A Complete Resolution Calculus for Signed Max-SAT[*]

Carlos Ansótegui
DIEI, UdL
Lleida, Spain

María L. Bonet
LSI, UPC
Barcelona, Spain

Jordi Levy
IIIA, CSIC
Bellaterra, Spain

Felip Manyà
DIEI, UdL
Lleida, Spain

Abstract

We define a resolution-style rule for solving the Max-SAT problem of Signed CNF formulas (Signed Max-SAT) and prove that our rule provides a complete calculus for that problem. From the completeness proof we derive an original exact algorithm for solving Signed Max-SAT. Finally, we present some connections between our approach and the work done in the Weighted CSP community.

1 Introduction

In the last years, there has been an increasing interest in the Boolean Max-SAT problem. Taking into account the success of SAT on solving NP-complete decision problems, the SAT community investigates how to transfer the technology created for SAT to Max-SAT with the aim of developing fast Max-SAT solvers, which can be used to solve NP-hard optimization problems via their reduction to Max-SAT.

The most recent and relevant results for Max-SAT can be summarized as follows: (i) there exists solvers like MaxSatz [8, 9], and Toolbar [7] which solve many instances that are beyond the reach of the solvers existing just five years ago; (ii) a complete resolution-style calculus preserving the number of unsatisfied clauses has been defined for Max-SAT [6], (iii) sound resolution refinements have been incorporated into Max-SAT solvers [7], (iv) formalisms like Partial Max-SAT have been investigated for solving problems with soft constraints [4], and (v) an evaluation of Max-SAT solvers has been performed for the first time as a co-located event of the International Conference on Theory and Applications of Satisfiability Testing (SAT-2006).

During the last decade, our research program has focused on many-valued satisfiability and related problems. Our aim is bridging the gap between Boolean SAT/MaxSAT encodings and constraint satisfaction formalisms. The challenge is to combine the inherent efficiencies of Boolean SAT/MaxSAT solvers operating on uniform encodings with the much more compact and natural representations, and more sophisticated propagation techniques of CSP/Weighted CSP formalisms. Regarding many-valued Max-SAT, we have recently started exploring the role that many-valued CNF formulas can play on solving NP-hard combinatorial optimization problems via their reduction to many-valued Max-SAT. The first results were presented in ISMVL 2006 [3].

In this paper, we define a resolution-style rule for solving the Max-SAT problem of Signed CNF formulas (Signed Max-SAT) and prove that our rule provides a complete calculus for that problem. From the completeness proof we derive an original exact algorithm for solving Signed Max-SAT. Finally, we present some connections between our approach and the work done in the Weighted CSP community.

The structure of the paper is as follows. Section 2 contains preliminary definitions and the signed encoding. Section 3 defines the inference rule for signed Max-SAT and proves its soundness and completeness. Section 4 describes an exact algorithm for solving Weighted CSP. Section 5 relates our work with Weighted CSP results.

2 Preliminaries

Definition 1. A *truth value set*, or *domain*, N is a non-empty finite set $\{i_1, i_2, \ldots, i_n\}$ where n denotes its cardinality. A *sign* is a subset $S \subseteq N$ of truth values. A *signed literal* is an expression of the form $S{:}p$, where S is a sign and p is a propositional variable. We say that S is the support of p. The *complement* of a signed literal l of the form $S{:}p$, denoted by $\overline{l}$, is $\overline{S}{:}p = (N \setminus S){:}p$. A *signed clause* is a disjunction of signed literals. A *signed CNF formula* is a *multiset* of 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

[*]This research was founded by the MEC research projects iDEAS (TIN2004-04343), Mulog (TIN2004-07933-C03-01/03) and IEA (TIN2006-15662-C02-02).

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 *Signed Max-SAT problem* for a signed CNF formula consists of finding an assignment that minimizes the number of falsified signed clauses.

3 The Inference Rule

We define a resolution rule for solving signed Max-SAT, called *Signed Max-SAT Resolution*, and prove its soundness and completeness. This rule was inspired by previous works [6, 7] for Max-SAT. The completeness proof for signed CNF formulas is technically more involved than the proof for Boolean CNF formulas in [6].

Definition 4. The *Signed Max-SAT Resolution* rule is defined as follows

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

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.
The tautologies concluded by the rule like $N{:}x \vee A$ are removed from the resulting multiset. Also we 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.

We would like to emphasize that the rule does not add the conclusions to the premises. It replaces the clauses in the premises by the clauses in the conclusions.

