \cdots \times \overline{S_{r,j}}}, k \geqslant r \geqslant 1, |var(D_1 \cup \cdots \cup D_k)| = i$ and $\bigcap_{i=1}^{k} S'_i = \emptyset$.

Example 41. For three premises, the signed MaxSAT (*i*,*j*)-consistency resolution rule has the following form:

$$\begin{array}{l}
S_{1,1}{:}x_1 \vee \cdots \vee S_{1,j}{:}x_j \vee D_1 \\
S_{2,1}{:}x_1 \vee \cdots \vee S_{2,j}{:}x_j \vee D_2 \\
S_{3,1}{:}x_1 \vee \cdots \vee S_{3,j}{:}x_j \vee D_3 \\
\hline\hline
D_1 \vee D_2 \vee D_3 \\
\begin{array}{rl}
S'_1 \cup S'_2{:}x' & \vee D_1 \vee \underline{D_2} \\
S'_1{:}x' & \vee \underline{D_1} \vee \overline{D_2} \\
S'_2{:}x' & \vee \overline{D_2} \vee D_1 \\
(S'_1 \cap S'_2) \cup S'_3{:}x' & \vee D_1 \vee D_2 \vee \underline{D_3} \\
(S'_1 \cap S'_2){:}x' & \vee \underline{D_1 \vee D_2} \vee \overline{D_3} \\
S'_3{:}x' & \vee \overline{D_1 \vee D_2} \vee D_3
\end{array}
\end{array}$$

where $S'_1 = \overline{\overline{S_{1,1}} \times \cdots \times \overline{S_{1,j}}},\ S'_2 = \overline{\overline{S_{2,1}} \times \cdots \times \overline{S_{2,j}}},\ S'_3 = \overline{\overline{S_{3,1}} \times \cdots \times \overline{S_{3,j}}},\ d(x') = d(x_1) \times \cdots \times d(x_j)$ and $S'_1 \cap S'_2 \cap S'_3 = \emptyset$.

Example 42. Given the set of variables $X = \{x_1, x_2, x_3\}$ with domains $d(x_1) = d(x_2) = d(x_3) = \{a, b\}$, the following are examples of the application of signed MaxSAT $(i, 2)$-consistency resolution rule (with x_1, x_2 as the resolving variables):

Consider $S_{1,1} = \{a\},\ S_{1,2} = \emptyset,\ S_{2,1} = \overline{\{a\}},\ S_{2,2} = \{a\},\ S_{3,1} = \overline{\{a\}},\ S_{3,2} = \overline{\{a\}}$,

$$S'_1 = \overline{\overline{S_{1,1}} \times \overline{S_{1,2}}} = \overline{\overline{\{a\}} \times \overline{\emptyset}} = \overline{\{b\} \times \{a, b\}} = \overline{\{(b, a), (b, b)\}} = \{(a, a), (a, b)\},$$

$$S'_2 = \overline{\overline{S_{2,1}} \times \overline{S_{2,2}}} = \overline{\overline{\overline{\{a\}}} \times \overline{\{a\}}} = \overline{\{a\} \times \{b\}} = \overline{\{(a, b)\}} = \{(a, a), (b, a), (b, b)\},$$

$$S'_3 = \overline{\overline{S_{3,1}} \times \overline{S_{3,2}}} = \overline{\overline{\overline{\{a\}}} \times \overline{\overline{\{a\}}}} = \overline{\{a\} \times \{a\}} = \overline{\{(a, a)\}} = \{(a, b), (b, a), (b, b)\},$$

$S'_1 \cap S'_2 \cap S'_3 = \emptyset$ and $d(x') = \{(a,a),(a,b),(b,a),(b,b)\}$

Signed MaxSAT $(i,2)$-consistency resolution:

$$\begin{array}{c} \{a\}{:}x_1 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \\ \hline\hline \Box \end{array} \quad \begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \vee \{b\}{:}x_3 \\ \hline\hline \{b\}{:}x_3 \end{array} \quad \begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \vee \{a\}{:}x_4 \\ \hline\hline \{b\}{:}x_3 \vee \{a\}{:}x_4 \\ \{(a,a)\}{:}x' \vee \{b\}{:}x_3 \vee \overline{\{a\}}{:}x_4 \\ \{(a,b),(b,a),(b,b)\}{:}x' \vee \overline{\{b\}}{:}x_3 \vee \{a\}{:}x_4 \end{array}$$

