# Improving Unsatisfiability-based Algorithms for Boolean Optimization

Vasco Manquinho, Ruben Martins, and Inês Lynce

IST/INESC-ID, Technical University of Lisbon, Portugal
{vmm,ruben,ines}@sat.inesc-id.pt

**Abstract.** Recently, several unsatisfiability-based algorithms have been proposed for Maximum Satisfiability (MaxSAT) and other Boolean Optimization problems. These algorithms are based on being able to iteratively identify and relax unsatisfiable sub-formulas with the use of fast Boolean satisfiability solvers. It has been shown that this approach is very effective for several classes of instances, but it can perform poorly on others for which classical Boolean optimization algorithms find it easy to solve. This paper proposes the use of Pseudo-Boolean Optimization (PBO) solvers as a preprocessor for unsatisfiability-based algorithms in order to increase its robustness. Moreover, the use of constraint branching, a well-known technique from Integer Linear Programming, is also proposed into the unsatisfiability-based framework. Experimental results show that the integration of these features in an unsatisfiability-based algorithm results in an improved and more effective solver for Boolean optimization problems.

## 1 Introduction

The success of Propositional Satisfiability (SAT) solvers has increased the interest in several generalizations of SAT, namely in Boolean optimization problems. As a result, several techniques first proposed for SAT algorithms have been extended for Pseudo-Boolean Optimization (PBO), Maximum Satisfiability (MaxSAT) and the more general problem of Weighted Boolean Optimization (WBO). Moreover, the acknowledgment of the strong relation between all these problems has led to the development of new algorithms based on the translation between these Boolean formalisms [8, 12, 15].

Algorithms based on the identification of unsatisfiable sub-formulas have also been developed and are now able to tackle all these Boolean optimization problems. The first proposal of unsatisfiability-based algorithm [13] was restricted to MaxSAT and partial MaxSAT problems. However, recent work has been done on improving this algorithmic solution [21] and generalizing it for weighted MaxSAT, PBO and WBO [2, 19].

The proposal in this paper is for a further integration of procedures in an unique Boolean optimization framework. Hence, it is proposed the encoding into PBO and the use of a PBO solver as a preprocessing step for finding a tight upper bound on the optimal solution before applying an unsatisfiability-based algorithm. Moreover, the use of constraint branching, a well-known technique initially presented for (Mixed) Integer Linear Programming, can also be integrated with success in an unsatisfiability-based algorithm for Boolean optimization.

The paper is organized as follows: in section 2 several formalisms used for Boolean optimization are introduced, namely Weighted Boolean Optimization (WBO), pseudo-Boolean Optimization (PBO) and the Maximum Satisfiability (MaxSAT) problem and its variants. Furthermore, several relations between these formalisms are reviewed, as well as the most common algorithmic solutions. In section 3, it is proposed the use of pseudo-Boolean solvers as a preprocessing step for an unsatisfiability-based algorithm. Next, in section 4, it is shown how to integrate constraint branching into an unsatisfiability-based solver for WBO. Finally, experimental results are presented in section 5 and the paper concludes in section 6.

## 2 Preliminaries

In this section several Boolean optimization problems are defined, starting with the more general Weighted Boolean Optimization problem. Next, translations between several formalisms are reviewed and the most common algorithmic solutions are briefly described. The approach based on the identification of unsatisfiable sub-formulas is presented in more detail since it will be extensively referred in the remaining of the paper.

### 2.1 Weighted Boolean Optimization

Weighted Boolean Optimization (WBO) is a natural extension of other Boolean problems, such as Maximum Satisfiability (MaxSAT) and Pseudo-Boolean Optimization (PBO). In WBO, constraints can be any linear inequality with integer coefficients (also known as pseudo-Boolean constraints) defined over a set of Boolean variables. In general, one can define a pseudo-Boolean constraint as follows:

$$\sum_{j \in N} a_j \; l_j \geq b \tag{1}$$

where $a_j$ and $b$ are positive integers and $l_j$ is a propositional literal that either denotes a variable $x_j$ or its complement $\bar{x}_j$. It is well-known that all other types of linear constraints with Boolean variables can be easily translated into this one [7]. Notice that propositional clauses are a particular case of pseudo-Boolean constraints where all coefficients $a_j$ and the right-hand side $b$ are equal to 1. If all $a_j$ are equal to 1 and $b > 1$, then the constraint is called a cardinality constraint.

