Resolution in Max-SAT and its relation to local consistency in weighted CSPs

Javier Larrosa, Federico Heras
{larrosa, fheras}@lsi.upc.edu
Dept. LSI, Universitat Politecnica de Catalunya
Barcelona, Spain

Abstract

Max-SAT is an optimization version of the well-known SAT problem. It is of great importance from both a theoretical and a practical point of view. In recent years, there has been considerable interest in finding efficient solving techniques [Alsinet *et al.*, 2003; Xing and Zhang, 2004; Shen and Zhang, 2004; de Givry *et al.*, 2003]. Most of this work focus on the computation of good quality lower bounds to be used within a branch and bound algorithm. Unfortunately, lower bounds are described in a procedural way. Because of that, it is difficult to realize the *logic* that is behind.

In this paper we introduce a logical framework for Max-SAT solving. Using this framework, we introduce an extension of the Davis-Putnam algorithm (that we call Max-DPLL) and the resolution rule. Our framework has the advantage of nicely integrating branch and bound concepts such as the lower and upper bound, as well as hiding away implementation details. We show that Max-DPLL augmented with a restricted form of resolution at each branching point is an effective solving strategy. We also show that the resulting algorithm is closely related with some local consistency properties developed for weighted constraint satisfaction problems.

1 Introduction

Since the eighties, both *boolean satisfiability* and *constraint satisfaction* have been the topic of intense algorithmic research. In both areas, the goal is to assign values to variables in such a way that no forbidden combination of values appear in the solution. In satisfiability, forbidden combinations are specified by means of *clauses*. In constraint satisfaction they are specified by means of arbitrary *constraints*. Given its similarity, it is hardly a surprise that both research communities have developed closely related techniques.

In both fields, the original decision problem (SAT and CSP, respectively) has been augmented to deal with unfeasible problems (namely, not all the clauses or constraints can be satisfied). The new goal is to find an assignment that best respects the clauses/constraints. Two well-known examples are Max-SAT [Hansen and Jaumard, 1990] and *weighted* CSP (WCSP) [Bistarelli *et al.*, 1999], where most of recent algorithmic work has focused. In both cases the importance of the clauses/constraints is given by a *weight* and the goal is to minimize the sum of weights associated with the clauses/constraints violated by the assignment. It is known that Max-SAT instances can be translated into WCSP instances and vice versa [de Givry *et al.*, 2003]. In both cases the main solving technique are enumeration algorithms based on branch and bound search.

In the CSP side, several local consistency properties have been recently generalized to the WCSP framework [Cooper, 2003; Larrosa and Schiex, 2004; 2003; de Givry *et al.*, 2005]. As a result, a new family of algorithms have been proposed. Besides being efficient, these algorithms have a highly desirable property: they can be neatly described as a basic backtracking search in which certain local consistency property is enforced at every search state.

In [de Givry *et al.*, 2003], it was shown that Max-SAT instances could be efficiently solved by translating them into equivalent WCSP instances and later using a WCSP solver. The level of local consistency maintained by the WCSP solver was fundamental in the efficiency of the algorithm.

In this paper we analyze the interpretation of WCSP local consistency properties and related algorithms when applied to Max-SAT instances. To facilitate the connection, we start by providing a reformulation of Max-SAT in which it is possible to explicitly express a lower and an upper bound of the optimal cost (Section 3). Such reformulation makes possible an elegant extension of DPLL in which each branching point is just a Max-SAT instance, similarly to what happens with DPLL in SAT (Section 4). In Section 5 we present our main contribution: a generalization of the resolution rule ($x \vee A, \bar{x} \vee B \Rightarrow A \vee B$) and the proof that neighborhood resolution (i.e., a restricted form of resolution) suffices to enforce (weighted) node and arc consistency. In Section 6 we put our approach in context with other Max-SAT algorithms. Finally, in Section 7 we provide conclusions and directions of future work.

2 Preliminaries

In the sequel $X = \{x_1, \ldots, x_n\}$ will denote a set of boolean variables. They take values over the set $\{\mathbf{t}, \mathbf{f}\}$, which stands

for *true* and *false*, respectively. A *literal* l is either a variable x or its negation $\bar{x}$. l stands for the negation of l.

