A New Algorithm for Weighted Partial MaxSAT[*]

Carlos Ansótegui
DIEI, UdL
Jaume II 69, Lleida, Spain

Maria Luisa Bonet
LSI, UPC
J. Girona 1–3, Barcelona, Spain

Jordi Levy
IIIA, CSIC
Campus UAB, Bellaterra, Spain

Abstract

We present and implement a Weighted Partial MaxSAT solver based on successive calls to a SAT solver. We prove the correctness of our algorithm and compare our solver with other Weighted Partial MaxSAT solvers.

Introduction

The Weighted Partial MaxSAT problem is a generalization of the satisfiability problem. The idea is that sometimes not all restrictions of a problem can be satisfied, so we divide the instance in two groups: the restrictions or clauses that must be satisfied (hard), and the ones that may or may not be satisfied (soft). In the last group, we may put different weights to the clauses, where the weight is the penalty to falsify the clause. The idea is that not all restrictions are equally important. The addition of weights to clauses makes the instance weighted, and the separation into hard and soft clauses makes the instance partial. Given a weighted partial MaxSAT instance, we want to find the assignment that satisfies the hard clauses, and the sum of the weights of the falsified clauses is minimal. Such an assignment will be optimal in this context.

The weighted partial MaxSAT problem is a natural combinatorial problem, and it can be used in several domains, such as: combinatorial auctions, scheduling and timetabling problems, FPGA routing, software package installation, etc. However, state-of-the-art solvers have not yet experienced the same success as SAT solvers for the Satisfiability problem in the industrial field. Weighted Partial MaxSAT is an NP-hard problem, so our aim is to produce efficient solvers to handle real/industrial problems that although generally big in size, are not as hard as the worst case instances.

A straightforward strategy to solve a Weighted Partial MaxSAT problem is to translate it to a Partial MaxSAT problem where every soft clause C with weight w has been replaced by w copies of C, and then use a Partial MaxSAT solver. However, the weights can be arbitrary large and therefore the size of the encoding prohibitive for any solver. Therefore this approach can only be used in instances with relatively small weights. Instead, in the SAT community there are mainly two kinds of solvers that can deal directly with weights: depth-first branch and bound solvers WMaxSatz (Li et al. 2009), MiniMaxSat (Heras, Larrosa, and Oliveras 2007), IncWMaxSatz (Lin, Su, and Li 2008), and solvers based on satisfiability testing: SAT4J (Berre), WBO and Msuncore (Manquinho, Marques-Silva, and Planes 2009) and WPM1 (Ansotegui, Bonet, and Levy 2009). The latest essentially make use of successive calls to a SAT solver, and have become very competitive for the industrial categories in the MaxSAT evaluation (Argelich et al. 2008). In general, the branch and bound solvers seem to be more competitive on random problems, while solvers based on calls to a SAT solver seem to be better for industrial or real problems. This will be the approach in our work.

To the best of our knowledge the solver we present here, WPM2, is the first weighted version of the Partial MaxSAT solver PM2 (Ansotegui, Bonet, and Levy 2009), that won the industrial category of partial MaxSAT instances at the MaxSAT evaluation. It is based on successive calls to a sat solver and it is able to exploit the information provided by the unsatisfiable cores to guide the next call. Also we use just one auxiliary variable per soft clause and we do not duplicate clauses unlike the solvers WPM1 (Ansotegui, Bonet, and Levy 2009) and WBO, Msuncore (Manquinho, Marques-Silva, and Planes 2009).

We tested and compared our solver using the industrial instances available from the MaxSat Evaluation 2009 and from the TimeTabling Competition 2007. Our solver is always competitive or better for the hardest instances than other solvers based on calls to a SAT solver, and is always better than branch and bound solvers.

Preliminaries

A *weighted clause* is a pair (C, w), where C is a clause and w is a natural number or infinity meaning the penalty for falsifying the clause C. A clause is called *hard* if the corresponding weight is infinity, otherwise the clause is called *soft*. A *weighted partial maxSAT formula* is a multiset of weighted clauses $\varphi = \{(C_1, w_1), \dots, (C_m, w_m), (C_{m+1}, \infty), \dots, (C_{m+m'}, \infty)\}$, where the first m clauses are soft and the last m' clauses are hard. The pair (C, w) is clearly equivalent to having w

[*]Research supported by the projects MULOG2 TIN2007-68005-C04-{01,03,04} and ARINF (TIN2009-14704-C03-01/02).
Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

copies of clause C in our multiset (in case C is soft).

Given a formula φ and a truth assignment I, the *cost* of the assignment I on φ, noted $\text{cost}(\varphi, I)$, is the sum of the weights of the clauses falsified by I.

The *Weighted Partial MaxSAT problem* for a multiset of weighted clauses φ is the problem of finding an *optimal assignment* to the variables of φ that minimizes the cost of the assignment on φ. If the cost is infinity, it means that we must falsify a hard clause, and we say that the multiset is *unsatisfiable*.