A WBO formula $\varphi$ is defined as the conjunction of two pseudo-Boolean formulas $\varphi_h$ and $\varphi_s$, where $\varphi_h$ contains the *hard* constraints and $\varphi_s$ contains the *soft* constraints. Moreover, each soft constraint $\omega_i$ has an associated positive weight $c_i$ that represents the cost of not satisfying constraint $\omega_i$. The WBO problem can be defined as finding an assignment to problem variables that satisfies all hard constraints in $\varphi_h$ and minimizes the total weight of unsatisfied soft constraints in $\varphi_s$.

*Example 1.* Consider the following example of a WBO formula:

$$\begin{array}{l} \varphi_h = \{\; x_1 + x_2 + x_3 \geq 2, \quad 2\bar{x}_1 + \bar{x}_2 + x_3 \geq 2\} \\ \varphi_s = \{\; (x_1 + \bar{x}_2 \geq 1, 2), \quad (\bar{x}_1 + \bar{x}_3 \geq 1, 3)\} \end{array} \tag{2}$$

In this example, there are only two possible assignments that satisfy all hard constraints in $\varphi_h$. These assignments are $x_1 = x_3 = 1, x_2 = 0$ and $x_1 = 0, x_2 = x_3 = 1$. However, for each of these assignments, at least one soft constraint in $\varphi_s$ is made unsatisfied. Therefore, the solution to the WBO instance would be $x_1 = 0, x_2 = x_3 = 1$ since it is the assignment that minimizes the total cost of unsatisfied soft constraints while satisfying all hard constraints.

### 2.2 Relating with MaxSAT and Pseudo-Boolean Optimization

One should note that WBO is a direct generalization of the Maximum Satisfiability (MaxSAT) problem and variants. The MaxSAT problem can be defined as finding an assignment that minimizes the number of unsatisfied clauses in a given CNF formula $\varphi$. Hence, a WBO instance where $\varphi_h = \emptyset$ and $\varphi_s$ contains only propositional clauses with weight 1 is in fact a MaxSAT instance.

The *partial* MaxSAT problem differs from MaxSAT since there is a set of clauses declared as hard and a set of clauses declared as soft. The objective in partial MaxSAT is to find an assignment such that all hard clauses are satisfied while minimizing the number of unsatisfied soft clauses. Again, a WBO formula where all constraints in $\varphi_h$ and $\varphi_s$ are propositional clauses and all soft clauses have weight 1 is a partial MaxSAT instance. Finally, there are also variants of MaxSAT and partial MaxSAT with weights greater than 1 which are respectively known as *weighted* MaxSAT and *partial weighted* MaxSAT. Clearly, the resulting instances are also specific cases of WBO instances.

Another well-known Boolean optimization formalism is Pseudo-Boolean Optimization (PBO), also known as 0-1 Integer Linear Programming (0-1 ILP). The PBO problem can be defined as finding an assignment to the Boolean variables such that a set of pseudo-Boolean constraints is satisfied and the value of a linear cost function is minimized. Formally, it is possible to define PBO as follows:

$$\begin{aligned} &\text{minimize} \quad \sum_{j \in N} c_j \, x_j \\ &\text{subject to} \quad \sum_{j \in N} a_{ij} \, l_j \geq b_i, \\ &\qquad\qquad l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i, c_j \in \mathbf{N}_0^+ \end{aligned} \tag{3}$$

It is also possible to encode a PBO instance into the WBO formalism. The constraint set of the PBO instance can be directly mapped into the set of hard constraints $\varphi_h$ of the resulting WBO instance, while the objective function is mapped using soft constraints. For each term $c_j x_j$ in the objective function, a new soft constraint $\bar{x}_j \geq 1$ is added to $\varphi_s$ with weight $c_j$. The optimal solution to the resulting WBO instance will also be an optimal solution to the original PBO instance [19].