Definition 5. We write $\mathcal{C} \vdash \mathcal{D}$ when the multiset of clauses $\mathcal{D}$ can be obtained from the multiset $\mathcal{C}$ applying the rule finitely many times. We write $\mathcal{C} \vdash_x \mathcal{C}'$ when this sequence of applications only resolves the variable x.

In the context of Max-SAT problems, an inference rule is *sound* iff the number of falsified clauses in the premises is equal to the number of falsified clauses in the conclusions for any assignment.

Theorem 6 Soundness. *The signed Max-SAT 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 $S'{:}x \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{i-1} \vee \overline{a_i}$ 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. ■

Definition 7. A multiset of clauses $\mathcal{C}$ is said to be *saturated w.r.t. x* if, for every pair of clauses $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ of $\mathcal{C}$, i) there are literals $S_1{:}y$ in A and $S_2{:}y$ in B such that $S_1 \cup S_2 = N$, or ii) $S \cap S' = S$ or $S \cap S' = S'$.
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.

We assign to every clause C a score $s(C)$ equal to the number of assignments to the variables that falsify C. The score of a multiset of clauses is the sum of scores of the clauses contained in it.

Lemma 8. *For every multiset of clauses $\mathcal{C}$ and variable x, there exists a multiset $\mathcal{C}'$ such that $\mathcal{C}'$ is a saturation of $\mathcal{C}$ w.r.t. x.*

PROOF: We proceed by applying nondeterministically the inference rule resolving x, until we obtain a saturated multiset. We only need to prove that this process terminates in finitely many inference steps, i.e that there does not exist infinite sequences $\mathcal{C} = \mathcal{C}_0 \vdash \mathcal{C}_1 \vdash \dots$, where at every inference we resolve the variable x and none of the sets $\mathcal{C}_i$ are saturated. Let M be the score of $\mathcal{C}$.

Let us partition the multiset $\mathcal{C}$ of clauses into n multisets (n is the size of the domain), $\{B_0, B_1, \dots, B_{n-1}\}$, where B_i contains the clauses where the cardinality of the support of x is i. Notice that B_0 is the multiset of clauses that do not contain the variable x. Let us denote by $s(B_i)$ the score of the multiset B_i.

We will look at these n multisets as a word of length n and base $M + 1$. So our multisets will be represented by the number $s(B_0)\, s(B_1) \cdots s(B_{n-1})$, taking $s(B_0)$ as the most significant digit. Since B_i is a subset of $\mathcal{C}$, for $i = 0, \dots, n-1$, $s(B_i) \leq M$.

When we apply our inference rule, we take two clauses, say one from B_i and one from B_j and substitute them by a set of clauses that we will distribute among the different B_k's. Now we have a new multiset of clauses and by the soundness of our rule the score of the new multiset is the same. But, if we again look at the multiset as a number in base M, the number will be different. We will argue that for each inference step, the number increases. Say that the clauses we do inference are $S{:}x \vee A \in B_{|S|}$ and $S'{:}x \vee B \in B_{|S'|}$. By the inference step we remove these clauses and add some clause in $B_{|S \cap S'|}$, and maybe also some clauses in $B_{|S|}$, $B_{|S'|}$ and $B_{|S \cup S'|}$. Since, by definition of saturation $S \cap S' \neq S$ and $S \cap S' \neq S'$, we know that $|S \cap S'| < |S|, |S'| < |S \cup S'|$, hence the digit of $B_{|S \cap S'|}$ is more significant than the digits of $B_{|S|}$, $B_{|S'|}$ and $B_{|S \cup S'|}$. We have to conclude that the new M-base number after the inference step is larger than before. Since the largest possible number we can obtain is the one represented as $s(B_0)s(B_1)\cdots s(B_{n-1}) = M0\cdots0$ the saturation procedure for x has to finish before M^n steps. ∎

Lemma 9. *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 doesn't satisfy any of the clauses. Our aim is to show that the intersection of all the supports of x in $(\mathcal{E} - \mathcal{E}')_F$ is non-empty. This way we will extend I by assigning x to a value in the intersection of all the supports.

Since $\mathcal{E}$ is saturated, for every pair of clauses $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ in $(\mathcal{E} - \mathcal{E}')_F$ either condition i) or ii) of the definition happens. Condition i) cannot happen because C_1 and C_2 cannot both be in $(\mathcal{E} - \mathcal{E}')_F$. Therefore, for every pair of clauses, $C_1 = S{:}x \vee A$ and $C_2 = S'{:}x \vee B$ in $(\mathcal{E} - \mathcal{E}')_F$, $S \cap S' = S$ or $S \cap S' = S'$. The relation $\subseteq$ constitutes a *total* order on the supports of x. The minimal support is unique, equal to the intersection of all the supports, and, like any support, non-empty. ∎

Theorem 10 Completeness. *For any multiset of clauses $\mathcal{C}$, we have* $\mathcal{C} \vdash \underbrace{\Box, \dots, \Box}_{m}, \mathcal{D}$,
where $\mathcal{D}$ is a satisfiable multiset of clauses, and m is the minimum number of unsatisfied clauses of $\mathcal{C}$.

PROOF: Let $x_1, \dots, x_n$ be any list of the variables of $\mathcal{C}$. We construct two sequences of multisets $\mathcal{C}_0, \dots, \mathcal{C}_n$ and $\mathcal{D}_1, \dots, \mathcal{D}_n$ such that (i) $\mathcal{C} = \mathcal{C}_0$, (ii) for $i = 1, \dots, n$, $\mathcal{C}_i \cup \mathcal{D}_i$ is a saturation of $\mathcal{C}_{i-1}$ w.r.t. x_i, and (iii) for $i = 1, \dots, n$, $\mathcal{C}_i$ is a multiset of clauses not containing $x_1, \dots, x_i$, and $\mathcal{D}_i$ is a multiset of clauses containing the variable x_i.

By lemma 8, these sequences can effectively be computed: for $i = 1, \dots, n$, 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 $\{\Box, \dots, \Box\}$.

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 $i = 1, \dots, n$, let $\mathcal{E}_i = \mathcal{D}_i \cup \dots \cup \mathcal{D}_n$, and let $\mathcal{E}_{n+1} = \emptyset$. Notice that, for $i = 1, \dots, n$,

1. the multiset $\mathcal{E}_i$ only contains the variables $\{x_i, \dots, x_n\}$,
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.

Claim 1 and 3 are trivial. For claim 2, notice that, since $\mathcal{C}_i \cup \mathcal{D}_i$ is saturated w.r.t. x_i, the subset $\mathcal{D}_i$ is also saturated. Now, since $\mathcal{D}_{i+1} \cup \dots \cup \mathcal{D}_n$ does not contain x_i, the set $\mathcal{E}_i$ will be saturated w.r.t. x_i.

Now, we construct a sequence of assignments $I_1, \dots, I_{n+1}$, 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 9, 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 6) the inference preserves the number of falsified clauses for every assignment, $m = |\mathcal{C}_n|$ is the minimum number of unsatisfied clauses of $\mathcal{C}$. ■

4 An Exact Signed Max-SAT Solver

From the proof of Theorem 10, we can extract the following exact algorithm for solving Signed Max-SAT.

```
input: A Signed Max-SAT instance $F$ with $k$ variables
$C_0 := F$
for $i := 1$ to $k$
        $C := saturation(C_{i-1}, x_i)$
        $\langle C_i, D_i \rangle := partition(C, x_i)$
endfor
$m := |C_k|$
$I := \emptyset$
for $i := k$ downto 1
        $I := I \cup [x_i \mapsto extension(x_i, I, D_i)]$
output: $m, I$
```

Given an initial signed Max-SAT instance F with k variables, this algorithm returns the minimal number of unsatisfied clauses (m) of F and an optimal assignment I.

The function $saturation(C_{i-1}, x_i)$ computes a saturation of C_{i-1} w.r.t. x_i applying the resolution rule resolving x until it gets a saturated set. Lemma 8 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 10 does not depend on which particular saturation we take.

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

The function $extension(x_i, I, D_i)$ computes an assignment for x_i extending the assignment I, to satisfy the clauses of D_i according to Lemma 9. The function filters all clauses of 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 an intersection. It returns a value from $\cap\{S \mid S{:}x_i \vee A \in D_i$ and I falsifies $A\}$. The argumentation of the proof of Lemma 9 ensures that this intersection is not empty.

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.

5 Signed Max-SAT and Weighted CSP

In this section we present the relationship between our work and the work developed in the Constraint Programming community. We first introduce the notions of constraint satisfaction problem (CSP), weighted CSP (WCSP), and some local consistency properties fro WCSP. Then, we define an original signed Max-SAT encoding for WCSP, which allows one to solve WCSP instances with signed Max-SAT solvers. Finally, we define a refinement of our rule such that (i) its saturation can be applied in polynomial time and (ii) it is not captured by the soft arc consistency properties defined so far. See [10] and the references therein for knowing more about WCSP.

5.1 Preliminaries

Definition 11. A *constraint satisfaction problem (CSP)* instance is defined as a triple $\langle X, D, C\rangle$, where $X = \{x_1, \ldots, x_n\}$ is a set of variables, $D = \{d(x_1), \ldots, d(x_n)\}$ 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 12. 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 $v(x_i) \in d(x_i)$. An 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 13. 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 14. The Weighted Constraint Satisfaction Problem (WCSP) for a WCSP instance consists of finding an optimal solution for that instance.

We next define the most relevant WCSP local consistency properties proposed in the literature. They do not ensure global consistency of a set of constraints, but they can be enforced efficiently.

We focus on binary WCSPs. We assume the existence of a unary constraint for every variable x_i. If no such a constraint is defined, we can always define a dummy constraint as $f(a_k) = 0$ for every $a_k \in d(x_i)$. We will use the standard notation for binary WCSP in the literature: C_i will denote a unary constraint over a variable x_i, and C_{ij} will denote a binary constraint between variables x_i and

x_j; $C_i(a_k)$, where $a_k \in d(x_i)$, will denote $f(a_k)$, and $C_{ij}(a_k, b_l)$, where $a_k \in d(x_i)$ and $b_l \in d(x_j)$, will denote $f(a_k, b_k)$.

Definition 15. Variable x_i is *node consistent* if there exists a value $a_k \in d(x_i)$ such that $C_i(a_k) = 0$. A WCSP is node consistent (NC*) if every variable is node consistent.

Definition 16. Given a binary constraint C_{ij}, the value $b \in d(x_j)$ is a *simple support* for $a \in d(x_i)$ if $C_{ij}(a, b) = 0$, and is a *full support* if $C_{ij}(a, b) + C_j(b) = 0$.

Variable x_i is *arc consistent* if every value $a \in d(x_i)$ has a simple support in every constraint C_{ij}. A WCSP is arc consistent (AC*) if every variable is node and arc consistent.

Variable x_i is *full arc consistent* if every value $a \in d(x_i)$ has a full support in every constraint C_{ij}. A WCSP is full arc consistent (FAC*) if every variable is node and full arc consistent.

Definition 17. Let $>$ be a total ordering over the variables of a WCSP. Variable x_i is *directional arc consistent* (DAC) if every value $a \in d(x_i)$ has a full support in every constraint C_{ij} such that $x_j > x_i$. It is *full directional arc consistent* (FDAC) if, in addition, every value $a \in d(x_i)$ has a simple support in every constraint C_{ij} such that $x_j < x_i$. A WCSP is full directional arc consistent (FDAC*) if every variable is node and full directional arc consistent.

Definition 18. Let $>$ be a total ordering over the variables of a WCSP. Variable x_i is *existential arc consistent* if there is at least one value $a \in d(x_i)$ such that $C_i(a) = 0$ and has a full support in every constraint C_{ij}. A WCSP is existential arc consistent (EAC*) if every variable is node and existential arc consistent. A WCSP is *existential directional arc consistent* (EDAC*) if it is FDAC* and EAC*.

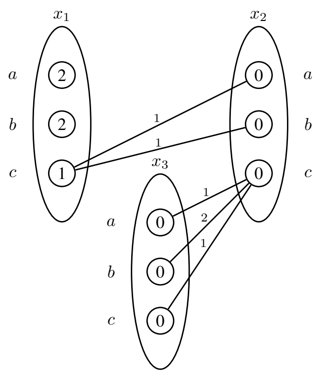
Signed Max-SAT inference rules capturing the above arc consistency properties are defined in [1].

5.2 Encoding WCSP into Signed Max-SAT

We next define how to encode WCSP instances as signed Max-SAT instances. This way, signed Max-SAT solvers can be used to solve WCSP instances.

Definition 19. The *signed encoding* of a WCSP instance $\langle X, D, C\rangle$ is the signed CNF formula over the domain $N = \bigcup_{x_i \in D} d(x_i)$ that contains for every possible tuple $\langle b_{i_1}, \ldots, b_{i_k}\rangle \in d(x_{i_1}) \times \cdots \times d(x_{i_k})$ 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: $\overline{\{b_{i_1}\}}{:}x_{i_1} \vee \cdots \vee \overline{\{b_{i_k}\}}{:}x_{i_k}$.

An alternative encoding is to consider signed clauses with weights instead of allowing multiple copies of a clause.



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

Figure 1. Example of signed encoding

For the sake of clarity we use unweighted clauses. Nevertheless, any efficient implementation of the algorithms proposed should deal with weighted clauses. The extension of our theoretical results to weighted clauses is straightforward.