The *Weighted MaxSAT problem* is the Weighted Partial MaxSAT problem when there are no hard clauses. The *Partial MaxSAT problem* is the Weighted Partial MaxSAT problem when the weights of the soft clauses are equal to one. The *MaxSAT problem* is the Partial MaxSAT problem when there are not hard clauses. The *SAT problem* is the Partial MaxSAT problem when there are not soft clauses.

SAT Based Weighted MaxSAT

In this section we describe the SAT based approach in general. The detailed description of our algorithm appears in the next section.

We can solve a Weighted Partial MaxSAT problem φ through the resolution of a sequence of SAT instances as follows. Let φ^k be a SAT formula that is satisfiable iff φ has an assignment with cost k. If the cost of the optimal assignment to φ is k_{opt}, the generated SAT problems φ^k for $k \geq k_{opt}$ are satisfiable, while for $k < k_{opt}$ are unsatisfiable. Therefore, the precise location of this transition between satisfiable and unsatisfiable corresponds to the search of the cost of the optimal assignment to φ. Moreover, the set of all satisfying assignments of $\varphi^{k_{opt}}$ is the set of optimal assignments of φ.

One way to encode φ^k is to extend every soft clause C_i with a fresh auxiliary variable b_i, and add the conversion to CNF of the *linear pseudo-Boolean constraint* $\sum_{i=1}^{m} w_i\, b_i = k$. Then $\varphi^k = \{C_1 \vee b_1, \ldots, C_m \vee b_m, C_{m+1}, \ldots, C_{m+m'}\} \cup \text{CNF}(\sum_{i=1}^{m} w_i\, b_i = k)$. Notice that k may range from 0 to $\sum_{i=1}^{m} w_i$ (the sum of the weights of the soft clauses). The search for the value k_{opt} can be done following different strategies; searching from $k = 0$ to k_{opt} (increasing k while φ^k is unsatisfiable); from $k = \sum_{i=1}^{m} w_i$ to some value smaller than k_{opt} (decreasing k while φ^k is satisfiable); or alternating unsatisfiable and satisfiable φ^k until the algorithm converges to k_{opt}. The key point to boost the efficiency of these approaches is to know whether we can exploit any additional information from the execution of the SAT solver for the next runs.

The approach used by the solver SAT4J (Berre) is to exploit the information of the satisfiable formulas φ^k. It starts with $k = \sum_{i=1}^{m} w_i$ and decreases this value until the SAT solver reports unsatisfiable. The value of k in the penultimate step corresponds to the optimal cost. Whenever the underlying SAT solver returns satisfiable it checks the satisfying assignment and sets the next k equal to the sum of the weights of the soft clauses with auxiliary variable set to true. In this case, the *linear pseudo-Boolean constraint* is of the form $\sum_{i=1}^{m} w_i\, b_i < k$.

The approach used by (Ansotegui, Bonet, and Levy 2009; Manquinho, Marques-Silva, and Planes 2009) is to exploit the information of the unsatisfiable formulas φ^k. It is the extension of the Fu and Malik algorithm (Fu 2007; Fu and Malik 2006)[1], described originally only for Partial MaxSAT, to the Weighted Partial MaxSAT problem. This algorithm starts with $k = 0$ and increases this value until φ^k is satisfiable. Whenever φ^k is unsatisfiable, the SAT solver also returns an unsatisfiable core. It is analyzed for adding auxiliary variables to the soft clauses belonging to the core, and the next φ^k is constructed also by adding *cardinality constraints* on this variables, stating that exactly one of them has to be true. This prevents the solver to find the same unsatisfiable core in the next iteration. The value of k is updated adding the minimum weight of the soft clauses involved in the core. This approach is quite effective since it allows to solve more efficiently the unsatisfiable φ^k instances, due to the addition of cardinality constraints at each iteration.

However, this approach has two weak points. First, a soft clause can be extended with more than one auxiliary variable (if it belongs to more than one core). This can hamper the efficiency of the SAT solver. Second, whenever an unsatisfiable core is found, every involved soft clause is replaced by two copies: one extended with an additional auxiliary variable, and weight equal to the minimum weight of the core; and an unextended one with weight equal to the original weight minus the minimum weight of the core. This duplication of the soft clauses increase the size of the working formula.

Our work presents an algorithm for solving the Weighted MaxSAT problem based on satisfiability testing that uses just one auxiliary variable for every soft clause, and that does not require to duplicate them. It is inspired on the solver PM2, described originally for the Partial MaxSAT problem.

The WPM2 Algorithm

The WPM2 algorithm computes the optimal cost K_{opt} of a Weighted Partial MaxSAT instance φ. It is based on unsatisfiable calls to a SAT solver, where in every iteration it works with a SAT formula φ^K that is satisfiable iff φ has an assignment with cost K. Starting from $K = 0$, to $K = K_{opt}$, it increases K while the SAT solver reports unsatisfiable.