### 2.3 Algorithmic Solutions

For each of these Boolean formalisms (WBO, MaxSAT and PBO), there is a wide variety of algorithmic solutions. One classical approach is the use of a branch and bound algorithm where an *upper bound* on the value of the objective function is updated whenever a better solution is found and *lower bounds* on the value of the objective function

**Algorithm 1** Unsatisfiability-based Algorithm for MaxSAT and partial MaxSAT

FUMALIKALG($\varphi$)

```
1  while true
2      do (st, φ_C) ← SAT(φ)
3         if st = UNSAT
4           then V_R ← ∅
5                for each ω ∈ φ_C ∧ soft(ω)
6                    do r is a new relaxation variable
7                       ω_R ← ω ∪ {r}
8                       φ ← φ\{ω} ∪ {ω_R}
9                       V_R ← V_R ∪ {r}
10               if V_R = ∅
11                 then return UNSAT
12                 else φ ← φ ∪ CNF(∑_{r∈V_R} r = 1)
13                   ▷ Additional clauses for Equals1 constraint are marked as hard clauses
14          else return Satisfiable assignment to φ
```

are estimated considering a set of variable assignments. Whenever the lower bound value is higher or equal to the upper bound, the search procedure can safely backtrack since it is guaranteed that the current best solution cannot be improved by extending the current set of variable assignments. Several MaxSAT and PBO algorithms follow this approach using different lower bounding procedures [16, 17, 3, 14, 18].

Another approach used in PBO solvers is to perform a linear search on the value of the objective function by iterating on the possible upper bound values [7]. Whenever a new solution to the problem constraints is found, the upper bound value is updated and a new constraint is added such that all solutions with an higher value are discarded. Several state of the art PBO solvers use this approach such as `Pueblo` [26], `minisat+` [12], among others [8, 1]. These solvers rely on the generalization of the most effective techniques already used in SAT solvers, such as Boolean Constraint Propagation, conflict-based learning and conflict-directed backtracking [18, 10].

There are other successful solvers that perform conversions of one Boolean formalism to another and subsequently use a specific solver on the new formalism. For instance, PBO solver `minisat+` [12] converts all pseudo-Boolean constraints to propositional clauses and uses a SAT solver to find an assignment that satisfies the problem constraints; `SAT4J MAXSAT` [8] converts MaxSAT instances into a PBO instance; `Toolbar` [15] converts MaxSAT instances into a weighted constraint network and uses a Constraint Satisfaction Problem (CSP) solver, among other solvers [23, 24].

A recent approach initially proposed by Fu and Malik [13] for MaxSAT and partial MaxSAT problems is based on the iterated use of a SAT solver to identify unsatisfiable sub-formulas. Algorithm 1 presents the pseudo-code for the original Fu and Malik's proposal. Consider that $\varphi$ is the Boolean working formula where constraints are marked as either soft or hard. At each iteration, a SAT solver is used and its output is a pair $(\text{st}, \varphi_C)$ where st denotes the resulting status of the solver (satisfiable or unsat-

**Algorithm 2** Unsatisfiability-based Weighted Boolean Optimization algorithm

```
WBO(φ)
 1  while true
 2      do (st, φ_C) ← PB(φ)
 3         if st = UNSAT
 4           then min_c ← ∞
 5                for each ω ∈ φ_C
 6                    do if soft(ω) and cost(ω) < min_c
 7                          then min_c ← cost(ω)
 8                V_R ← ∅
 9                for each ω ∈ φ_C ∧ soft(ω)
10                    do r is a new relaxation variable and ω = Σ a_j l_j ≥ b
11                       V_R ← V_R ∪ {r}
12                       ω_R ← (b r + Σ a_j l_j ≥ b)
13                       cost(ω_R) ← min_c
14                       if cost(ω) > min_c
15                         then φ ← φ ∪ {ω_R}
16                              cost(ω) ← cost(ω) − min_c
17                         else φ ← φ\{ω} ∪ {ω_R}
18                if V_R = ∅
19                  then return UNSAT
20                  else φ_W ← φ_W ∪ {Σ_{r∈V_R} r = 1}
21           else return Satisfiable assignment to φ
```