If variable x is instantiated to **t**, noted $x \leftarrow \mathbf{t}$ literal x is satisfied and literal $\bar{x}$ is falsified. Similarly, if x is instantiated to **f**, $\bar{x}$ is satisfied and x is falsified. An *assignment* is an instantiation of a subset of the variables. The assignment is *complete* if it instantiates all the variables in X (otherwise it is partial). An assignment satisfies a clause (i.e., a disjunction of literals) C iff it satisfies one or more of its literals. It satisfies a set of clauses $\mathcal{F}$ iff it satisfies all its clauses. A satisfying complete assignment is called a *model*. Given a boolean formula encoded by a set of clauses $\mathcal{F}$, the SAT problem consists in determining whether there is any model for it or not.

We will use the symbol $\square$ to denote the *empty clause* which, obviously, cannot be satisfied. When $\square \in \mathcal{F}$ we say that there is an explicit *contradiction*.

When there is no model for the formula $\mathcal{F}$, one may be interested in finding the complete assignment with minimum number of violated clauses. This problem is known as Max-SAT.

3 An equivalent reformulation of Max-SAT

There is a *weighted* version of Max-SAT in which (weighted) clauses are pairs (C,w) such that C is a classical clause and w is the cost of its falsification. In weighted *Max-SAT* $\mathcal{F}$ is a set of weighted clauses. The cost of an assignment is the sum of weights of all the clauses that it falsifies. The goal is to find complete assignments with minimal cost. We make the usual assumption of weights being natural numbers.

It is easy to see that Max-SAT and weighted Max-SAT have exactly the same expressive power. A Max-SAT instance can be rewritten as a weighted instance replacing every clause C by a weighted clause $(C,1)$. A weighted Max-SAT instance can be rewritten as a Max-SAT instance replacing every weighted clause (C,w) by w copies of clause C. Clearly, weighted Max-SAT encodings may be exponentially more compact than Max-SAT. Thus, in the following, we will assume, without loss of generality, weighted Max-SAT.

Following previous work in weighted constraint satisfaction [Larrosa and Schiex, 2004], we assume the existence of a known upper bound k of the optimal solution. This is also done without loss of generality because, if a tight upper bound is not known, k can be set to the sum of weights of all the clauses.

Consider the set $\mathcal{F}$ of weighed clauses. We say that a *model* is a *complete assignment with cost less than k*. Max-SAT is the problem of *finding a model of minimal cost*, if there is any.

Observe that weights $w \geq k$ indicate that the associated clause *must be necessarily satisfied*. Thus, we can replace every weight $w \geq k$ by k without changing the problem. Thus, without loss of generality we assume all costs in the interval $[0..k]$ and, accordingly, redefine the sum of costs as

$$a \oplus b = min\{a+b, k\}$$

in order to keep the result within the interval. For convenience of notation, we will refer to k as $\top$. We say that a weighted clause is *hard* (or mandatory) iff its weight is $\top$. Observe that Max-SAT with $\top = 1$ is equivalent to SAT.

Example 1 *Consider the formula* $\{(x,1),(\bar{y},5),(\bar{x} \vee y,2),(x \vee y,4)\}$*, with* $\top = 5$*. The second clause is hard. The assignment* $x = \mathbf{f}, y = \mathbf{f}$ *is not a model because its cost is* $1 \oplus 4 = \top$*. The assignment* $x \leftarrow \mathbf{t}, y \leftarrow \mathbf{f}$ *is a (optimal) model with cost* 2*.*

Note that in Max-SAT truth tables are tables with a cost associated to each truth assignments. A brute-force solving method consists in computing the truth table of the input formula and finding the minimal cost model. For instance, the truth-table of the previous formula is,

$x\ y$	cost
f f	$1 \oplus 4 = \top$
f t	$1 \oplus \top = \top$
t f	2
t t	$\top$

It is worth mentioning the role of the empty clause $(\square, w)$. Since it cannot be satisfied, w will be added to the cost of any model. Therefore, w is an explicit *lower bound* of the optimal model. When the lower bound and the upper bound have the same value (i.e., $(\square, \top) \in \mathcal{F}$) the formula does not have any model and we call this situation an explicit *contradiction*.

4 Generalization of DPLL to Max-SAT

4.1 Max-SAT Basic Simplification Rules

SAT solvers take advantage from some equivalence rules that are used to simplify the CNF formula without changing its set of models. Not all of these formulas can be applied directly to Max-SAT. In this Section we state some useful Max-SAT specific rules. We use the notation $[P,\ldots,Q] \Rightarrow [R,\ldots,S]$, where $P,Q,\ldots$ are weighted clauses. It means that if there are in $\mathcal{F}$ weighted clauses matching with $[P,\ldots,Q]$, they can be replaced by $[R,\ldots,S]$. A and B are arbitrary disjunctions of literals.