Like in PM2 (Ansotegui, Bonet, and Levy 2009), we extend every soft clause C_i with a unique fresh auxiliary blocking variable b_i. The algorithm works with a set AL of at-least linear pseudo-boolean constraints on the variables b_i, and a similar set AM of at-most constraints, that are modified at every iteration of the algorithm. The formula φ^K sent to the SAT solver is $\varphi^K = \varphi^e \cup \text{CNF}(AL \cup AM)$, where $\varphi^e = \{C_1 \vee b_1, \ldots, C_m \vee b_m, C_{m+1}, \ldots, C_{m+m'}\}$, and the linear constraints have been encoded as CNF formulas. In order to understand the purpose of AL and AM, we introduce the notion of partial solution.

Definition 1 (Partial Solution and Cost) *An assignment* $I : \{b_1, \ldots, b_m\} \rightarrow \{0, 1\}$ *is called a partial solution*

[1]The first known implementation of the Fu and Malik algorithm for Partial MaxSAT is the solver msu1.2 (Marques-Silva and Manquinho 2008; Marques-Silva and Planes 2007).

if $I(\varphi^e) = \{C_i \mid I(b_i) = 0\} \cup \{C_{m+1}, \dots, C_{m+m'}\}$ is satisfiable.

The cost of a partial solution $I : \{b_1, \dots, b_m\} \to \{0,1\}$ is $cost(I) = \sum_{i=1}^m w_i\, I(b_i)$.

The at-least constraints AL exclude assignments to b_i's that are not partial solutions. The at-most constraints AM enforce that all solutions of the set of constraints $AL \cup AM$ are the solutions of AL of minimal cost, if AL has any solution. This ensures that any solution of $\varphi^e \cup \mathrm{CNF}(AL \cup AM)$, if there is any, is a solution of φ^e of minimal cost, hence a MaxSAT solution of φ.

The pseudo-code of WPM2 is described in Figure 1. The algorithm starts with $AL = \emptyset$ and the corresponding $AM := \{w_1\, b_1 \le 0, \dots, w_m\, b_m \le 0\}$ that ensures that the unique solution of $AL \cup AM$ is $b_1 = \dots = b_m = 0$ with cost 0.

At every iteration, the algorithm calls a SAT solver with $\varphi^e \cup \mathrm{CNF}(AL \cup AM)$. If it returns satisfiable, then the assignment is a Max-SAT solution of φ. If it returns unsatisfiable, then we use the information of the unsatisfiable core obtained by the SAT solver to enlarge the set AL, excluding more interpretations on the b_i's that are not partial solutions. We update AM conveniently, to ensure that solutions to the new constraints $AL \cup AM$ are still minimal solutions of the new AL constraint set. Notice that AM depends on AL, and it is modified accordingly when AL is extended. Moreover, in every iteration, the set of solutions of $\{b_1, \dots, b_m\}$ defined by AL is decreased, whereas the set of solutions of AM is increased. The constraints of AL are of the form $\sum_{i\in B} w_i\, b_i \ge k$. Therefore, they are characterized by a set of indexes $B \subseteq \{1, \dots, m\}$, called its *base*, and a value $k \in \mathbb{N}$, called its *bound*. Similarly, the inequalities of AM are of the form $\sum_{i\in B} w_i\, b_i \le k$.

Before describing the algorithm in detail, it is convenient to introduce the notion of cover.

Definition 2 (Cover) *Given a set of cores L, where any $A \in L$, $A \subseteq \{1, \dots, m\}$, its set of covers, noted $\mathrm{SC}(L)$, is defined as the minimal partition of $\{1, \dots, m\}$ such that for every $A \in L$ and $B \in \mathrm{SC}(L)$, if $A \cap B \neq \emptyset$, then $A \subseteq B$.*

At every iteration of the algorithm, the bases of the at-most constraints AM are the set of covers SC. Moreover, the cost K is equal to the sum of the bounds of the at-most constraints, $K = \sum \big\{k \bigm| (\sum_{i\in B} w_i\, b_i \le k) \in AM\big\}$.

When an unsatisfiable core φ_c is found by the SAT solver, we compute the set $A \subseteq \{1, \dots, m\}$ of indexes of soft clauses contained in φ_c. We also call this set a *core*. Then, we compute the set of covers that intersect with A, i.e. $RC = \{B' \in SC \mid B' \cap A \neq \emptyset\}$, and their union $B = \bigcup_{B'\in RC} B'$. The new set of covers is $SC = SC \setminus RC \cup \{B\}$. The set of at-least constraints AL is augmented with a new constraint $\sum_{i\in B} w_i\, b_i \ge \mathrm{newbound}(AL, B)$. The set AM is updated conveniently by removing all constraints of the form $\sum_{i\in B} w_i\, b_i \le k$ where $B \in RC$, and adding $\sum_{i\in B} w_i\, b_i \le \mathrm{newbound}(AL, B)$. The key point is how to calculate this value $\mathrm{newbound}(AL, B)$, which must be greater that the sum of the bounds of the removed at-most constraints. In fact its is the minimal value greater than this sum and ensuring that the new $AL \cup AM$ has some solution. Later, we will describe how to compute this new bound in practice. In the following example we show the execution of our algorithm and describe some problems that arise when trying to compute the new bound.