isfiable) and $\varphi_C$ contains the unsatisfiable sub-formula provided by the SAT solver if $\varphi$ is unsatisfiable. In this latter case, for each soft constraint in $\varphi_C$, a new relaxation variable is added. Moreover, $\varphi$ is changed to encode that exactly one of the new relaxation variables can be assigned value 1 (Equals1 constraint in line 12) and the algorithm continues to the next iteration. Otherwise, if $\varphi$ is satisfiable, the SAT solver was able to find an assignment which is an optimal solution to the original MaxSAT or partial MaxSAT problem [13].

Different algorithms have been proposed for MaxSAT and partial MaxSAT based on this approach. For instance, effective encodings for the Equals1 constraint have been proposed with better results [21, 20] than the pairwise encoding of the original algorithm [13]. Moreover, different strategies have been used regarding the total number of relaxation variables needed [20, 2].

Finally, the unsatisfiability-based approach has also been extended for weighted and partial weighted MaxSAT [2, 19] and generalized to WBO [19] by using a pseudo-Boolean solver instead of a SAT solver. Algorithm 2 presents the pseudo-code for the WBO solver and one can clearly notice that it follows the same structure as Algorithm 1. However, in this case, $\varphi$ is now a WBO formula, i.e. constraints can be any type of pseudo-Boolean constraints and a positive cost is associated with each soft constraint. One difference from Algorithm 1 is in lines 4-7 where $min_c$ denotes the cost associated to the unsatisfiable sub-formula $\varphi_C$, defined as the minimum cost of soft constraints

in $\varphi_C$. Moreover, if the weight of a soft constraint in $\varphi_C$ is larger than $min_c$, then the relaxation also differs since the original constraint is kept, but with a smaller weight as shown in lines 9-17. Finally, notice that the Equals1 constraint in line 20 does not need to be encoded into CNF, since a pseudo-Boolean solver is used instead of a SAT solver.

## 3 Improving Unsatisfiability-based Algorithms

As shown previously, unsatisfiability-based algorithms are able to tackle several Boolean optimization problems. These algorithms work by making a linear search on the lower bounds of the optimal solution value. However, it has been shown that in some cases, it is preferable to search on the upper bounds of the optimal solution.

In this section, we propose to translate Weighted Boolean Optimization (WBO) to the more specific Pseudo-Boolean Optimization (PBO) problem before applying an unsatisfiability-based algorithm. This approach has two main goals: (i) to apply simplification techniques that are used as preprocessing procedures in PBO and (ii) to find a tight upper bound on the optimal solution. Afterwards, the problem is again translated into WBO and solved using an unsatisfiability-based algorithm.

### 3.1 Pseudo-Boolean Optimization as Preprocessing

We start by reviewing the translation from WBO formulas into PBO. Clearly, hard constraints $\varphi_h$ can be directly mapped as constraints into the resulting PBO formula. However, for soft constraints in $\varphi_s$, additional variables are needed. Each soft constraint of the form $\sum a_j\ l_j \geq b$, is mapped into a new PBO constraint $b\ r + \sum a_j l_j \geq b$, where $r$ is a new relaxation variable. The objective function will be to minimize the weighted sum of the relaxation variables. The coefficient of variable $r$ in the objective function is the weight of the original constraint associated with variable $r$.

*Example 2.* Consider the following WBO formula:

$$\begin{array}{l} \varphi_h = \{\ x_1 + x_2 + x_3 \geq 2, \quad 2\bar{x}_1 + \bar{x}_2 + x_3 \geq 2, \quad x_1 + x_4 \geq 1\} \\ \varphi_s = \{\ (x_1 + \bar{x}_2 \geq 1, 2), \quad (\bar{x}_1 + \bar{x}_3 \geq 1, 3), \quad (\bar{x}_4 \geq 1, 4)\} \end{array} \tag{4}$$