- BR1: $[(A,\top),(A \vee B,w)] \Rightarrow [(A,\top)]$
- BR2: $[(A,w),(A,u)] \Rightarrow [(A,w \oplus u)]$
- BR3: If $(w \oplus u = \top)$ then $[(A,w),(A \vee B,u)] \Rightarrow [(A,w),(A \vee B,\top)]$
- BR4: $[(A,0)] \Rightarrow [\,]$

BR1 shows that classical absorption can only be applied when the absorbing clause is hard. BR2 generalizes the standard idempotency of the conjunction: In Max-SAT the weights of the repeated clauses must be added in the resulting clause. BR3 is used to harden a soft clause. BR4 indicates that cost-free clause can be eliminated. The correctness of these equivalences is direct and we omit the proof.

Example 2 *Consider the formula* $\{(x,1),(x,1),(x \vee y,3),(x \vee y \vee z,1)\}$*, with* $\top = 5$*. We can apply BR2, which produces* $\{(x,2),(x \vee y,3),(x \vee y \vee z,1)\}$*. Now, we can apply BR3, producing* $\{(x,2),(x \vee y,\top),(x \vee y \vee z,1)\}$*. Finally, BR1 produces* $\{(x,2),(x \vee y,\top)\}$*. The equivalence between the original and the final formula can be checked by constructing and comparing the costs of the two truth-tables.*

4.2 Max-DPLL

Davis Putnam (DPLL) is the most popular algorithm for SAT and the starting point of most state-of-the-art solvers [Davis *et al.*, 1962]. It takes as input a CNF formula $\mathcal{F}$ and decides whether or not there exists a model. In this Section, we present a natural extension of DPLL to Max-SAT that we call Max-DPLL. Let $\mathcal{F}$ be a weighted CNF formula and $\top$ its upper bound. Max-DPLL($\mathcal{F},\top$) returns the cost of the optimal model of $\mathcal{F}$ if there is such a model, else it returns $\top$. The following description is inspired by the description of DPLL given in [Bacchus, 2002].

As in classical SAT, Max-DPLL performs basic simplifications on its input prior to invoking itself recursively. The *instantiation* of a variable by forcing the satisfaction of a literal l, denoted $\mathcal{F}[l]$ produces a new formula generated from $\mathcal{F}$ as follows: all clauses containing l are eliminated, and $\bar{l}$ is removed from all clauses where it appears. *Unit Clause Reduction* (UCR) is another simplification rule that selects a clause $(l,\top)$ (namely, a unit hard clause) and instantiates the corresponding variable in accordance to the literal in that clause. *Unit Propagation* (UP) is the algorithm that performs UCR and the basic simplification rules BR1 – 4 until either (*a*) a contradiction is achieved, or (*b*) there are no more possible simplifications to do.

Example 3 *Consider the application of* UP *to the formula* $\{(\square,1),(\bar{x},4),(x,1),(y,3),(\bar{y},1),(\bar{x}\vee\bar{y}\vee z,1)\}$, *with* $\top=5$. *Rule BR*3 *transforms it into* $\{(\square,1),(\bar{x},\top),(x,1),(y,3),(\bar{y},1),(\bar{x}\vee\bar{y}\vee z,1)\}$. *UCR instantiates* $\bar{x}$ *and produces* $\{(\square,3),(y,3),(\bar{y},1),(\bar{y}\vee z,1)\}$. *Again BR*3 *produces* $\{(\square,3),(y,\top),(\bar{y},1),(\bar{y}\vee z,1)\}$, *which allows UCR to instantiate y,* $\{(\square,4),(z,1)\}$. *Again BR*3 *produces* $\{(\square,4),(z,\top)\}$, *which UCR to instantiate z producing the trivial* $\{(\square,4)\}$. *The model of the original formula* $(\bar{x},y,z)$ *can be constructed by tracking back the truth assignments made by UCR.*

A recursive description of Max-DPLL is given in Algorithm 1. First, UP is applied to the input formula (line 1). If the resulting formula contains a contradiction, the algorithm returns $\top$ and backtracks (line 2). Else, if it does not contain any variable the trivial cost of the optimal model is returned (lines 3 and 4). Otherwise, an arbitrary literal l is selected (line 5). The formula is instantiated with l and $\bar{l}$ and Max-DPLL is recursively called with each case (lines 6 and 7). Observe that the first recursive call is made with the $\top$ inherited from its parent, but the second call uses the output of the first call. This implements the typical upper bound updating of branch and bound. Finally, the best value of the two recursive calls is returned (line 8). Note that if Max-DPLL is executed with a SAT instance (i.e., $\top=1$) it behaves exactly as DPLL.