Example 3 Suppose that we have the following MaxSAT formula $\varphi = \big\{(C_1, 10),\ (C_2, 4),\ (C_3, 8),\ (C_4, 2)\big\}$. The extended formula is $\varphi^e = \{(C_1 \vee b_1, C_2 \vee b_2, C_3 \vee b_3, C_4 \vee b_4\}$. Suppose that we get the cores $A_1 = \{1, 2\}$ and later $A_2 = \{3, 4\}$. The constraints are $AL = \{10\, b_1 + 4\, b_2 \ge 4,\ 8\, b_3 + 2\, b_4 \ge 2\}$ and $AM = \{10\, b_1 + 4\, b_2 \le 4,\ 8\, b_3 + 2\, b_4 \le 2\}$. And the set of covers is $SC = \big\{\{1, 2\}, \{3, 4\}\big\}$.

Now, suppose that, in the third iteration, the SAT solver gives us the core $A_3 = \{2, 3\}$. Notice that, eventhough b_2 must be true, this core A_3 may be obtained because we assume that the SAT solver does not have to return minimal cores. A_3 intersects with the two old covers, therefore we will have a unique new cover $B = \{1, 2, 3, 4\}$. Notice that from AL, and the existence of a new core, we can infer $10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4 > 6$, where 6 is the sum of the two bounds in the removed at-most constraints. However, AL has no model I satisfying $I(10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4) = 7$. Therefore, we can improve the inequality by finding the smallest value $k > 6$ such that there exist a model I of the new $AL \cup \{10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4 \ge k\}$. This value is $k = 12$, which we obtain setting $I(b_1) = I(b_4) = 0$ and $I(b_2) = I(b_3) = 1$.

Correctness of the WPM2 Algorithm

Next we state some basic properties of covers.

Recall that, since $\mathrm{SC}(L)$ is a partition of $\{1, \dots, m\}$, for any $i = 1, \dots, m$, there exists one and only one set in $\mathrm{SC}(L)$ containing i, and for any $A \in L$, there exists one and only one set in $\mathrm{SC}(L)$ containing A.

Lemma 4 *For any two sets of cores L_1 and L_2, if $L_1 \subseteq L_2$ then any cover $B \in \mathrm{SC}(L_2)$ satisfies $B = \bigcup_{\substack{A\in \mathrm{SC}(L_1)\\ A\subseteq B}} A$.*

PROOF: Notice that, if $L_1 \subseteq L_2$, then for any $A \in \mathrm{SC}(L_1)$ there exists one and only one set $B \in \mathrm{SC}(L_2)$ such that $A \subseteq B$. Therefore, for any $A \in \mathrm{SC}(L_1)$ and any $B \in \mathrm{SC}(L_2)$, if $B \cap A \neq \emptyset$ then $A \subseteq B$, and the lemma follows. ■

Definition 5 (Bound) *For any set of at-least constraints AL, and set $B \subseteq \{1, \dots, m\}$, we define its* bound *as*

$$\mathrm{bound}(AL, B) = \max\Big\{k \in \mathbb{N} \Bigm| AL \vdash \sum_{i\in B} w_i\, b_i \ge k\Big\}$$

where $AL \vdash C$ means that all assignments to b_i's satisfying the constraints AL also satisfy the constraint C.

For instance, for the set at-least constraints AL of Example 3, we have $\mathrm{bound}(AL, \{1, 2, 3, 4\}) = 6$.

The following lemma basically rephrases the definition of bound in a more convenient way.

Lemma 6 *For any set of constraints AL, and set $B \subseteq \{1, \dots, m\}$, $\mathrm{bound}(AL, B) = k$ iff*