Signed MaxSAT $(i,2)$-consistency resolution (alternative version):

$$\begin{array}{c} \{a\}{:}x_1 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \\ \hline\hline \Box \end{array} \quad \begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \vee \{b\}{:}x_3 \\ \hline\hline \{b\}{:}x_3 \end{array} \quad \begin{array}{c} \{a\}{:}x_1 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \{a\}{:}x_2 \vee \{b\}{:}x_3 \\ \overline{\{a\}}{:}x_1 \vee \overline{\{a\}}{:}x_2 \vee \{a\}{:}x_4 \\ \hline\hline \{b\}{:}x_3 \vee \{a\}{:}x_4 \\ \{a\}{:}x_1 \vee \{b\}{:}x_2 \vee \{b\}{:}x_3 \vee \overline{\{a\}}{:}x_4 \\ \{b\}{:}x_1 \vee \{a\}{:}x_2 \vee \{b\}{:}x_3 \vee \overline{\{a\}}{:}x_4 \\ \{b\}{:}x_1 \vee \{b\}{:}x_2 \vee \{b\}{:}x_3 \vee \overline{\{a\}}{:}x_4 \\ \{a\}{:}x_1 \vee \{a\}{:}x_2 \vee \overline{\{b\}}{:}x_3 \vee \{a\}{:}x_4 \end{array}$$

5.4. A directional local consistency algorithm

Before we give the algorithm and prove its correctness we need some definitions and a lemma.

Definition 43. A multiset of clauses $\mathcal{C}$ is said to be i-saturated with respect to x if the multiset of all the clauses of $\mathcal{C}$ that have $\leqslant i$ literals is saturated with respect to x.

Definition 44. A CSP is directional i-consistent, for $i \geqslant 1$, iff there exists an ordering of variables such that any consistent instantiation of $i-1$ variables can be extended to a consistent instantiation of any additional variable that is greater in the ordering.

A WCSP is *directional i-consistent*, for $i \geqslant 1$, iff there exists an ordering of variables such that any instantiation of $i-1$ variables V that satisfies all constraints C_D with scope D, where $\emptyset \neq D \subseteq V$, can be extended to an instantiation of any additional variable x greater in the ordering, and satisfying all constraints $C_{D'}$ with scope D', where $\emptyset \neq D' \subseteq V \cup \{x\}$.

Now, from the proof sketch of Theorem 21, we can extract an algorithm for applying complete and incomplete inference on WCSP. The only difference with the saturation procedure described in it is that we apply inference only to clauses of a bounded number of literals. This bound on the number of literals is the parameter i in the saturation function. Function *saturation*$(\mathcal{C}_{s-1}, i, x_s)$ computes a saturation of $\mathcal{C}_{s-1}$ w.r.t. x_i applying the strong signed MaxSAT $(i-1,1)$-consistency resolution rule resolving x_s until it obtains a i-saturated set. Lemma 20 ensures that this process terminates, in particular that it does not cycle.

Function *partition*$(\mathcal{C}, x_s)$ computes a partition of $\mathcal{C}$, already saturated, into the subset of clauses containing x_s and the subset of clauses not containing x_s.