Figure 1 reports some empirical evaluation of Max-DPLL on 40-variable random instances of Max-2-SAT and Max-3-SAT generated with the *Cnfgen* generator.[1] The horizontal axis indicates the number of clauses and the vertical axis indicates the search effort as the number of visited nodes. Each point is the mean over 10 instances. The results of Max-DPLL are those labelled as UP (ignore, for the moment, the other curves). As can be observe, the performance of the algorithm degenerates as the number of clauses increases.

[1] A. van Gelder ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/contributed/UCSC/instances

Function Max-DPLL($\mathcal{F},\top$) : nat
1 $\mathcal{F}$:=UP($\mathcal{F}$);
2 **if** $(\square,\top)\in\mathcal{F}$ **then return** $\top$;
3 **if** $\mathcal{F}=\emptyset$ **then return** 0;
4 **if** $\mathcal{F}=\{(\square,w)\}$ **then return** w;
5 l :=SelectLit($\mathcal{F}$);
6 v :=Max-DPLL($\mathcal{F}[l],\top$);
7 v :=Max-DPLL($\mathcal{F}[\bar{l}],v$);
8 **return** v;

Algorithm 1: Max-DPLL. $\mathcal{F}$ is a set of weighted clauses with all weights in the interval $[0,\ldots,\top]$. If the weighted formula has models, Max-DPLL returns the cost of the optimal one, else it returns $\top$

Max-DPLL can be enhanced by *dominance* rules that exploit situations where it is easy to detect that one literal is never worse than its negation. A well-known SAT case, also applicable to Max-SAT, is the *pure literal rule*. It says that if there is a literal such that it appears in the formula and its negation does not appear, then all clauses mentioning it can be removed. More sophisticated dominance rules for Max-SAT and WCSP can be found in [Xing and Zhang, 2004; de Givry, 2004].

5 Resolution for Max-SAT

While DPLL seems to be the best option to find models, *resolution* might be more appropriate to detect contradictions. [Robinson, 1965] showed that the resolution rule is sound and complete for SAT, although it is usually too space consuming. In the SAT context, the performance of DPLL has been improved by the addition of limited forms of resolution at each search node [Gelder, 1995; Rish and Dechter, 2000; Bacchus, 2002; Drake *et al.*, 2002] in order to anticipate the detection of dead-ends. In this section we generalize the resolution rule to Max-SAT. Then we show that some local consistency techniques used in [Larrosa and Schiex, 2004] are just the application of a restricted form of the weighted resolution rule.

First we define the *subtraction* of weights ($\ominus$): Let $a,b\in[0,\ldots,\top]$ be two weights such that $a\geq b$,

$$a\ominus b=\begin{cases} a-b & : \quad a\neq\top \\ \top & : \quad a=\top \end{cases}$$

The *weighted resolution* rule (RES) is defined as,

$$(x\vee A,u),(\bar{x}\vee B,w)\Rightarrow\begin{cases}(A\vee B,m)\\(x\vee A,u\ominus m)\\(\bar{x}\vee B,w\ominus m)\\(x\vee A\vee B,m)\\(\bar{x}\vee A\vee B,m)\end{cases}$$

where A and B are arbitrary disjunctions of literals and $m=\min\{u,w\}$. Variable x is called the clashing variable. Observe that in the $u=w=\top$ case the fourth and fifth new