1. *the bound is probable: $AL \vdash \sum_{i\in B} w_i\, b_i \ge k$, and*
2. *feasible: $AL \not\vdash \sum_{i\in B} w_i\, b_i > k$, i.e. exists an interpretation $I : \{b_1, \dots, b_m\} \to \{0, 1\}$ such that $I(AL) = true$ and $I(AL \vdash \sum_{i\in B} w_i\, b_i = k) = true$.*

```
input: $\varphi = \{(C_1, w_1), \ldots, (C_m, w_m), (C_{m+1}, \infty), \ldots, (C_{m+m'}, \infty)\}$
$\varphi^e := \{C_1 \vee b_1, \ldots, C_m \vee b_m, C_{m+1}, \ldots, C_{m+m'}\}$          Protect all soft clauses
$SC := \{\{1\}, \{2\}, \ldots, \{m\}\}$          Set of Covers
$AL := \emptyset$          Set of at least constraints
$AM := \{w_1\, b_1 \leq 0, \ldots, w_m\, b_m \leq 0\}$          Set of at most constraints
while $true$ do
    $(st, \varphi_c) := SAT(\varphi^e \cup \mathrm{CNF}(AL \cup AM))$          Call to the SAT solver
    if $st = SAT$ then return $\sum\{k' \mid \sum_{i\in B'} w_i\, b_i \leq k' \in AM\}$
    remove the hard clauses from $\varphi_c$
    if $\varphi_c = \emptyset$ then return UNSAT
    $A := \emptyset$          Blocking variables of the core
    for each $C = C_i \vee b_i \in \varphi_c$ do
        $A := A \cup \{i\}$
    $RC := \{B \in SC \mid B \cap A \neq \emptyset\}$          Covers to be removed
    $B := \bigcup_{B' \in RC} B'$          New cover
    $k := \mathrm{newbound}(AL, B)$          New bound
    $SC := SC \setminus RC \cup \{B\}$          New set of Covers
    $AL := AL \cup \{\sum_{i\in B} w_i\, b_i \geq k\}$          Add new at-least cardinality constraint
    $AM := AM \setminus \{\sum_{i\in B'} w_i\, b_i \leq k' \mid B' \in RC\} \cup \{\sum_{i\in B} w_i\, b_i \leq k\}$          Actualize at-most constraints
```

Figure 1: The pseudo-code of the WPM2 algorithm. The code of function *newbound* is in a forthcoming section.

As we said in the previous section, the set of at-most constraints AM depends on the set AL.

Definition 7 *Given a set of at-least constraints AL, the set of at-most constraints AM associated to AL is*

$$AM = \Big\{ \sum_{i\in B} w_i\, b_i \leq \mathrm{bound}(AL, B) \;\Big|\; B \in \mathrm{SC}(L) \Big\}$$

where $L = \big\{A \;\big|\; (\sum_{i\in A} w_i\, b_i \geq k) \in AL\big\}$.

Lemma 8 *For any set of at-least constraints AL and any subset of covers $\{B_1, \cdots, B_s\} \subseteq \mathrm{SC}(L)$, where L is the set of bases of AL, we have*

$$\mathrm{bound}(AL, \bigcup_{i=1}^{s} B_i) = \sum_{i=1}^{s} \mathrm{bound}(AL, B_i)$$

Hence, the bound of $\bigcup_{i=1}^{s} B_i$ is the sum of the bounds of the constraints $\sum_{j\in B_i} w_j\, b_j \leq k_i \in AM$ associated to AL.

PROOF: The inequality $\mathrm{bound}(AL, \bigcup_{i=1}^{s} B_i) \geq \sum_{i=1}^{s} \mathrm{bound}(AL, B_i)$ is trivial, and holds for any set $\{B_1, \ldots, B_s\}$ of pairwise disjoint sets, even if they are not covers. However the opposite inequality is more difficult to prove.

On one hand, by Lemma 6, we have $AL \vdash \sum_{j\in B_i} w_j\, b_j \geq \mathrm{bound}(AL, B_i)$. Since B_i's are pairwise disjoint, adding these inequalities we get $AL \vdash \sum_{i=1}^{s}\sum_{j\in B_i} w_j\, b_j = \sum_{j\in \bigcup_{i=1}^{s} B_i} w_j\, b_j \geq \sum_{i=1}^{s} \mathrm{bound}(AL, B_i)$.

On the other hand, again by Lemma 6, for every $i = 1, \ldots s$ we have an interpretation I_i such that $I_i(AL) = true$ and $I_i(\sum_{j\in B_i} w_j\, b_j \leq \mathrm{bound}(AL, B_i)) = true$. Now, construct a new interpretation I defined as $I(x) = I_i(x)$, if $x \in B_i$; and $I(x) = I_1(x)$, if $x \notin \bigcup_{i=1}^{s} B_i$. Notice that all constraints of AL have all their variables inside one of the sets B_i. Therefore, $I(AL) = true$. Notice also that $I(\sum_{j\in B_i} w_j\, b_j) = I_i(\sum_{j\in B_i} w_j\, b_j) = \mathrm{bound}(AL, B_i)$. Therefore, $I(\sum_{i=1}^{s}\sum_{j\in B_i} w_j\, b_j) = \sum_{i=1}^{s} \mathrm{bound}(AL, B_i)$.