The resulting PBO instance would be:

$$\begin{array}{ll} \text{minimize} & 2r_1 + 3r_2 + 4r_3 \\ \text{subject to} & x_1 + x_2 + x_3 \geq 2 \\ & 2\bar{x}_1 + \bar{x}_2 + x_3 \geq 2 \\ & x_1 + x_4 \geq 1 \\ & r_1 + x_1 + \bar{x}_2 \geq 1 \\ & r_2 + \bar{x}_1 + \bar{x}_3 \geq 1 \\ & r_3 + \bar{x}_4 \geq 1 \end{array} \tag{5}$$

Notice that in this example variable $r_3$ is not necessary in the resulting PBO instance. Since $\bar{x}_4 \geq 1$ is a unit clause in (4), one can remove this constraint and just add $x_4$ with weight 4 to the objective function, which would result in minimizing $2r_1 + 3r_2 + 4x_4$.

This is an important simplification, as many industrial instances have unit clauses as soft constraints [4].

After translating the WBO formula to PBO, two steps are applied:

1. Simplification techniques are used in the PBO formula;
2. The PBO formula is solved using tight limits.

In the first step we use a generalization of Hypre [5] for pseudo-Boolean formulas. As a result, literal equivalence detection and hyper-binary resolution are used to eliminate variables from the formula. In fact, besides these techniques, other preprocessing procedures could have also been used, such as clause and variable subsumption, among others [22].

After the first step, a search procedure is carried out using a pseudo-Boolean solver and making a linear search on the upper bound of the optimal solution. However, our use of the pseudo-Boolean solver is limited to 10% of the time limit given to solve the formula. Notice that the goal is not to solve the problem using a PBO solver, but rather to quickly find a tight upper bound on the optimal solution such that it prunes the search space on the upper bound side.

Given a tight time limit, the pseudo-Boolean solver will not find the optimal solution in most cases. Therefore, if the solver is unable to prove optimality, the problem instance is encoded back to WBO and solved using an unsatisfiability-based algorithm. Remember that the unsatisfiability-based algorithm will make a search on the lower bound of the optimal solutions, but in this case it will be already limited on the upper bound side. We note that searching on both the upper and the lower bound on the value of the objective function is not new [21], but to the best of our knowledge, the presented approach is novel.

Although the objective is to find a tight upper bound, it is possible that the PBO solver proves the optimality of the found upper bound. In that case, the optimal solution to the original problem has been found without having to make the search on the lower bound value. However, even if the solver is unable to prove optimality, small clauses learned by the pseudo-Boolean solver are kept in the WBO formula as hard clauses, further constraining the search space.

## 4 Using Constraint Branching

One of the main problems of using unsatisfiability-based algorithms for WBO is that after a given number of iterations, the number of relaxation variables can be much larger than the initial number of problem variables [21]. This might occur even when using a pseudo-Boolean solver where the encoding of the Equals1 constraint to CNF is not necessary. Furthermore, when solving a formula with several Equals1 constraints, setting a single variable to 0 or 1 may cause a dramatic difference on the number of propagations that results from this assignment.

Remember that in each iteration of Algorithm 2, a new Equals1 constraint is added (line 20), thus constraining that only one of the new relaxation variables can be assigned

value 1. Consider that, at any given iteration, $k$ new relaxation variables are added. As a result, a new Equals1 constraint is added as follows:

$$\sum_{i=1}^{k} r_i = 1 \tag{6}$$

Notice that, by assigning one variable $r_i$ with value 1, all other variables $r_j \neq r_i$ (with $1 \leq j \leq k$) must be assigned value 0. However, if $r_i$ is assigned value 0, no propagation occurs due to (6). As a result, assigning a value to any of these variables tends to produce very different search trees, in particular for large values of $k$. Therefore, if the solver assigns one single variable that appears in these constraints early in the search tree, that assignment might be too strong or too weak depending on the chosen value. This problem has already been observed in (Mixed) Integer Linear Programming problems [6] and one way to balance the search tree is to use constraint branching [25].

Constraint branching is a well-known technique used in specific cases of (Mixed) Integer Linear Programming in which the formula to be solved is split into two subproblems such that new constraints are added to each branch. In our case, we would like to take advantage of the Equals1 constraints in order to assign large sets (hundreads or even thousands) of variables in a single step. Therefore, it is proposed the use of a branching step due on Equals1 constraints and integrate it into an unsatisfiability-based algorithm.