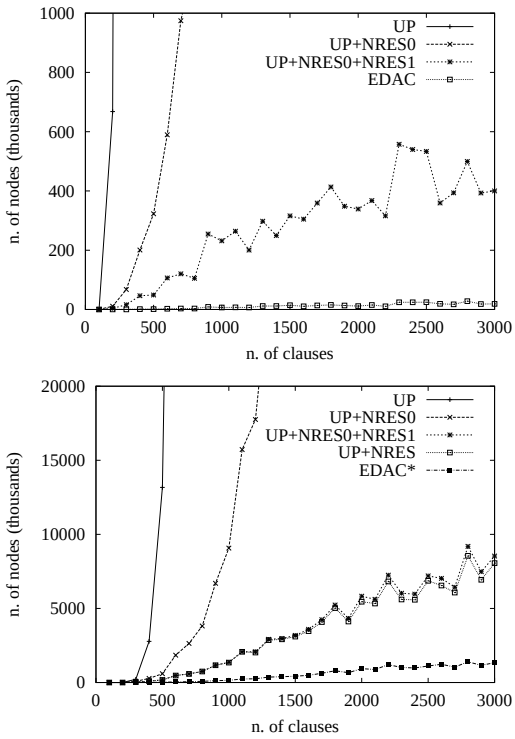

Figure 1: Experimental results on random Max-2SAT (top) and Max-3SAT (bottom).

clauses can be removed (by BR1), yielding a rule similar to classical resolution. Observe as well that in the $u = w < \top$ case, the second and third new clauses will have weight 0 and can be removed (by BR4). In the $u \neq w$ case, either the second or the third new clause will have weight 0 and can be removed (by BR4). Recall that in RES, the left-hand clauses are replaced by the right-hand, while in classical resolution right-hand clauses are just added.

Theorem 1 *The transformation rule RES is sound. Namely, it preserves the set of models and their cost.*

proof 1 *The following table contains in the first columns all the truth assignments, in the second column the cost of the assignment according to the clauses on the left-hand of the RES rule, and in the third column the cost of the assignment according to the clauses on the right-hand of the RES rule. As it can be observed, the costs the are same, so the resulting problem is equivalent.*

x A B	*Left*	*Right*
f f f	u	$m \oplus (u \ominus m)$
f f t	u	$m \oplus (u \ominus m)$
f t f	0	0
f t t	0	0
t f f	w	$m \oplus (w \ominus m)$
t f t	0	0
t t f	w	$m \oplus (w \ominus m)$
t t t	0	0

The interest of RES is that it makes explicit some previously hidden connection between variables in A and B. It is well known that in the SAT case (i.e, $\top = 1$) the rule is complete (its application suffices to achieve a contradiction). We are still investigating under which other cases the completeness of RES is guaranteed. One important problem we have detected is that the fourth and fifth new weighted formulas (i.e., $(x \vee A \vee B, m)$, $(\bar{x} \vee A \vee B, m)$) may not be in conjunctive normal form.

5.1 Neighborhood Resolution

Neighborhood Resolution [Cha and Iwama, 1996] is the classical resolution rule restricted to pairs of clauses that differ only in the clashing variable. Similarly, in the Max-SAT context we define the neighborhood resolution rule (NRES) as RES restricted to the $A = B$ case, which simplifies to,

$$(x \vee A, u), (\bar{x} \vee A, w) \Rightarrow \left\{ \begin{array}{l} (A, m) \\ (x \vee A, u \ominus m) \\ (\bar{x} \vee A, w \ominus m) \end{array} \right.$$

whith $m = \min\{u, w\}$. This rule is specially useful because it projects to A costs that were implicit in the formula. We demonstrate the interest of NRES considering its application to bounded-size clauses. Let NRES$_k$ denote NRES restricted to $|A| = k$. NRES$_0$ yields,

$$(x, u), (\bar{x}, w) \Rightarrow \left\{ \begin{array}{l} (\Box, m) \\ (x, u \ominus m) \\ (\bar{x}, w \ominus m) \end{array} \right.$$

This rule is extremely useful because it produces a direct increment of the lower bound, which may raise a contradiction, or produce new unit clause reductions.

Example 4 *Consider the formula* $\{(x,1), (\bar{x},2), (y,1), (\bar{y},1), (z,1)\}$ *with* $\top = 3$. *UP is unable to simplify the problem. Nevertheless, NRES$_0$ can be applied to the first and second clauses producing* $\{(\Box,1), (\bar{x},1), (y,1), (\bar{y},1), (z,1)\}$. *Appling NRES$_0$ to the third and fourth clauses produces* $\{(\Box,2), (\bar{x},1), (z,1)\}$. *Using BR3 we obtain* $\{(\Box,2), (\bar{x},\top), (z,\top)\}$. *The two unit clauses can be reduced, producing* $\{(\Box,2)\}$.