These two facts, by Lemma 6, prove that statement of the Lemma. ■

The following lemma allows us to strengthen the set of at-least constraints. For any set of indexes B, we always have $\varphi \vdash \sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL, B)$. The lemma states that, whenever we find a core A, we can improve this inequality for B being the union of covers intersecting with A, i.e. the cover of $\mathrm{SC}(\{A_1, \ldots, A_r, A\})$ that contains A. Therefore, we can enlarge AL getting $AL' = AL \cup \{\sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL, B) + 1)\}$. However, it could be the case that AL' had no models satisfying $\sum_{i\in B} w_i\, b_i = \mathrm{bound}(AL, B) + 1$ (see Example 3). In this case, we enlarge AL with $\sum_{i\in B} w_i\, b_i = \mathrm{newbound}(AL, B)$, where $\mathrm{newbound}(AL, B)$ is the minimum integer greater than $\mathrm{bound}(AL, B) + 1$ such that AL' has a model with satisfying $\sum_{i\in B} w_i\, b_i = \mathrm{newbound}(AL, B)$.

Lemma 9 *Let $\varphi^e = \{C_1 \vee b_1, \ldots, C_m \vee b_m, C_{m+1}, \ldots, C_{m+m'}\}$ be an extended formula and AL be a set of at-least constraints. Let AM be the set of at-most constraints associated to AL.*

If, for some core $A \subseteq \{1, \ldots, m\}$, we have

1. *$\{C_i \vee b_i\}_{i\in A} \cup \{C_{m+1}, \ldots, C_{m+m'}\} \cup AL \cup AM$ is unsatisfiable*
2. *$\varphi^e \vdash AL$*

then

$$\varphi^e \vdash \sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL, B) + 1$$

where $B \in \mathrm{SC}(L \cup \{A\})$, L is the set of bases of AL, and $A \subseteq B$.

Moreover,
$$\varphi^e \vdash \sum_{i\in B} w_i\, b_i \geq \mathrm{newbound}(AL, B)$$
where
$$\mathrm{newbound}(AL,B) = \mathrm{bound}(AL',\ B)$$
$$AL' = AL \cup \big\{ \textstyle\sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL,B)+1 \big\}$$

PROOF: Suppose that $\varphi^e \not\vdash \sum_{i\in A} w_i\, b_i \geq \mathrm{bound}(AL,B)+1$. This means that there exists an assignment I such that $I(\varphi^e) = true$ and $I\big(\sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL,B)+1\big) = false$. We have $I\big(\sum_{i\in B} w_i\, b_i \leq \mathrm{bound}(AL,B)\big) = true$. Since $\varphi^e \vdash AL$, $I(AL) = true$. By definition of bound, we have
$$I\Big(\sum_{i\in B} w_i\, b_i = \mathrm{bound}(AL,B)\Big) = true \qquad (1)$$
Since $I(AL) = true$, by definition of bound, we have
$$I\Big(\sum_{i\in B'} w_i\, b_i \geq \mathrm{bound}(AL,B')\Big) = true \qquad (2)$$
for any $B' \in \mathrm{SC}(L)$. On the other side, since $B \in \mathrm{SC}(L \cup \{A\})$, by Lemmas 4 and 8
$$B = \bigcup_{\substack{B'\in \mathrm{SC}(L)\\ B'\subseteq B}} B'$$
$$\mathrm{bound}(AL,B) = \sum_{\substack{B'\in \mathrm{SC}(L)\\ B'\subseteq B}} \mathrm{bound}(AL,B') \qquad (3)$$
From (1), (2) and (3) we have
$$I\Big(\sum_{i\in B'} w_i\, b_i \leq \mathrm{bound}(AL,B')\Big) = true$$
for any $B' \in \mathrm{SC}(L)$ satisfying $B' \subseteq B$. Hence, I satisfies all the at-most constraints of AM that makes reference to the covers B' included in B. We have to modify the interpretation I to get an interpretation that also satisfies the constraints of AM that make reference to covers not included in B.

By definition of bound, there exists an interpretation I' such that $I'(AL) = true$ and $I'(AM) = true$. Let I'' be another interpretation defined as $I''(x) = I(x)$ if x is a variables of $\{C_1,\dots,C_{m+m'}\}$ or $\{b_i\}_{i\in B}$, and $I''(x) = I'(x)$, otherwise. Notice that all clauses of $AL \cup AM \cup \{C_i \vee b_i\}_{i\in A} \cup \{C_{m+1},\dots,C_{m+m'}\}$ either contain variables of one subset or from the other. Therefore, I'' satisfies all these clauses. This contradicts the first assumption of the lemma. ■

Example 10 Suppose that we have the situation described in Example 3. The list of cores is $L = \{\{1,2\},\ \{3,4\}\}$, and the constraints
$$AL = \{10\, b_1 + 4\, b_2 \geq 4,\ 8\, b_3 + 2\, b_4 \geq 2\}$$
$$AM = \{10\, b_1 + 4\, b_2 \leq 4,\ 8\, b_3 + 2\, b_4 \leq 2\}$$
The third core $A = \{2,3\}$ generates a new cover $B = \{1,2,3,4\}$ that belongs to the set of covers of $\{\{1,2\},\ \{3,4\},\ \{2,3\}\}$. We have $\mathrm{bound}(AL,\{1,2,3,4\}) = 6$. The first part of Lemma 9 allows us to conclude that $\varphi^e \vdash 10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4 \geq 7$. After adding this constraint we get $AL' = AL \cup \{10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4 \geq 7\}$, using the second part of the lemma, we get $\mathrm{bound}(AL',\{1,2,3,4\}) = \mathrm{newbound}(AL,\{1,2,3,4\}) = 12$ and $\varphi^e \vdash 10\, b_1 + 4\, b_2 + 8\, b_3 + 2\, b_4 \geq 12$.