By using constraint branching on an Equals1 constraint, instead of assigning just one variable, half of the $k$ variables in (6) are assigned value 0. Without loss of generality, assume that variables $r_1$ to $r_{k/2}$ are assigned value 0. This is done by adding the following constraint to the working formula $\varphi$:

$$\omega_{c1} : \sum_{i=1}^{k/2} r_i = 0 \tag{7}$$

This means that the variable to be assigned value 1 is one of $r_{k/2+1}$ to $r_k$. If the formula $\varphi \cup \{\omega_{c1}\}$ is not satisfiable, then it is possible to infer that one of the variables from $r_1$ to $r_{k/2}$ must be assigned value 1, while all others from $r_{k/2+1}$ to $r_k$ must be assigned value 0. Hence, if $\varphi \cup \{\omega_{c1}\}$ is unsatisfiable, the following constraint can be safely inferred:

$$\omega_{c2} : \sum_{i=k/2+1}^{k} r_i = 0 \tag{8}$$

Algorithm 3 illustrates the use of constraint branching in the computation of unsatisfiable sub-formulas. This procedure can replace the call for the pseudo-Boolean solver in line 2 of Algorithm 2. In Algorithm 3 we start by selecting a *large*[1] Equals1 constraint in order to maximize the number of variables to be assigned due to $\omega_{c1}$. Notice that $\varphi_{C1}$ denotes an unsatisfiable sub-formula from $\varphi \cup \{\omega_{c1}\}$. If $\varphi_{C1}$ does not include $\omega_{c1}$, then $\varphi_{C1}$ is also an unsatisfiable sub-formula from $\varphi$ and the procedure

[1] An Equals1 constraint with than 100 relaxation variables is considered large in the context of our solver.

**Algorithm 3** Using Constraint Branching in Unsatisfiability-based WBO Algorithm

```
COMPUTE_CORE(φ)
 1  ▷ Compute an unsatisfiable sub-formula from φ
 2  if (no large Equals1 constraint exist in φ)
 3     then (st, φ_C) ← PB(φ)
 4          return (st, φ_C)
 5     else Select a large Equals1 constraint ω from φ
 6          k = size(ω)
 7          ω_c1 : Σ_{i=1}^{k/2} r_i = 0
 8          (st, φ_C1) ← COMPUTE_CORE(φ ∪ {ω_c1})
 9          if (st = SAT ∨ ω_c1 ∉ φ_C1)
10            then return (st, φ_C1)
11            else ω_c2 : Σ_{i=k/2+1}^{k} r_i = 0
12                 (st, φ_C2) ← COMPUTE_CORE(φ ∪ {ω_c2})
13                 if (st = SAT ∨ ω_c2 ∉ φ_C2)
14                   then return (st, φ_C2)
15                   else return (st, φ_C1 ∪ φ_C2)
```

returns. The same applies to $\varphi_{C2}$ when it does not include $\omega_{c2}$. Otherwise, if both $\varphi_{C1}$ and $\varphi_{C2}$ include the respective added constraints, then an unsatisfiable sub-formula for $\varphi$ is $\varphi_{C1} \cup \varphi_{C2}$.

Finally, it should be noticed that, in practice, this technique is applied parsimoniously. It was observed that if we were to make constraint branching on *all* large Equals1 constraints, then the resulting unsatisfiable sub-formula would usually be much larger than a single call to the pseudo-Boolean solver. This occurs, since the search space is explored differently in each sub-problem and the set union of both unsatisfiable sub-formulas results in a larger unsatisfiable sub-formula for the main problem. Hence, before making a constraint branching step, the solver is called with a limited number of conflicts (approx. 30,000). Afterwards, if the solver has been unable to produce an unsatisfiable sub-formula, a constraint branching step is applied.

## 5 Experimental Results

In order to evaluate the techniques proposed in the paper, solver `wbo` was modified to include pseudo-Boolean optimization techniques described in section 3, as well as the use of constraint branching, presented in section 4. The new version of solver `wbo` is 1.2, while version 1.0 is the one submitted to the last MaxSAT evaluation [4].