Signed MaxSAT DP algorithm enforcing directional i-consistency:

```
input: A WCSP instance $P$, an index $i$
$\mathcal{C}_0 :=$ signed_encoding$(P)$
for $s := 1$ to $n$
  $\mathcal{C} :=$ saturation$(\mathcal{C}_{s-1}, i, x_s)$
  $\langle \mathcal{C}_s, \mathcal{D}_s \rangle :=$ partition$(\mathcal{C}, x_s)$
endfor
output: $\mathcal{C}_n \cup \bigcup_{s=1}^{n} \mathcal{D}_s$
```

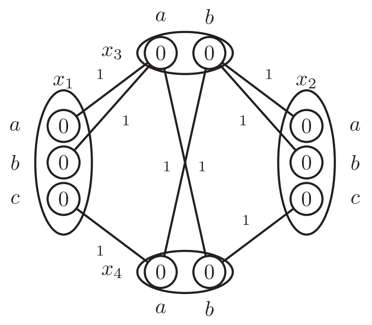
Given an initial WCSP instance P with n variables, the above algorithm returns an equivalent WCSP instance. The order on the saturation of the variables can be freely chosen, i.e. the sequence $x_1, \ldots x_n$ can be any enumeration of the variables. Notice that if i is the number of variables, the previous algorithm is complete.

Lemma 45. *The previous algorithm enforces directional i-consistency, i.e. the WCSP that corresponds to the set $\mathcal{C}_n \cup \bigcup_{s=1}^{n} \mathcal{D}_s$ is directional i-consistent. Also equivalently, the CSP that corresponds to $\bigcup_{s=1}^{n} \mathcal{D}_s$ is directional i-consistent.*

Proof. The argument is similar to the proof of lemma 17. Consider the variable ordering $x_n, \ldots, x_1$. Pick any set of $i-1$ variables V, and let I be a consistent instantiation of the $i-1$ variables V. W.l.o.g. let x_l be the greatest of these variables. I is a consistent instantiation of the CSP that corresponds to $\bigcup_{s=l}^{n}\mathcal{D}_s$. i.e. I satisfies all constraints that correspond to $\bigcup_{s=1}^{n}\mathcal{D}_s$ and with scope a subset of V. Now we will show how we can extend the assignment to any variable x_t consistently where t is such that $1 \leqslant t \leqslant l-1$.

Let $\mathcal{E} = \{S_1 : x_t \vee D_1, \ldots, S_k : x_t \vee D_k\}$ be the set of all clauses on the variables $V \cup \{x_t\}$ still not satisfied by I. Note that $\mathcal{E} \subseteq \mathcal{D}_t$. Since $\mathcal{D}_t$ is i-saturated with respect to x_t, either $S_1 \cap \cdots \cap S_k \neq \emptyset$ or $D_1 \cup \cdots \cup D_k$ is a tautology. If $S_1 \cap \cdots \cap S_k \neq \emptyset$, we extend I to assign a value inside the intersection for the variable x_t. If $S_1 \cap \cdots \cap S_k = \emptyset$, then $D_1 \cup \cdots \cup D_k$ is a tautology. Then, there exist $S_{i_1} : y \in D_{i_1}, \ldots, S_{i_l} : y \in D_{i_l}, l \leqslant k$, such that $S_{i_1} \cup \cdots \cup S_{i_l} = N$. Then, I satisfies one of these literals and the corresponding clause. So, some of the clauses are not in $\mathcal{E}$. Contradiction. □

Example 46. Fig. 2 shows the application of the algorithm to a WCSP, with parameter $i = 3$ and the order on the variables given by the sequence x_1, x_2, x_3, x_4. C_1 and D_1 are obtained applying the signed MaxSAT (2,1)-consistency resolution rule on clauses 1,2,3 with resolving variable x_1. C_2 and D_2 are obtained applying the signed MaxSAT (2,1)-consistency resolution rule on clauses 6,7,8 with resolving variable x_2. C_3 is obtained applying the signed MaxSAT (1,1)-consistency on clauses 4,12 and 5,9 with the resolving variable x_3. Finally, the empty clause is obtained from clauses 15,16 with resolving variable x_4.



$$\begin{array}{rl}
1: & \overline{\{a\}} : x_3 \vee \overline{\{a\}} : x_1 \\
2: & \overline{\{a\}} : x_3 \vee \overline{\{b\}} : x_1 \\
3: & \overline{\{a\}} : x_4 \vee \overline{\{c\}} : x_1 \\
4: & \overline{\{b\}} : x_4 \vee \overline{\{a\}} : x_3 \\
5: & \overline{\{a\}} : x_4 \vee \overline{\{b\}} : x_3 \\
6: & \overline{\{b\}} : x_3 \vee \overline{\{a\}} : x_2 \\
7: & \overline{\{b\}} : x_3 \vee \overline{\{b\}} : x_2 \\
8: & \overline{\{b\}} : x_4 \vee \overline{\{c\}} : x_2
\end{array}$$