Theorem 11 *Given a Weighted Partial MaxSAT problem, the algorithm WPM2 computes an optimal assignment of minimal cost.*

PROOF: Once the algorithm finds a satisfying assignment I for $\varphi^e \cup \mathrm{CNF}(AL \cup AM)$, it returns $K = \sum\{k' \mid \sum_{i\in B'} w_i\, b_i \leq k' \in AM\}$. Since the bases of at-most constraints are a partition of $\{1,\dots,m\}$ and I satisfies AM, the cost of I is bounded by K. By Lemma 8, $K = \mathrm{bound}(AL,\{1,\dots,m\})$, and by Lemma 6, $AL \vdash \sum_{i=1}^{m} w_i\, b_i \geq K$. Lemma 9 ensures that the new at-least constraint added to AL in each iteration does not exclude partial solutions: $\varphi^e \vdash AL$ i.e. it only excludes values of the b_i's such thatφ^e is unsatisfiable. Therefore, the cost K of I is smaller than the cost of any partial solution of φ^e, hence of the minimal cost of φ. Termination of the algorithm is ensured by the fact that the value of $\sum\{k' \mid \sum_{i\in B'} w_i\, b_i \leq k' \in AM\}$ is increased in every iteration, and it is bounded by the minimal cost of φ. ■

Computation of the New Bound

The value of $\mathrm{bound}(AL,B)$ can be computed easily using the at-most constraints as
$$\mathrm{bound}(AL,B) = \sum\{k' \mid \sum_{i\in B'} w_i\, b_i \leq k' \in AM \wedge B' \subseteq B\}$$

However, the computation of $\mathrm{newbound}(AL,B)$ is not so simple. Recall the definition:
$$\mathrm{newbound}(AL,B) = \mathrm{bound}(AL',\ B)$$
$$\text{where } AL' = AL \cup \big\{ \textstyle\sum_{i\in B} w_i\, b_i \geq \mathrm{bound}(AL,B)+1 \big\}$$

Given AL and B, the calculation of $\mathrm{newbound}(AL,B)$ is an NP-complete optimization problem. This can be seen by a reduction from the following version of the subset sum problem: given $\{w_1,\dots,w_n\}$ and k, minimize $\sum_{j=1}^{n} w_j\, x_j$ subject to $\sum_{j=1}^{n} w_j\, x_j > k$ and $x_j \in \{0,1\}$. This is equivalent to computing $\mathrm{newbound}(AL,B)$, where the weights are w_j, $B = \{1,\dots,n\}$ and $AL = \{\sum_{j=1}^{n} w_j\, x_j \geq k\}$.

In our implementation we use the following function to compute newbound.

```
function newbound($AL, B$)
$k := \mathrm{bound}(AL, B)$
repeat
    $k = subsetsum(\{w_i \mid i \in B\}, k)$
until $SAT(\mathrm{CNF}(AL \cup \{\sum_{i\in B} w_i\, b_i = k\}))$
return $k$
```

Notice that the satisfiability check of $AL \cup \{\sum_{i\in B} w_i\, b_i = k\}$ is necessary for soundness (see Example 10). Also notice that the subset sum problem, even though it is NP-complete, in practical situations can be computed very efficiently. As many other numerical problems, it is pseudo-polynomial. Our algorithm could

MaxSat Evaluation 2009 (median time in seconds)									
Weighted Partial MaxSAT (Industrial)									
set	#	WMSz	IncWMSz	MiniM	SAT4J	MCore	WBO	WPM1	WPM2
10	20	1963	10	22	21	15	26	25	47
20	20	2977	11	22	22	17	29	28	52
30	20	-	13	22	22	18	32	31	57
40	20	-	14	22	22	19	35	33	61
TimeTabling Competition 2007 (time in seconds)									
instance		WMSz	IncWMSz	MiniM	SAT4J	MCore	WBO	WPM1	WPM2
comp04		-	-	-	-	-	43	-	108
comp08		-	-	-	-	-	-	-	156
comp10		-	-	-	-	92	-	95	108
comp13		-	-	-	-	-	-	-	1333
comp14		-	-	-	-	-	-	-	244
comp16		-	-	-	-	1041	-	-	114
comp19		-	-	-	-	-	-	-	142
comp20		-	-	-	-	690	-	-	-

Table 1: Timeout of 1 hour. # stands for number of instances of the benchmark. '-' means that either the solver run out of memory or the timeout expired.

be implemented on top of a Pseudo-Boolean solver with support for core extraction to circumvent the size issue of translation to SAT.