It is interesting to observe that the application of NRES$_0$ is somehow similar to the computation of the lower bound of [Alsinet *et al.*, 2003]. The practical importance of adding NRES$_0$ to Max-DPLL is illustrated in Figure 1. The lines labelled UP+NRES0 report the efficiency of Max-DPLL when UP is augmented with NRES$_0$ until quiescence. As it can be observed NRES$_0$ produces huge savings over the very inefficient Max-DPLL.

The rule NRES$_1$ is,

$$(x \vee l, u), (\bar{x} \vee l, w) \Rightarrow \left\{ \begin{array}{l} (l, m) \\ (x \vee l, u \ominus m) \\ (\bar{x} \vee l, w \ominus m) \end{array} \right.$$

This rule is also of great interest because it increases the weight of a unary clauses, which may allow further application NRES$_0$ and UCR.

Example 5 *Consider the formula* $\{(x \vee y, 1), (\bar{x} \vee y, 1), (\bar{y}, 1), (z, 1)$ *with* $\top = 2$. *Neither* UP *nor NRES$_0$ can simplify the problem. However, NRES$_1$ produces*

$\{(\bar{y},1),(y,1),(z,1)\}$ *which allows NRES*$_0$ *to transform the problem into* $\{(\square,1),(z,1)\}$. *We can apply BR*3 *to the unary clause obtaining* $\{(\square,1),(z,\top)$. *Now, UCR produces* $\{(\square,1)\}$.

The practical importance of adding NRES$_1$ to Max-DPLL is also illustrated in Figure 1. The lines labelled UP+NRES0+NRES1 report the efficiency of Max-DPLL when UP is augmented with NRES$_0$ and NRES$_1$ until quiescence. As it can be observed, the addition of NRES$_1$ also produces huge gains. For the sake of completeness, we also evaluate the effect of NRES$_k$ for $k > 1$. Obviously, in 2-SAT it does not have any effect, since the application of NRES$_k$ requires clauses of length $k+1$. In 3-SAT, NRES$_2$ can only be applied in the original ternary clauses. Its effect is reported in Figure 1 under the label UP+NRES. It can be observed that NRES$_2$ has a very limited effect.

5.2 Neighborhood Resolution and Local Consistency

In this Section we relate the simplification rules discussed along the paper with local consistency properties developed for WCSP. In order to do so, we recall that the usual way to map a Max-SAT instance with a WCSP is to group clauses mentioning exactly the same set of variables and associate them to a cost function f defined as follows: Let $\mathcal{V} \subset \mathcal{F}$ be the group of clauses over the set of variables $Y \subset X$. $\mathcal{V}$ defines a cost function f with scope Y. Let t be an instantiation of the variables in Y. If t falsifies a clause $(C,w) \in \mathcal{V}$, $f(t) = w$, else $f(t) = 0$. Now it is straightforward to redefine the WCSP local consistency properties in Max-SAT terms. In the following consider a boolean formula $\mathcal{F}$ where $\underline{w}(\square)$ denotes the weight of the empty clause; $w(i)$ and $w(\bar{i})$ the weights of the the unary clauses x_i and $\bar{x_i}$, respectively; analogously $w(ij)$, $w(i\bar{j})$, $w(\bar{i}j)$ and $w(\bar{i}\bar{j})$ are the weights of the four possible binary clauses over x_i and x_j. If any of these clauses is not in $\mathcal{F}$ the corresponding weight is 0.

Definition 1 *$\mathcal{F}$ is node-consistent (NC) iff for all variable x_i, $w(\square) \oplus w(i) < \top$ and $w(\square) \oplus w(\bar{i}) < \top$.*

Theorem 2 *Algorithm* UP *enforces the NC property.*

proof 2 *Suppose that $\mathcal{F}$ is not NC. Then there is some clause $(l,w) \in \mathcal{F}$ such that $w(\square) \oplus w = \top$. Therefore BR*3 *can be applied replacing clause (l,w) by $(l,\top)$. It will allow the application of UCR which will eliminate the clause.*

Definition 2 *$\mathcal{F}$ is* star *node-consistent (NC*) iff it is NC and for all variable x_i, $w(i) = 0$ or $w(\bar{i}) = 0$*

Theorem 3 *Let* UP +NRES$_0$ *denote the algorithm that applies* UP *and NRES$_0$ until quiescence. It enforces NC*.*

proof 3 *We only need to proof that the application of NRES$_0$ guarantees that for all variable x_i, $w(i) = 0$ or $w(\bar{i}) = 0$. Assume that $0 < w(i) \leq w(\bar{i})$. There are two possible situations. The first one is $w(i) = \top$. In that case* UP *will trigger UCR and x_i will disappear from the formula. The second situation is $w(i) < \top$. Then, the application of NRES$_0$ will add $w(i)$ to $w(\square)$ and replace $(x_i,w(i)),(\bar{x_i},w(\bar{i}))$ by $(x_i,0),(\bar{x_i},w(\bar{i}) \ominus w(i))$.*