Proposition 20. *Solving a WCSP instance is equivalent to solving the Signed Max-SAT problem of its signed encoding.*

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. ■

We plan to investigate other encodings from WCSP to Signed Max-SAT and investigate their complexity in the style of the satisfiability preserving encodings defined in [5, 2].

Example 1. Figure 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.

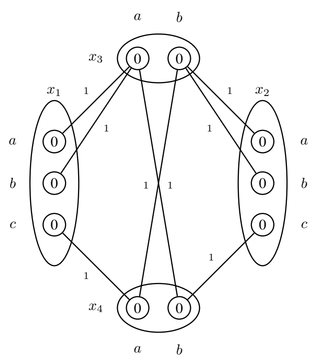
5.3 A refinement of signed Max-SAT resolution

We define a refinement of our rule that (i) it is sound but incomplete, (ii) its saturation can be applied in polynomial time, and (ii) it is not captured by the soft arc consistency properties defined so far.

Definition 21. The *Signed Max-SAT Binary Resolution* rule is defined as follows

$$\begin{array}{l} S{:}x \vee a \\ S'{:}x \vee b \\ \hline\hline S \cap S'{:}x \vee a \vee b \\ S \cup S'{:}x \vee a \vee b \\ S{:}x \vee a \vee \overline{b} \\ S'{:}x \vee b \vee \overline{a} \end{array}$$

It is obtained by restricting the signed Max-SAT resolution rule to be applied on clauses with at most two literals.



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

$9 : (1, 2, x_1)$	$\{c\} : x_1 \vee \overline{\{a\}} : x_3$
$10 : (3, 9, x_1)$	$\overline{\{a\}} : x_3 \vee \overline{\{a\}} : x_4$
$11 : (3, 9, x_1)$	*ternary clause*
$12 : (3, 9, x_1)$	*ternary clause*
$13 : (4, 10, x_4)$	$\overline{\{a\}} : x_3$
$14 : (7, 8, x_2)$	$\{c\} : x_2 \vee \overline{\{b\}} : x_3$
$15 : (6, 14, x_2)$	$\overline{\{b\}} : x_3 \vee \overline{\{b\}} : x_4$
$16 : (6, 14, x_2)$	*ternary clause*
$17 : (6, 14, x_2)$	*ternary clause*
$18 : (5, 15, x_4)$	$\overline{\{b\}} : x_3$
$19 : (13, 18, x_3)$	$\square$

Figure 2. Application of the signed Max-SAT binary resolution rule

Example 2. Figure 2 shows a WCSP instance and its signed encoding. The set of variables is $\{x_1, x_2, x_3, x_4\}$, with domains $d(x_1) = d(x_2) = \{a, b, c\}$ and $d(x_3) = d(x_4) = \{a, b\}$. The instance, whose optimal cost is 1, is *existential directional arc consistent*, but its cost cannot be detected enforcing the soft arc consistency properties defined so far. The first 8 signed clauses represent the initial WCSP instance. Clauses 9 to 19 are derived by applying signed Max-SAT binary resolution; e.g., clause 9 is derived from clauses 1 and 2 resolving the variable x_1. By applying signed Max-SAT binary resolution, we are able to derive the empty clause (clause 19) and compute its optimal cost.

References

[1] C. Ansótegui, M. Bonet, J. Levy, and F. Manyà. The logic behind weighted CSP. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, Hyderabad, India*, pages 32–37, 2007.

[2] C. Ansótegui and F. Manyà. Mapping problems with finite-domain variables into problems with boolean variables. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (Revised Selected Papers), SAT-2004, Vancouver, Canada*, pages 1–15. Springer LNCS 3542, 2004.

[3] J. Argelich, X. Domingo, C. M. Li, F. Manyà, and J. Planes. Towards solving many-valued MaxSAT. In *Proceedings, 36th International Symposium on Multiple-Valued Logics (ISMVL), Singapore*. IEEE CS Press, 2006.

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

[5] B. Beckert, R. Hähnle, and F. Manyà. Transformations between signed and classical clause logic. In *Proceedings, International Symposium on Multiple-Valued Logics, ISMVL'99, Freiburg, Germany*, pages 248–255. IEEE Press, 1999.

[6] M. 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*, pages 240–251. Springer LNCS, 2006.

[7] 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*, pages 193–198. Morgan Kaufmann, 2005.

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

[9] C. M. Li, F. Manyà, and J. Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *Proceedings of the 21st National Conference on Artificial Intelligence, AAAI-2006, Boston/MA, USA*, pages 86–91, 2006.

[10] P. Meseguer, F. Rossi, and T. Schiex. Soft constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, pages 281–328. Elsevier, Academic Press, 2006.