Experimental Results

In order to know if our approach is promising, we implemented the WPM2 algorithm and compared it with other Weighted Partial MaxSat Solvers on Weighted Partial MaxSAT instances from the MaxSAT09 evaluation (Argelich et al. 2008) and the International TimeTabling Competition (ITC 2007)[2].

The WPM2 solver is built on top of the PM2 solver (Ansotegui, Bonet, and Levy 2009). The PM2 solver is implemented on top of the SAT solver picosat846 (Biere 2008). In order to translate into SAT the at-least and at-most linear pseudo-Boolean constraints that WPM2 generates, we used part of the code of minisat+ (Eén and Sörensson 2006) as a translator.

The Weighted Partial MaxSAT solvers we have compared ours with are: MiniMaxSat (MiniM) (Heras, Larrosa, and Oliveras 2007), Msuncore (MCore) (Manquinho, Marques-Silva, and Planes 2009), SAT4J (Berre), WBO (Manquinho, Marques-Silva, and Planes 2009), WPM1 (Ansotegui, Bonet, and Levy 2009), WPM2, WMaxSatz (WMSz) (Li et al. 2009) and IncWMaxSatz (IncWMSz) (Lin, Su, and Li 2008).

Our experiments have been run on machines with the following specifications. Operating System: Rocks Cluster 4.0.0 Linux 2.6.9. Processor: AMD Opteron 248 Processor, 2 GHz. Memory: 0.5 GB. Compiler: GCC 3.4.3.

Table 1 shows the results of the experimental investigation. We discarded those instances which are satisfiable. We used a cutoff of 1 hour.

[2]The Translation to Weighted Partial MaxSAT instances was provided by Roberto Asin from the BarceLogic Team at the UPC University: `http://www.lsi.upc.edu/~rasin/timetabling.html`.

For the instances at the MaxSAT evaluation we show the median time of each set. As we can see these instances are solvable by all solvers. WPM2 performs worse than the other solvers based on satisfiability testing. Our solver makes system calls to minisat+ to translate the Pseudo-Boolean constraints into SAT and interchanges the data through files. This causes an overhead in the computation time that partly explains why our solver performs worse on these instances. In particular, as in the case of these instances, this is important when the solver needs to perform many iterations where each call to the SAT solver has a low cost.

For the instances from the TimeTabling Competition 2007 (track 3) we show the running time in seconds. There are a total of 20 instances. We only show results on those instances that were solved by at least one solver. These instances are much harder, and as we can see now WPM2 is clearly the best performing solver.

Since the weights of the timetabling instances are small, we also solve them with PM2 with duplication of weighted clauses. We find that WPM2 is faster than PM2 with duplication.

References

Ansotegui, C.; Bonet, M. L.; and Levy, J. 2009. Solving (weighted) partial maxsat through satisfiability testing. In *SAT'09*, 427–440.

Argelich, J.; Li, C. M.; Manyà, F.; and Planes, J. 2008. The first and second Max-SAT evaluations. *JSAT* 4:251–278.

Berre, D. L. Sat4jmaxsat. In *www.sat4j.org*.

Biere, A. 2008. PicoSAT essentials. *JSAT* 4:75–97.

Eén, N., and Sörensson, N. 2006. Translating pseudo-boolean constraints into SAT. *JSAT* 2(1-4):1–26.

Fu, Z., and Malik, S. 2006. On solving the partial max-sat problem. In *SAT'06*, 252–265.

Fu, Z. 2007. *Extending the Power of Boolean Satisfiability: Techniques and Applications*. Ph.D. Dissertation, Princeton.

Heras, F.; Larrosa, J.; and Oliveras, A. 2007. MiniMaxSat: A new weighted Max-SAT solver. In *SAT'07*, 41–55.

ITC. 2007. Second international timetabling competition. In *http://www.cs.qub.ac.uk/itc2007/*.

Li, C. M.; Manyà, F.; Mohamedou, N. O.; and Planes, J. 2009. Exploiting cycle structures in Max-SAT. In *SAT'09*.

Lin, H.; Su, K.; and Li, C. M. 2008. Within-problem learning for efficient lower bound computation in Max-SAT solving. In *AAAI'08*, 351–356.

Manquinho, V.; Marques-Silva, J.; and Planes, J. 2009. Algorithms for weighted boolean optimization. In *SAT'09*, 495–508.

Marques-Silva, J., and Manquinho, V. 2008. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *SAT'08*, 225–230.

Marques-Silva, J., and Planes, J. 2007. On using unsatisfiability for solving maximum satisfiability. *CoRR* abs/0712.1097.