C_1 :

$$\boxed{\begin{array}{rl}
4: & \overline{\{b\}} : x_4 \vee \overline{\{a\}} : x_3 \\
5: & \overline{\{a\}} : x_4 \vee \overline{\{b\}} : x_3 \\
6: & \overline{\{b\}} : x_3 \vee \overline{\{a\}} : x_2 \\
7: & \overline{\{b\}} : x_3 \vee \overline{\{b\}} : x_2 \\
8: & \overline{\{b\}} : x_4 \vee \overline{\{c\}} : x_2 \\
9: & \overline{\{a\}} : x_4 \vee \overline{\{a\}} : x_3
\end{array}}$$

D_1 :

$$\boxed{\begin{array}{rl}
10: & \{c\} : x_1 \vee \overline{\{a\}} : x_3 \vee \{a\} : x_4 \\
11: & \overline{\{c\}} : x_1 \vee \{a\} : x_3 \vee \overline{\{a\}} : x_4
\end{array}}$$

C_2 :

$$\boxed{\begin{array}{rl}
4: & \overline{\{b\}} : x_4 \vee \overline{\{a\}} : x_3 \\
5: & \overline{\{a\}} : x_4 \vee \overline{\{b\}} : x_3 \\
9: & \overline{\{a\}} : x_4 \vee \overline{\{a\}} : x_3 \\
12: & \overline{\{b\}} : x_4 \vee \overline{\{b\}} : x_3
\end{array}}$$

D_2 :

$$\boxed{\begin{array}{rl}
13: & \{c\} : x_2 \vee \overline{\{b\}} : x_3 \vee \{b\} : x_4 \\
14: & \overline{\{c\}} : x_2 \vee \{b\} : x_3 \vee \overline{\{b\}} : x_4
\end{array}}$$

C_3 :

$$\boxed{\begin{array}{rl}
15: & \overline{\{b\}} : x_4 \\
16: & \overline{\{a\}} : x_4
\end{array}}$$

D_3 :

C_4 :

$$\boxed{\begin{array}{rl}
17: & \square
\end{array}}$$

D_4 :

$$\begin{array}{rl}
1: & \overline{\{a\}} : x_3 \vee \overline{\{a\}} : x_1 \\
2: & \overline{\{a\}} : x_3 \vee \overline{\{b\}} : x_1 \\
3: & \overline{\{a\}} : x_4 \vee \overline{\{c\}} : x_1 \\
\hline\hline
9: & \overline{\{a\}} : x_4 \vee \overline{\{a\}} : x_3 \\
10: & \{c\} : x_1 \vee \overline{\{a\}} : x_3 \vee \{a\} : x_4 \\
11: & \overline{\{c\}} : x_1 \vee \{a\} : x_3 \vee \overline{\{a\}} : x_4
\end{array}
\qquad
\begin{array}{rl}
6: & \overline{\{b\}} : x_3 \vee \overline{\{a\}} : x_2 \\
7: & \overline{\{b\}} : x_3 \vee \overline{\{b\}} : x_2 \\
8: & \overline{\{b\}} : x_4 \vee \overline{\{c\}} : x_2 \\
\hline\hline
12: & \overline{\{b\}} : x_4 \vee \overline{\{b\}} : x_3 \\
13: & \{c\} : x_2 \vee \overline{\{b\}} : x_3 \vee \{b\} : x_4 \\
14: & \overline{\{c\}} : x_2 \vee \{b\} : x_3 \vee \overline{\{b\}} : x_4
\end{array}$$

$$\begin{array}{rl}
4: & \overline{\{b\}} : x_4 \vee \overline{\{a\}} : x_3 \\
12: & \overline{\{b\}} : x_4 \vee \overline{\{b\}} : x_3 \\
\hline\hline
15: & \overline{\{b\}} : x_4 \\
16: & \overline{\{a\}} : x_4
\end{array}
\qquad
\begin{array}{rl}
15: & \overline{\{b\}} : x_4 \\
16: & \overline{\{a\}} : x_4 \\
\hline\hline
 & \square
\end{array}$$