The algorithm to enforce NC* introduced in [Larrosa and Schiex, 2004] proves the following result,

Corollary 1 UP +NRES$_0$ *can be implemented with time complexity $O(n)$, where n is the number of variables in the formula.*

Definition 3 *$\mathcal{F}$ is arc-consistent (AC*) iff it is NC* and for all pair of variables (x_i,x_j), $\min\{w(ij),w(i\bar{j})\} = 0$ and $\min\{w(\bar{i}j),w(\bar{i}\bar{j})\} = 0$*

Theorem 4 *Let* UP +NRES$_0$ +NRES$_1$ *denote the algorithm that applies* UP, *NRES$_0$ and NRES$_1$ until quiescence. It enforces arc-consistency (AC*).*

proof 4 *We only need to proof that the application of NRES$_1$ guarantees that for all pair of variables (x_i,x_j), $\min\{w(ij),w(i\bar{j})\} = 0$ and $\min\{w(\bar{i}j),w(\bar{i}\bar{j})\} = 0$. We proof the first condition (the second is similar). Assume that $0 < w(ij) \leq w(i\bar{j})$. There are two possible situations. The first one is that $w(ij) = \top$ (which implies $w(i\bar{j}) = \top$. In that case NRES$_1$ will add $(x_i,\top)$ to the formula. Then, UCR will be trigger and x_i will disappear from the formula. The second situation is $w(ij) < \top$. Then, the application of NRES$_1$ will add $w(ij)$ to $w(i)$ and replace $(x_i \vee x_j,w(ij)),(x_i \vee \bar{x_j},w(i\bar{j}))$ by $(x_i \vee x_j,0),(x_i \vee \bar{x_j},w(i\bar{j}) \ominus w(ij))$.*

The algorithm to enforce AC* introduced in [Larrosa and Schiex, 2004] proofs the following result.

Corollary 2 UP +NRES$_0$ *+NREO can be implemented with time complexity $O(n^2)$, where n is the number of variables in the formula.*

6 Neighborhood Resolution and state-of-the-art Max-SAT lower bounds

In the last years several algorithms for Max-SAT have been proposed [Alsinet *et al.*, 2003; Xing and Zhang, 2004; Shen and Zhang, 2004]. All these works have in common a basic branch and bound algorithm. They mainly differ in the lower bound that they use. [Shen and Zhang, 2004] show that their lower bound is better (i.e., higher than or equal) than [Alsinet *et al.*, 2003] and, under some reasonable conditions, it is also better than [Xing and Zhang, 2004]. In the following, by means of two examples, we show that UP +NRES$_0$ +NRES$_1$ is not comparable with them.

Consider the formula $\{(x \vee z,1),(\bar{x} \vee z,1),(y \vee \bar{z},1),(\bar{y} \vee \bar{z},1)\}$. Both [Xing and Zhang, 2004] and [Shen and Zhang, 2004] would compute a lower bound 0. However, NRES$_1$ can be applied twice, producing the equivalent formula $\{(z,1),(\bar{z},1)\}$ and now NRES$_0$ would transform the problem into $\{(\square,1)\}$, which means a lower bound of 1.

Consider now the formula $\{(x,1),(y,1),(\bar{x} \vee \bar{y},1)\}$. While both [Xing and Zhang, 2004] and [Shen and Zhang, 2004] would compute a lower bound of 1, NRES cannot be applied, so the implicit lower bound of our algorithm would be 0. Nevertheless, it is important to observe that, if we encode this problem as a weighted CSP and enforce a stronger form of consistency called existential directional arc consistency (EDAC*)[de Givry *et al.*, 2005], we transform the formula into the equivalent $\{(\square,1),(x \vee y,1)\}$ which also has

an implicit lower bound of 1. The performance of enforcing EDAC* on random Max-SAT instances is also illustrated in Figure 1. As it can be seen, it clearly provides further improvement over the application of the NRES rule. Experiments on Max-SAT using weighted CSP technology can be performed using the freely available *toolbar* solver.[2]