For the experimental evaluation, the industrial benchmark sets of the partial MaxSAT problem (a specific case of WBO) were selected. Besides `wbo`, we also run other solvers among the most effective for these benchmark sets, namely `MSUncore` [21, 19], `SAT4J MaxSAT` [8] and `pm2` [2]. Experiments were run on a set of Intel Xeon 5160 servers (3.0GHZ, 1333Mhz, 3GB) running Red Hat Enterprise Linux WS 4. For each instance, the CPU time limit was 1800 seconds.

**Table 1.** Solved Instances for Industrial Partial MaxSAT

| Benchmark set | #I | MSUncore | SAT4J (MS) | pm2 | wbo1.0 | wbo1.2 |
|---|---|---|---|---|---|---|
| bcp-fir | 59 | 49 | 10 | 58 | 40 | 47 |
| bcp-hipp-yRa1 | 176 | 139 | 140 | 166 | 144 | 137 |
| bcp-msp | 148 | 121 | 95 | 93 | 26 | 95 |
| bcp-mtg | 215 | 173 | 196 | 215 | 181 | 207 |
| bcp-syn | 74 | 32 | 21 | 39 | 34 | 33 |
| CircuitTraceCompaction | 4 | 0 | 4 | 4 | 0 | 4 |
| HaplotypeAssembly | 6 | 5 | 0 | 5 | 5 | 5 |
| pbo-mqc | 256 | 119 | 250 | 217 | 131 | 210 |
| pbo-routing | 15 | 15 | 13 | 15 | 15 | 15 |
| PROTEIN_INS | 12 | 0 | 2 | 3 | 1 | 2 |
| Total | 965 | 553 | 731 | 815 | 577 | 755 |

Table 1 shows the number of solved instances by each solver for all benchmark sets. The improvements from version 1.0 to version 1.2 of `wbo` are clear. The overall number of solved instances is vastly improved as it now solves more instances than `MSUncore` and `SAT4J MaxSAT`. Nevertheless, version 1.2 of `wbo` is not as effective as the current version of `pm2`. However, `wbo` has an additional overhead since it is a more general solver able to tackle any WBO problem instance, whereas `pm2` is specific for partial MaxSAT and it cannot handle formulas with weights. Furthermore, `pm2` needs to use the encoding of cardinality constraints to CNF that depends on the number of iterations, but since the number of iterations for most instances is not large, the respective CNF encoding should tend to produce manageable CNF formulas. Finally, `wbo` is built on top of `minisat 2.0` [11], while `pm2` is built on the more effective `PicoSAT` solver [9].

The improvements of `wbo` are due to different reasons for the several benchmark sets. For example, improvements in `bcp-fir` are due to the use of constraint branching technique, while in `bcp-msp` several instances are trivially solved by the use of a PBO solver at preprocessing. Preprocessing techniques from PBO are also extensively applied in the `CircuitTraceCompaction` where the initial formula can be significantly reduced. Overall, the integration of all these techniques into an unsatisfiability-based algorithm improve its performance and robustness for several sets of industrial instances.

Observe that it was choosen not to present results from other industrial categories from the MaxSAT evaluation for two main reasons: (i) version 1.0 of the `wbo` solver was already able to solve all instances from the partial weighted MaxSAT problem and (ii) the proposed techniques do not apply on the industrial MaxSAT instances without hard constraints for which `wbo` was already one of the best performing solvers [4]. Note that PBO preprocessing techniques can only be applied when literal implications can be extracted from the formula and that does not occur for those benchmark sets. Furthermore, most of industrial MaxSAT instances are solved after finding a single unsatisfiable sub-formula. Hence, there are not enough iterations to apply constraint branching and overall results from version 1.0 and 1.2 for solver `wbo` would be the same for these sets of instances.

## 6 Conclusions

This paper proposes to extend an unsatisfiability-based algorithm for Weighted Boolean Optimization, by first encoding the problem into pseudo-Boolean Optimization such that powerful inference preprocessing techniques can be used. Furthermore, the pseudo-Boolean solver can also be used to learn hard constraints and deal with problem instances that are trivially solved using a linear search on the upper bound value of the solution. Moreover, the paper also shows how to selectively apply constraint branching in the unsatisfiability-based framework.

Preliminary experimental results show that these techniques significantly improve the performance of our unsatisfiability-based algorithm when solving industrial instances of the partial MaxSAT problem (a special case of Weighted Boolean Optimization). As a result, our solver is now competitive with dedicated algorithms for the partial MaxSAT problem.

The success obtained on solving these problem instances with the integration of techniques from Pseudo-Boolean Optimization and constraint branching, first proposed for (Mixed) Integer Linear Programming, provide a strong stimulus for further integration of several Boolean optimization techniques into an unique framework.

**Acknowledgement.** This work was supported by FCT grant PTDC/EIA/76572/2006.

## References

1. F. Aloul, A. Ramani, I. Markov, and K. A. Sakallah. Generic ILP versus specialized 0-1 ILP: An update. In *International Conference on Computer-Aided Design*, pages 450–457, 2002.
2. C. Ansótegui, M. Bonet, and J. Levy. Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 427–440, 2009.
3. J. Argelich, C. M. Li, and F. Manyà. An improved exact solver for partial max-sat. In *Proceedings of the International Conference on Nonconvex Programming: Local and Global Approaches (NCP-2007)*, pages 230–231, 2007.
4. J. Argelich, C. M. Li, F. Manyà, and J. Planes. Fourth Max-SAT evaluation. www.maxsat.udl.cat/09/, 2009.
5. F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction. In *Sixth International Conference on Theory and Applications of Satisfiability Testing*, pages 183–192, 2003.
6. C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.
7. P. Barth. A Davis-Putnam Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Technical Report MPI-I-95-2-003, Max Plank Institute for Computer Science, 1995.
8. D. L. Berre. SAT4J library. www.sat4j.org.
9. A. Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:75–97, 2008.
10. D. Chai and A. Kuehlmann. A fast pseudo-Boolean constraint solver. In *Design Automation Conference*, pages 830–835, 2003.
11. N. Eén and N. Sörensson. Minisat 2.0 sat solver. http://minisat.se/MiniSat.html.

12. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2, 2006.
13. Z. Fu and S. Malik. On solving the partial MAX-SAT problem. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 252–265, August 2006.
14. F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.
15. J. Larrosa, F. Heras, and S. de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2-3):204–233, 2008.
16. C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.
17. H. Lin and K. Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *International Joint Conference on Artificial Intelligence*, pages 2334–2339, 2007.
18. V. Manquinho and J. Marques-Silva. Search pruning techniques in SAT-based branch-and-bound algorithms for the binate covering problem. *IEEE Transactions on Computer-Aided Design*, 21(5):505–516, 2002.
19. V. Manquinho, J. Marques-Silva, and J. Planes. Algorithms for weighted boolean optimization. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 495–508, 2009.
20. J. Marques-Silva and V. Manquinho. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 225–230, 2008.
21. J. Marques-Silva and J. Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Design, Automation and Testing in Europe Conference*, pages 408–413, March 2008.
22. R. Martins, I. Lynce, and V. Manquinho. Preprocessing in pseudo-boolean optimization: An experimental evaluation. In *Eighth International Workshop on Constraint Modelling and Reformulation*, 2009.
23. K. Pipatsrisawat, A. Palyan, M. Chavira, A. Choi, and A. Darwiche. Solving weighted Max-SAT problems in a reduced search space: A performance analysis. *Journal on Satisfiability Boolean Modeling and Computation*, 4:191–217, 2008.
24. M. Ramírez and H. Geffner. Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In C. Bessiere, editor, *The 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 605–619. Springer, 2007.
25. D. Ryan and B. Foster. An integer programming approach to scheduling. In *Computer Scheduling of Public Transport*, pages 269–280, 1981.
26. H. Sheini and K. Sakallah. Pueblo: A Modern Pseudo-Boolean SAT Solver. In *Design, Automation and Testing in Europe Conference*, pages 684–685, March 2005.