Fig. 2. Example of application of the directional local consistency algorithm to a WCSP. The WCSP has the set of variables $X = \{x_1, x_2, x_3, x_4\}$ with domains $d(x_1) = d(x_2) = \{a, b, c\}$ and $d(x_3) = d(x_4) = \{a, b\}$. The optimal cost is 1.

6. Conclusions

We have defined two resolution rules, called signed MaxSAT binary resolution and signed MaxSAT parallel resolution, and proved that they are sound and complete for signed MaxSAT. Based on these two rules, we have described a complete algorithm for solving Signed MaxSAT and Weighted CSP. This algorithm is the first generalization of the Davis–Putnam algorithm to Signed MaxSAT. It follows the bucket elimination schema for CSP or Weighted CSP but using the more compact language of clauses.

Eventhough Davis–Putnam or Bucket elimination type algorithms are no efficient in general, they can be useful for certain kinds of instances, as pointed out by [22], as long as we use them in combination with branch and bound techniques. The use of the more compact language of clauses in this type of algorithms, can make the algorithms even more practical with some instances.

We have also introduced a restriction and a generalization called signed MaxSAT i-consistency resolution and signed MaxSAT (i,j)-consistency resolution, respectively. If a WCSP signed encoding is closed under signed MaxSAT i-consistency, then the WCSP is i-consistent, and if it is closed under signed MaxSAT (i,j)-consistency, then the WCSP is (i,j)-consistent.

The new rules for local consistency we have introduced could be used to speed-up the search of exact signed MaxSAT algorithms, following the strategy of [20]. In the work of [20], a satisfiability-based algorithm for MaxSAT is combined with incomplete resolution rules. Given that satisfiability-based MaxSAT algorithms are the most efficient for industrial instances, incorporating our local consistency rules into these types of algorithms could give a good performance for signed MaxSAT and WCSP.

Finally, we have described an incomplete algorithm that applies directional soft consistency using the previous rules, that enforces directional i-consistency.

Acknowledgement

We would like to thank Javier Larrosa for insightful discussions.

References

[1] C. Ansótegui, M.L. Bonet, J. Levy, F. Manyà, Inference rules for high-order consistency in weighted CSP, in: Proc. of the 22th National Conf. on Artificial Intelligence, AAAI-07, AAAI Press, 2007, pp. 167–172.

[2] C. Ansótegui, M.L. Bonet, J. Levy, F. Manyà, The logic behind weighted CSP, in: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, IJCAI-07, 2007, pp. 32–37.

[3] C. Ansótegui, J. Larrubia, C.M. Li, F. Manyà, Exploiting multivalued knowledge in variable selection heuristics for sat solvers, Annals of Mathematics and Artificial Intelligence 49 (1–4) (2007) 191–205.

[4] M. Baaz, C.G. Fermüller, Resolution-based theorem proving for many-valued logics, Journal of Symbolic Computation 19 (1995) 353–391.

[5] B. Beckert, R. Hähnle, F. Manyà, The SAT problem of signed CNF formulas, in: D. Basin, M. D'Agostino, D. Gabbay, S. Matthews, L. Viganò (Eds.), Labelled Deduction, Applied Logic Series, vol. 17, Kluwer, Dordrecht, 2000, pp. 61–82.

[6] R. Béjar, F. Manyà, A. Cabiscol, C. Fernández, C. Gomes, Regular-sat: a many-valued approach to solving combinatorial problems, Discrete Applied Mathematics 155 (12) (2007) 1613–1626.

[7] M.L. Bonet, J. Levy, F. Manyà, A complete calculus for Max-SAT, in: Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing, SAT'06, Lecture Notes in Computer Science, vol. 4121, Springer, 2006. pp. 240–251.

[8] M.L. Bonet, J. Levy, F. Manyà, Resolution for Max-SAT, Artificial Intelligence 171 (8–9) (2007) 606–618.

[9] M. Davis, H. Putnam, A computing procedure for quantification theory, Journal of the ACM 7 (3) (1960) 201–215.

[10] S. de Givry, F. Heras, J. Larrosa, M. Zytnicki, Existencial arc consistency: getting closer to full arc consistency in weighted CSPs, in: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI-05, 2005, pp. 84–89.

[11] R. Dechter, Bucket elimination: a unifying framework for reasoning, Artificial Intelligence 113 (1–2) (1999) 41–85.

[12] R. Dechter, I. Rish, Directional resolution: the Davis–Putnam procedure, revisited, in: Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'94, 1994, pp. 134–145.

[13] R. Dechter, I. Rish, Mini-buckets: a general scheme for bounded inference, Journal of ACM 50 (2) (2003) 107–153.

[14] H. Ganzinger, V. Sofronie-Stokkermans, Chaining techniques for automated theorem proving in many-valued logics, in: Proc. 30th IEEE Symposium on Multiple-Valued Logic, IEEE Computer Society Press, 2000, pp. 337–344.

[15] R. Hähnle, Efficient deduction in many-valued logics, in: Proc. of the 24th Int. Symp. on Multiple-Valued Logics, ISMVL'94, IEEE Press, 1994, pp. 240–249.

[16] R. Hähnle, Many-valued logic and mixed integer programming, Annals of Mathematics and Artificial Intelligence 12 (1994) 231–263.

[17] R. Hähnle, Short conjunctive normal forms in finitely-valued logics, Journal of Logic and Computation 4 (6) (1994) 905–927.

[18] R. Hähnle, Exploiting data dependencies in many-valued logics, Journal of Applied Non-Classical Logics 6 (1996) 49–69.

[19] R. Hähnle, Advanced many-valued logics, in: D. Gabbay (Ed.), Handbook of Philosophical Logic, second ed., vol. D2: Classical Logics 2, Kluwer, Dordrecht, 2001 (Chapter 5).

[20] F. Heras, J. Marques-Silva, Read-once resolution for unsatisfiability-based max-sat algorithms, in: IJCAI, 2011, pp. 572–577

[21] S. Jain, E. O'Mahony, M. Sellmann, A complete multi-valued SAT solver, in: Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming, LNCS, vol. 6308, Springer, CP-2010, St. Andrews, Scotland, 2010, pp. 281–296.

[22] J. Larrosa, R. Dechter, Boosting search with variable elimination in constraint optimization and constraint satisfaction problems, Constraints 8 (3) (2003) 303–326.

[23] J. Larrosa, F. Heras, S. de Givry, A logical approach to efficient Max-SAT solving, Artificial Intelligence 172 (2–3) (2008) 204–233.

[24] J. Larrosa, T. Schiex, In the quest of the best form of local consistency for weighted CSP, in: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence, IJCAI-03, Morgan Kaufmann, 2003. pp. 239–244.

[25] J. Larrosa, T. Schiex, Solving weighted CSP by maintaining arc-consistency, Artificial Intelligence 159 (1–2) (2004) 1–26.

[26] C.M. Li, F. Manyà, MaxSAT, hard and soft constraints, in: A. Biere, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, IOS Press, 2009. pp. 613–631.

[27] C. Liu, A. Kuehlmann, M.W. Moskewicz, CAMA: a multi-valued satisfiability solver, in: Proceedings of the 2003 International Conference on Computer-Aided Design (ICCAD'03), San Jose, CA, USA, 2003, pp. 326–333.
[28] J.J. Lu, N.V. Murray, E. Rosenthal, A framework for automated reasoning in multiple-valued logics, Journal of Automated Reasoning 21 (1) (1998) 39–67.
[29] P. Meseguer, F. Rossi, T. Schiex, Soft constraints, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, Academic Press, 2006, pp. 281–328.