7 Conclusions and Future work

Motivated by the success of [de Givry *et al.*, 2003] in solving Max-SAT instances as weighted CSPs, we have studied the interpretation of WCSP local consistency properties within the Max-SAT context. The result of our work is a logical framework for Max-SAT in which the solving process can be seen as a set of transformation rules. Interestingly, our approach leads to a natural extension of the Davis-Putnam algorithm as well as an extension of the resolution rule (RES). The application of a limited form of RES, called neighborhood resolution (NRES), provides an interesting and effective algorithm. We have shown the relation between the application of NRES and some local consistency properties in weighted CSP. Finally, we have also shown that NRES provides (implicit) lower bounds not subsumed by state-of-the-art Max-SAT solvers.

We believe our work leaves several directions of future work. First, it would be interesting to analyze under which conditions the application of RES provides a complete method for Max-SAT. Since it has been shown that *directional resolution* is just an instantiation of *adaptive consistency* [Rish and Dechter, 2000], it would also be interesting to know if there is any relation between the application of RES and the more general *bucket elimination* [Dechter, 1999].

Our framework facilitates the relation between Max-SAT and weighted CSP solving. The resolution rule that we have introduced seems only valid to enforce node and arc consistency. However, it is known in the WCSP context that stronger forms of local consistency such as EDAC* may be more effective in practice [de Givry *et al.*, 2005]. Thus, it seems natural to search for new transformation rules such that, when added to NRES, can achieve higher levels of local consistency.

References

[Alsinet *et al.*, 2003] T. Alsinet, F. Manya, and J. Planas. Improved branch and bound algorithms for max-sat. In *Proc. of the 6th SAT*, pages 408–415, 2003.

[Bacchus, 2002] Fahiem Bacchus. Enhancing davis putnam with extended binary clause reasoning. In *Proceedings of the 18th AAAI*, pages 613–619, 2002.

[Bistarelli *et al.*, 1999] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison. *Constraints*, 4:199–240, 1999.

[Cha and Iwama, 1996] Byungki Cha and Kazuo Iwama. Adding new clauses for faster local search. In *Proc. of the 13thAAAI*, pages 332–337, Portland, OR, 1996.

[Cooper, 2003] M. Cooper. Reductions operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems*, 134(3):311–342, 2003.

[Davis *et al.*, 1962] M. Davis, G. Logemann, and G. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[de Givry *et al.*, 2003] S. de Givry, J. Larrosa, P. Meseguer, and T. Schiex. Solving max-sat as weighted csp. In *Proc. of the 9th CP*, pages 363–376, Kinsale, Ireland, 2003. LNCS 2833. Springer Verlag.

[de Givry *et al.*, 2005] S. de Givry, F. Heras, J. Larrosa, and M. Zytnicki. Existential arc consistency: getting closer to full arc consistency in weighted csps. In *Proc. of the 19th IJCAI*, Edinburgh, U.K., August 2005.

[de Givry, 2004] S. de Givry. Singleton consistency and dominance testing for weighted csp. In *Proc. of the 6th Intl. Workshop on Soft Constraints and Preferences*, pages 363–376, Toronto, Canada, 2004.

[Dechter, 1999] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[Drake *et al.*, 2002] L. Drake, A. Frisch, and T. Walsh. Adding resolution to the dpll procedure for boolean satisfiability. In *Proceedings of 5th SAT*, pages 122–129, 2002.

[Gelder, 1995] A. Van Gelder. *Satisfiability testing with more reasoning and less guessing*, pages 0–1. American Mathematical Society, 1995.

[Hansen and Jaumard, 1990] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279–303, 1990.

[Larrosa and Schiex, 2003] J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted csp. In *Proc. of the 18th IJCAI*, Acapulco, Mexico, August 2003.

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

[Rish and Dechter, 2000] I. Rish and R. Dechter. Resolution vs. sat: two approaches to sat. *Journal of Automated Reasoning*, 24(1):225–275, 2000.

[Robinson, 1965] J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[Shen and Zhang, 2004] Haiou Shen and Hantao Zhang. Study of lower bounds for max-2-sat. In *Proceedings of the 19th AAAI*, 2004.

[Xing and Zhang, 2004] Zhao Xing and Weixiong Zhang. Efficient strategies for (weighted) maximum satisfiability. In *Proc. of the 10th CP*, pages 690–705, Toronto, Canada, 2004. LNCS 3258. Springer Verlag.

[2] http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro