Model-based Partitioning for MaxSAT Solving

Ruben Martins, Vasco Manquinho, and Inês Lynce

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

Abstract. Linear search algorithms have been shown to be particularly effective for solving partial Maximum Satisfiability (MaxSAT) problem instances. These algorithms start by adding a new relaxation variable to each soft clause and solving the resulting formula with a SAT solver. Whenever a model is found, a new constraint on the relaxation variables is added such that models with a greater or equal value are excluded. However, if the problem instance has a large number of relaxation variables, then adding a new constraint over these variables can lead to the exploration of a much larger search space.
This paper proposes new algorithms that use the models found by the SAT solver to partition the relaxation variables. These algorithms add a new constraint on a subset of relaxation variables, thus intensifying the search on that subspace. Preliminary results show that model-based algorithms can outperform a traditional linear search algorithm in several problem instances.

1 Introduction

Linear search algorithms for MaxSAT have shown to be effective for solving several classes of MaxSAT problems. These algorithms work by iteratively finding models of a relaxed MaxSAT formula, such that the number of unsatisfied soft clauses of the original MaxSAT formula is minimized. At each SAT call, a cardinality constraint over all relaxation variables is added to the working formula so that it excludes models with a greater or equal value. However, if the number of soft clauses is large, then the number of relaxation variables will be also large. Adding a cardinality constraint over a large number of relaxation variables can lead to the exploration of a much larger search space.

This paper describes new algorithms that use the models found by the SAT solver to iteratively increase the set of relaxation variables that are used in the cardinality constraint. By using only a small subset of relaxation variables, we are able to intensify the search on a smaller search space. These model-based algorithms are expected to improve the performance of linear search algorithms for MaxSAT problem instances having a large number of soft clauses.

Proceedings of the 20$^{\text{th}}$ RCRA workshop on *Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion* (RCRA 2013).
Rome, Italy, June 14–15, 2013.

The paper is organized as follows. In the next section the MaxSAT problem is defined and linear search algorithms are described. Section 3 describes the proposed model-based algorithms for MaxSAT. Afterwards, section 4 presents an experimental evaluation of the proposed algorithms. Finally, the paper concludes and suggests future work.

2 Preliminaries

A Boolean formula in conjunctive normal form (CNF) is defined as a conjunction ($\wedge$) of clauses, where a clause is a disjunction ($\vee$) of literals and a literal is a Boolean variable x or its negation $\bar{x}$. A Boolean variable may be assigned truth values 1 (true) or 0 (false). A positive (negative) literal x ($\bar{x}$) is said to be satisfied if the respective variable is assigned value true (false). A positive (negative) literal x ($\bar{x}$) is said to be unsatisfied if the respective variable is assigned value false (true). A clause is said to be satisfied if at least one of its literals is satisfied. A clause is said to be unsatisfied if all of its literals are unsatisfied. A formula is satisfied if all of its clauses are satisfied. The Boolean Satisfiability (SAT) problem is to decide whether there exists an assignment that makes the formula satisfied. Such assignment is called a solution or model.

The Maximum Satisfiability (MaxSAT) problem is an optimization version of the SAT problem which consists in finding an assignment that minimizes (maximizes) the number of unsatisfied (satisfied) clauses. In the remainder of the paper, it is assumed that MaxSAT is defined as a minimization problem.

MaxSAT has several variants such as partial MaxSAT, weighted MaxSAT and weighted partial MaxSAT. In the partial MaxSAT problem, some clauses are declared as hard, while the rest are declared as soft. The objective in partial MaxSAT is to find an assignment to the problem variables such that all hard clauses are satisfied, while minimizing the number of unsatisfied soft clauses. Finally, in the weighted versions of MaxSAT, soft clauses can have weights greater than 1 and the objective is to satisfy all hard clauses while minimizing the total weight of unsatisfied soft clauses. For simplicity, in the remainder of the paper we will consider a formula as being a multiset of clauses.

Example 1. Consider the partial MaxSAT formula φ such that $\varphi = \varphi_h \cup \varphi_s$, where φ_h denotes the set of hard clauses and φ_s the set of soft clauses. Furthermore, consider the following example:

$$\begin{aligned} \varphi_h &= \{(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4)\} \\ \varphi_s &= \{(x_1), (x_2 \vee \bar{x}_1), (x_3), (\bar{x}_3 \vee x_1), (x_4)\} \end{aligned} \tag{1}$$

An optimal solution would be $x_1 = x_2 = x_3 = 0, x_4 = 1$. This assignment satisfies all hard clauses and only two soft clauses are unsatisfied.

A generalization of clauses are cardinality constraints. These constraints define that a sum of n literals must be smaller than or equal to a given value k, i.e. $\sum_{n=1}^{k} x_i \leq k$. In other words, a cardinality constraint over n literals ensures that

Algorithm 1: Linear search algorithm for partial MaxSAT

Input: $\varphi = \varphi_h \cup \varphi_s$
Output: satisfiable assignment to φ or UNSAT

```
1  (V_R, model, μ, φ_W) ← (∅, ∅, +∞, φ_h)
2  foreach ω ∈ φ_s do
3      V_R ← V_R ∪ {r}                                  // r is a new variable
4      ω_R ← ω ∪ {r}                                    // relax soft clause
5      φ_W ← φ_W ∪ ω_R
6  while true do
7      (st, ν, φ_C) ← SAT(φ_W)
8      if st = SAT then
9          model ← ν
10         μ ← |{r ∈ V_R | ν(r) = 1}|   // number of r variables assigned to 1
11         φ_W ← φ_W ∪ {CNF(Σ_{r∈V_R} r ≤ μ − 1)}
12     else
13         if model = ∅ then
14             return UNSAT          // the MaxSAT formula is unsatisfiable
15         else
16             return model           // return satisfiable assignment to φ
```

at most k literals can be assigned truth value 1. Although cardinality constraints do not occur in MaxSAT formulations, several algorithms for MaxSAT rely on these constraints [3, 11, 19]. Usually, cardinality constraints are encoded to CNF so that a SAT solver can handle the resulting formula [9, 20, 7].

In the last decade, several algorithms for solving MaxSAT have been proposed. They can be mostly categorized into branch and bound algorithms [5, 16, 10, 15], linear search algorithms [14, 13] and unsatisfiability-based algorithms [17, 3, 4, 11]. Since this paper focus on new methods for linear search algorithms, the next section provides a detailed description of linear search algorithms for partial MaxSAT.

2.1 Linear search algorithms for partial MaxSAT

Algorithm 1 shows the traditional linear search algorithm for partial MaxSAT [14, 13]. The algorithm starts by relaxing the partial MaxSAT formula. For each soft clause ω, a new variable is created and added to ω (lines 2-3). Next, the relaxed soft clause ω_R is added to the working formula φ_W. The goal is to find an assignment to the problem variables such that it minimizes the number of relaxation variables that are assigned truth value 1. In any optimal solution, if a relaxation variable is assigned truth value 1, it corresponds to the unsatisfiability of a soft clause in the original partial MaxSAT formula.

Linear search algorithms work by iteratively calling a SAT solver over a working formula φ_W. A SAT solver returns a triple (st, ν, φ_C), where st denotes the outcome of the solver: satisfiable (SAT) or unsatisfiable (UNSAT). If the solver returns SAT, then the model that satisfies all clauses is stored in ν. On

the other hand, if the solver returns UNSAT, then φ_C contains an unsatisfiable subformula.

The working formula φ_W is then given to the SAT solver. While the working formula remains satisfiable, the model ν provided by the SAT solver is stored and we compute the upper bound value μ corresponding to the model ν. This upper bound value corresponds to the number of soft clauses that are unsatisfied in the original partial MaxSAT formula and consequently to the number of relaxation variables that are assigned truth value 1 (line 10). Next, the working formula φ_W is updated by adding a cardinality constraint that excludes models with a cost greater than or equal to μ. This procedure is repeated until the SAT solver returns unsatisfiable. When this occurs, we have found an optimal solution to φ (line 16). If there was no satisfiable call to the SAT solver, the original formula is unsatisfiable and the algorithm returns UNSAT (line 14).

In practice, cardinality constraints are usually encoded into CNF [9, 20, 7] so that a SAT solver can be called. Moreover, for several cardinality encodings we do not need to re-encode the cardinality constraint at each SAT call. Since the upper bound value is always decreasing, it is possible to update the CNF representation of the cardinality constraint by setting some specific literals to false. This procedure is denoted by *incremental strengthening* [7].

Example 2. Consider the partial MaxSAT formula φ as defined in Equation (1). Algorithm 1 starts by relaxing the partial MaxSAT formula by introducing a new relaxation variable in each soft clause. As a result, the working formula φ_W will be updated as follows:

$$\begin{aligned}\varphi_W = \{&(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4),\\ &(x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5)\}\end{aligned} \tag{2}$$

The set of relaxation variables is $V_R = \{r_1, r_2, r_3, r_4, r_5\}$. Next, φ_W is given to a SAT solver. Suppose that the SAT solver returns the following satisfiable assignment ν: $x_2 = x_3 = x_4 = r_1 = r_4 = 0$, $x_1 = r_2 = r_3 = r_5 = 1$. We store ν in *model* and compute the upper bound value μ. Since r_2, r_3 and r_5 were assigned truth value 1, we have found a solution that unsatisfies three soft clauses. Therefore, we can update the upper bound value μ to 3.

The working formula is now updated with a cardinality constraint over all relaxation variables, such that assignments corresponding to the unsatisfiability of 3 or more soft clauses are excluded. As a result, the working formula is now as follows:

$$\begin{aligned}\varphi_W = \{&(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4),\\ &(x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5),\\ &\text{CNF}(r_1 + r_2 + r_3 + r_4 + r_5 \leq 2)\}\end{aligned} \tag{3}$$

After updating the working formula, the algorithm makes another call to the SAT solver. Consider that the SAT solver returns another satisfiable assignment

ν: $x_1 = x_2 = x_3 = r_2 = r_4 = r_5 = 0$, $x_4 = r_1 = r_3 = 1$. Since two relaxation variables were assigned truth value 1, then $\mu = 2$. Next, we update the cardinality constraint of the working formula to CNF($\sum_{r \in V_R} r \leq 1$).

Finally, we make another call to the SAT solver which returns unsatisfiable. This means that there is no satisfiable assignment such that only one of the relaxation variables is assigned truth value 1. Therefore, the optimal solution is given by the previous model that corresponds to unsatisfying two soft clauses.

3 Model-based Algorithms for MaxSAT

In the previous section, we have seen that linear search algorithms add cardinality constraints over all relaxation variables. If the number of relaxation variables is large, then this can lead to the exploration of a large search space.

In contrast with linear search algorithms, one may use the models given by the SAT solver to iteratively increase the set of relaxation variables that are used in the cardinality constraint. By using only a subset of relaxation variables, we are able to intensify the search on a smaller search space.

Recently, a model-based algorithm [12] was proposed for solving the Minimum Satisfiability (MinSAT) problem. The goal of the MinSAT problem is to find an assignment to the problem variables such that it minimizes the number of satisfied clauses. Even though this algorithm was not proposed for MaxSAT, the authors mention that it can be adapted to solve MaxSAT problems [12].

In this section we present model-based algorithms for solving partial MaxSAT problems. The first algorithm adapts the model-based algorithm from MinSAT to MaxSAT. The second algorithm extends the first algorithm by inducing a clear partitioning between the relaxation variables that are used in the cardinality constraint and the ones that are not used.

3.1 Model-based Algorithm

Algorithm 2 shows our adaptation for solving partial MaxSAT problems of the model-based algorithm first proposed to solve MinSAT [12]. Similarly to the linear search algorithm, the model-based algorithm starts by relaxing the soft clauses of the MaxSAT formula (lines 2-5).

The working formula φ_W is then given to the SAT solver. At each call of the solver, if the working formula remains satisfiable, then the set of active relaxation variables V_A is updated (line 9). This set corresponds to relaxation variables that will be used in the cardinality constraint.

If the value μ' of the new satisfiable assignment is less than the best known upper bound value μ, then μ is updated and the current model is stored. Note that, since we do not add a cardinality constraint over all relaxation variables, it is possible to obtain models that have upper bound values greater than the ones that were known before. Next, we add a cardinality constraint over the active relaxation variables that excludes models using more than μ active relaxation variables (line 14).

Algorithm 2: Model-based algorithm for partial MaxSAT

Input: $\varphi = \varphi_h \cup \varphi_s$
Output: satisfiable assignment to φ or UNSAT

1 $(V_R, V_A, \mathsf{model}, \mu, \mu', \varphi_W) \leftarrow (\emptyset, \emptyset, \emptyset, +\infty, +\infty, \varphi_h)$
2 **foreach** $\omega \in \varphi_s$ **do**
3 $\quad V_R \leftarrow V_R \cup \{r\}$ `// r is a new variable`
4 $\quad \omega_R \leftarrow \omega \cup \{r\}$ `// relax soft clause`
5 $\quad \varphi_W \leftarrow \varphi_W \cup \omega_R$
6 **while** true **do**
7 $\quad (st, \nu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi_W)$
8 $\quad$ **if** $st = \mathsf{SAT}$ **then**
9 $\quad\quad V_A \leftarrow V_A \cup \{r \in V_R \mid \nu(r) = 1\}$ `// update active r variables`
10 $\quad\quad \mu' \leftarrow |\{r \in V_R \mid \nu(r) = 1\}|$
11 $\quad\quad$ **if** $\mu' < \mu$ **then**
12 $\quad\quad\quad \mathsf{model} \leftarrow \nu$
13 $\quad\quad\quad \mu \leftarrow \mu'$ `// update the upper bound value`
14 $\quad\quad \varphi_W \leftarrow \varphi_W \cup \{\mathtt{CNF}(\sum_{r \in V_A} r \leq \mu - 1)\}$
15 $\quad$ **else**
16 $\quad\quad$ **if** $\mathsf{model} = \emptyset$ **then**
17 $\quad\quad\quad$ **return** UNSAT
18 $\quad\quad$ **else**
19 $\quad\quad\quad$ **return** model `// return satisfiable assignment to` φ

This procedure is repeated until the formula becomes unsatisfiable. When this occurs, we have found an optimal solution to φ (line 19). Similarly to the linear search algorithm, if there was no satisfiable call to the SAT solver, the original formula is unsatisfiable and the algorithm returns UNSAT (line 17).

The main difference between Algorithm 1 and Algorithm 2 is the set of relaxation variables that is being used in the cardinality constraint. The model-based algorithm uses the models to iteratively increase the number of active relaxation variables used in the cardinality constraint. This allows the search to focus on a subset of the relaxation variables. As a side effect, Algorithm 2 may require additional SAT calls, since some of those calls may not improve the current upper bound but only increase the set of active relaxation variables.

Every time a new upper bound is found, we update the working formula with the corresponding cardinality constraint. However, these cardinality constraints may use different relaxation variables. Therefore, we cannot use incremental strengthening as in linear search algorithms.

Example 3. Consider again the same partial MaxSAT formula φ as defined in Equation (1). Similarly to the linear search algorithm, Algorithm 2 starts by relaxing the partial MaxSAT formula. As a result, the working formula φ_W will be updated as follows:

$$\varphi_W = \{(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4), \\ (x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5)\} \tag{4}$$

Next, φ_W is given to a SAT solver. Suppose it returns the following satisfiable assignment ν: $x_2 = x_3 = x_4 = r_1 = r_4 = 0$, $x_1 = r_2 = r_3 = r_5 = 1$.

We update the set of active relaxation variables V_A by extending it with the relaxation variables that were assigned truth value 1. Therefore, the set of active variables is now updated to $V_A = \{r_2, r_3, r_5\}$. At each SAT call, if the value of the new satisfying assignment μ' is less than the best known upper bound value μ, then we update μ to μ'. Moreover, if μ is updated we also store the corresponding model ν in *model*. Since the new satisfiable assignment improves our best known upper bound, we update the upper bound μ to 3.

The working formula is now updated with a cardinality constraint over the active relaxation variables. As a result, the working formula is now as follows:

$$\varphi_W = \{(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4), \\ (x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5), \\ \text{CNF}(r_2 + r_3 + r_5 \leq 2)\} \tag{5}$$

The SAT solver is called again on the working formula. Suppose it returns the satisfiable assignment ν: $x_1 = x_2 = x_3 = x_4 = r_2 = r_4 = 0$, $r_1 = r_3 = r_5 = 1$.

Since r_1 is assigned truth value 1 and does not occur in V_A, we must add r_1 to V_A. The value μ' for the new satisfiable assignment is 3, since three relaxation variables are assigned truth value 1. However, μ' does not improve the best known upper bound value μ. Therefore, μ does not change, but the cardinality constraint over V_A must be updated. As a result, the cardinality constraint of the working formula is updated to $\text{CNF}(r_1 + r_2 + r_3 + r_5 \leq 2)$.

Consider that in the next call to the SAT solver a new satisfiable assignment ν is found: $x_1 = r_2 = r_3 = r_5 = 0$, $x_2 = x_3 = x_4 = r_1 = r_4 = 1$. Since r_4 is assigned truth value 1 and does not occur in V_A, r_4 is added to V_A. The value μ' of the new satisfiable assignment is 2. Hence, μ is updated to 2 and the current model is saved.

The cardinality constraint of the working formula is now $\text{CNF}(r_1 + r_2 + r_3 + r_4 + r_5 \leq 1)$ and another call to the SAT solver is made. Finally, the SAT solver returns unsatisfiable. The optimal solution was found and is given by the previous model that corresponds to unsatisfying 2 soft clauses.

3.2 Model-based Partitioning Algorithm

The model-based algorithm presented in Algorithm 2 does not impose restrictions on the relaxation variables that are not active. This may lead to calls to the SAT solver returning new satisfiable assignments that do not improve the upper bound value. Moreover, if we impose restrictions over the relaxation variables, then we can further reduce the search space.

Algorithm 3: Model-based partitioning algorithm for MaxSAT

Input: $\varphi = \varphi_h \cup \varphi_s$
Output: satisfiable assignment to φ or UNSAT

1 $(V_R, \mathsf{model}, \mu, \varphi_W) \leftarrow (\emptyset, \emptyset, +\infty, \varphi_h)$
2 **foreach** $\omega \in \varphi_s$ **do**
3 $\quad V_R \leftarrow V_R \cup \{r\}$ `// r is a new variable`
4 $\quad \omega_R \leftarrow \omega \cup \{r\}$ `// relax soft clause`
5 $\quad \varphi_W \leftarrow \varphi_W \cup \omega_R$
6 $\varphi_{init} \leftarrow \varphi_W$ `// store the original relaxed formula`
7 $(st, \nu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi_W)$ `// get the first model`
8 **if** $st = \mathsf{SAT}$ **then**
9 $\quad V_A \leftarrow \{r \in V_R \mid \nu(r) = 1\}$ `// set of active r variables`
10 $\quad V_I \leftarrow V_R \setminus V_A$ `// set of inactive r variables`
11 $\quad \mathsf{model} \leftarrow \nu$
12 $\quad \mu \leftarrow |V_A|$
13 $\quad \varphi_W \leftarrow \varphi_W \cup \{(\neg r) \mid r \in V_I\} \cup \{\mathtt{CNF}(\sum_{r \in V_A} r \leq \mu - 1)\}$
14 **else**
15 $\quad$ **return** UNSAT `// the MaxSAT formula is unsatisfiable`
16 **while** true **do**
17 $\quad (st, \nu, \varphi_C) \leftarrow \mathtt{SAT}(\varphi_W)$
18 $\quad$ **if** $st = \mathsf{SAT}$ **then**
19 $\quad\quad \mathsf{model} \leftarrow \nu$
20 $\quad\quad \mu \leftarrow |\{r \in V_A \mid \nu(r) = 1\}|$
21 $\quad\quad \varphi_W \leftarrow \varphi_W \cup \{\mathtt{CNF}(\sum_{r \in V_A} r \leq \mu - 1)\}$
22 $\quad$ **else**
23 $\quad\quad$ **if** $\varphi_C \cap \{(\neg r) \mid r \in V_I\} = \emptyset$ `// core does not depend on inactive r variables`
24 $\quad\quad$ **then**
25 $\quad\quad\quad$ **return** model `// return satisfiable assignment to` φ
26 $\quad\quad$ **else**
27 $\quad\quad\quad V_A \leftarrow V_A \cup \{r \mid (\neg r) \in \varphi_C \cap r \in V_I\}$ `// update active r variables`
28 $\quad\quad\quad V_I \leftarrow V_I \setminus V_A$ `// update inactive r variables`
29 $\quad\quad\quad \varphi_W \leftarrow \varphi_{init} \cup \{(\neg r) \mid r \in V_I\} \cup \{\mathtt{CNF}(\sum_{r \in V_A} r \leq \mu - 1)\}$

Algorithm 3 shows the model-based partitioning algorithm. This algorithm extends the model-based algorithm by disabling the relaxation variables that are not active. The goal is to make an optimistic assumption that non active relaxation variables can be assigned value 0. If this is not the case, the working formula becomes unsatisfiable. However, in case the formula is unsatisfiable, current SAT solvers are able to provide certificates of unsatisfiability. In our algorithm, these certificates of unsatisfiability are then used to extend the set of active relaxation variables until the working formula becomes satisfiable.

As with previous algorithms, Algorithm 3 also starts by relaxing the MaxSAT formula (lines 2-5). However, notice that the initial working formula φ_{init} is stored for latter use (line 6). Next, the working formula φ_W is given to the

SAT solver. The first model is used to partition the relaxation variables into two disjoint sets: active relaxation variables and inactive relaxation variables (lines 9-10). Active relaxation variables (V_A) are used in the cardinality constraint, whereas inactive relaxation variables (V_I) are added as unit clauses to disable their occurrence in the next model (line 13). Note that, if the first call to the SAT solver returns unsatisfiable, then the MaxSAT formula is unsatisfiable and the algorithm returns UNSAT (line 15).

The current working formula is again given to the SAT solver. If the formula is satisfiable then a new better solution has been found and we update the cardinality constraint as in the linear search algorithm (line 21). However, if the SAT solver returns unsatisfiable then we need to analyze the unsatisfiable subformula φ_C. If φ_C does not contain unit clauses with inactive relaxation variables, then an optimal solution has been found and we return the last stored model (line 25).

Otherwise, φ_C contains unit clauses with inactive relaxation variables. Therefore, the inactive relaxation variables in those unit clauses are added to V_A (line 27) and removed from V_I (line 28). Afterwards, the working formula is rebuilt from φ_{init} together with the cardinality constraint over the updated set of active relaxation variables and the unit clauses of the updated set of inactive relaxation variables.

Notice that when the working formula becomes unsatisfiable, it must be rebuilt. All learned clauses from the previous SAT call are removed since they may not be valid in the next SAT call. It is clear that a more selective cleaning process can be devised, but it is not included in our current implementation. Another drawback of this approach is that one can have a large number of consecutive unsatisfiable calls to the SAT solver. In our implementation, if the SAT solver returns more than 3 consecutive unsatisfiable calls, then we set V_A to V_R and V_I to $\emptyset$ and the algorithm proceeds as in the classical linear search approach.

Example 4. Consider again the same partial MaxSAT formula φ as defined in Equation (1). Similarly to the previous algorithms, the initial working formula φ_W is the relaxed partial MaxSAT formula.

$$\begin{aligned}\varphi_W = \{&(\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4),\\ &(x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5)\}\end{aligned} \tag{6}$$

Suppose that the first call to the SAT solver (line 7) returns the following satisfiable assignment ν: $x_2 = x_3 = x_4 = r_1 = r_4 = 0$, $x_1 = r_2 = r_3 = r_5 = 1$.

We update the set of active relaxation variables to $V_A = \{r_2, r_3, r_5\}$ and the set of inactive relaxation variables to $V_I = \{r_1, r_4\}$. Moreover, the current model is saved and μ is set to 3.

The working formula is now updated with a cardinality constraint over the active relaxation variables. Moreover, we also add to the working formula the unit clauses that disable the inactive relaxation variables. As a result, the working formula is now as follows:

$$\varphi_W = \{\ (\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4), \\ (x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5), \\ (\bar{r}_1), (\bar{r}_4), \mathrm{CNF}(r_2 + r_3 + r_5 \leq 2)\} \quad (7)$$

After updating the working formula, the algorithm makes another call to the SAT solver. Consider that the SAT solver returns unsatisfiable and φ_C contains $(\bar{r}_1)$. We update the set of active relaxation variables to $V_A = \{r_1, r_2, r_3, r_5\}$ and the set of inactive relaxation variables to $V_I = \{r_4\}$. Next, we rebuild the working formula from the initial relaxed MaxSAT formula (φ_{init}), together with the cardinality constraint over the active relaxation variables and the unit clauses from the inactive relaxation variables. As a result, the working formula is now as follows:

$$\varphi_W = \{\ (\bar{x}_2 \vee \bar{x}_1), (x_2 \vee \bar{x}_3), (\bar{x}_1 \vee \bar{x}_4), \\ (x_1 \vee r_1), (x_2 \vee \bar{x}_1 \vee r_2), (x_3 \vee r_3), (\bar{x}_3 \vee x_1 \vee r_4), (x_4 \vee r_5), \\ (\bar{r}_4), \mathrm{CNF}(r_1 + r_2 + r_3 + r_5 \leq 2)\} \quad (8)$$

Next, the SAT solver is called. Suppose it returns a satisfiable assignment ν: $x_1 = x_2 = x_3 = r_2 = r_4 = r_5 = 0$, $x_4 = r_1 = r_3 = 1$. A new better solution has been found and μ is set to 2. The cardinality constraint of the working formula is updated to $\mathrm{CNF}(r_1 + r_2 + r_3 + r_5 \leq 1)$.

In the next call, the formula is unsatisfiable and φ_C contains $(\bar{r}_4)$. The sets of active and inactive relaxation variables are updated to $V_A = \{r_1, r_2, r_3, r_4, r_5\}$ and $V_I = \emptyset$. The working formula is rebuilt as previously described.

Finally, the next call to the SAT solver returns unsatisfiable and φ_C does not contain any unit clauses from the inactive relaxation variables. Therefore, an optimal solution was already found and is given by the previous model that corresponds to unsatisfying 2 soft clauses.

4 Experimental Results

In this section we evaluate the performance of linear search algorithms in a selection of benchmark instance sets. We compare the solvers, `LinearMS`, `ModelMS`, and `PartMS` against `QMaxSAT` [13] (winner of the industrial category of partial MaxSAT in the MaxSAT evaluation of 2012).

`LinearMS` denotes our solver that implements the linear search algorithm described in section 2.1. `ModelMS` and `PartMS` denotes the solvers that implement the model-based algorithm described in section 3.1 and the model-based partitioning algorithm described in section 3.2, respectively. These solvers were implemented on top of `Glucose 2.1` [8] and use the cardinality networks encoding [7] to encode cardinality constraints into CNF. The unsatisfiable subformulas for the `PartMS` solver were extracted using an assumption-based approach [6].

All experiments were run on two AMD Opteron 6276 processors (2.3 GHz) running Linux Fedora Core 18 with a timeout of 1,800 seconds and a memory

Benchmark	#I	Avg. #Soft	LinearMS	ModelMS	PartMS	QMaxSAT
ms_industrial	55	2,382,542	7	3	11	8
pms_industrial	504	1,997	408	399	366	408
close_solution	486	166,105	187	210	201	183

Table 1. Number of instances solved by each solver.

limit of 16 GB. The evaluation was performed on the 55 industrial MaxSAT instances and on the 504 partial MaxSAT instances of the MaxSAT evaluation of 2012[1].

Additionally, we have also considered 486 instances coming from *close solution* problems[2] [1, 2]. The close solution problem is as follows. Consider a SAT formula φ and a model ν. If some new clauses are added to φ, then the goal is to find a new model ν' that is as similar as possible to ν. Similarity is measured as the number of assignments to the variables that are common to ν and ν'. This problem can be encoded into partial MaxSAT and may contain a large number of soft clauses.

Table 1 shows the total number of instances, the average number of soft clauses and the number of instances solved by each solver for the different benchmark sets.

Notice that the MaxSAT industrial benchmarks have a very large number of soft clauses. On average, each instance has over 2 million soft clauses. Moreover, note that these instances do not have hard clauses. Therefore, they are only composed of soft clauses and in some cases solvers can run out of memory when handling such huge MaxSAT formulas. If each soft clause is relaxed by adding a new relaxation variable, encoding a cardinality constraint into CNF using a huge set of relaxation variables can easily lead to memory problems. Note that we are using a memory limit of 16GB, which is a significantly larger limit than the one used in the MaxSAT evaluation of 2012 (450MB).

In contrast, partial industrial MaxSAT benchmarks have a small number of soft clauses, being on average 2,000 soft clauses per instance. Finally, the number of soft clauses in the close solution problems is not as large as in the industrial MaxSAT benchmarks but it is much larger than in the partial industrial benchmarks. On average, each close solution problem instance has over 160,000 soft clauses.

The number of instances solved by each solver show that the number of instances solved by `LinearMS` is similar to `QMaxSAT`. This shows that our baseline solver has a similar performance to the best linear search solver in the MaxSAT evaluation of 2012.

For the partial industrial MaxSAT benchmarks, model-based algorithms did not perform as well as our baseline solver. Moreover, model-based algorithms did not solve any instance that could not be solved by `LinearMS`. Since the average number of soft clauses is small, model-based algorithms tend to use the majority of the relaxation variables in the cardinality constraint to find

[1] Available at http://maxsat.ia.udl.cat/

[2] We would like to thank Ignasi Abío for his assistance with these benchmarks.

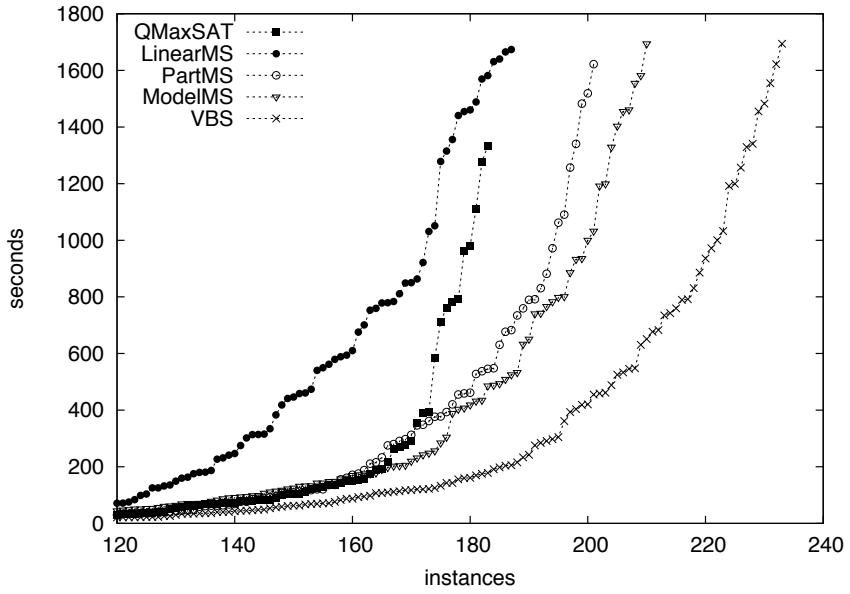

Fig. 1. Running times of solvers for the close solution problems.

the optimal solution. Therefore, starting with a large subset of the relaxation variables and iteratively increasing this set with a few more variables represents an overhead that deteriorates the performance of the solver.

For the industrial MaxSAT benchmarks, `PartMS` clearly outperformed the remaining solvers. `PartMS` can solve instances in this benchmark set by using a small fraction of the relaxation variables. Moreover, since the relaxation variables that do not belong to the cardinality constraint are not active, we are able to effectively prune the search space and solve more instances than `LinearMS`. On the other hand, `ModelMS` performed poorly in this benchmark set. Since `ModelMS` does not deactivate the relaxation variables that are not used in the cardinality constraint, it does not prune the search space efficiently for this huge number of soft clauses. `LinearMS` was able to solve 4 instances that were not solved by `PartMS`, whereas `ModelMS` was able to solve 1 instance that was not solved by the other solvers. This shows that, for this set of benchmarks, each algorithm is able to solve some instances that others are unable to.

For the close solution problems, `ModelMS` exhibited the best performance. This solver is able to solve most of the the instances using only a small subset of the relaxation variables. On average, `ModelMS` uses only around 20% of the relaxation variables in the cardinality constraint. On the other hand, `PartMS` needs to use more relaxation variables to solve instances from this benchmark set. Using the unsatisfiable subformulas to increase the set of used relaxation variables may lead to the unnecessary inclusion of some relaxation variables in the cardinality constraint. Moreover, the opposite problem may also happen. If the unsatisfiable subformulas are small, then we may require several unsatisfiable

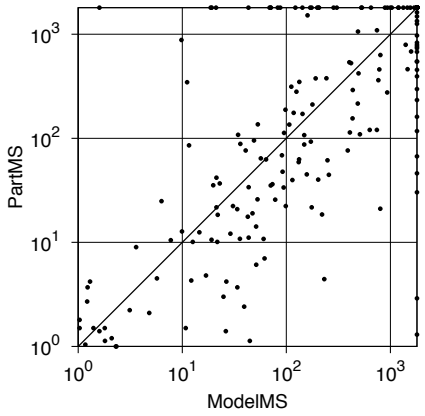

Fig. 2. Running times of `ModelMS` and `PartMS` for the close solution problems.

calls until we reach a satisfiable state. To prevent this effect, we imposed a limit on the consecutive number of unsatisfiable calls. Even though it is better to impose a limit on the number of consecutive unsatisfiable calls, this is a limitation of the current approach since we will then use all relaxation variables.

For the close solution problems, `PartMS` was able to solve 23 instances that were not solved by `ModelMS`, whereas `LinearMS` was able to solve 4 instances that were not solved by the other solvers. This shows that for this set of benchmarks `ModelMS` and `PartMS` are complementary, since they can solve a significant number of different instances.

Since both model-based algorithms improve the performance of linear search algorithms for the close solution problems, we will analyze in more detail its performance for this benchmark set. Figure 1 shows a cactus plot with the running times of the different solvers for the close solution problems. In addition to the solvers previously mentioned, we have also included the Virtual Best Solver (VBS) between `ModelMS` and `PartMS`. The VBS shows the number of instances that were either solved by `ModelMS` or by `PartMS`.

`LinearMS` solves more instances than `QMaxSAT` but it is slower than `QMaxSAT` for most benchmarks. Note that `QMaxSAT` uses a different encoding for cardinality constraints. Different encodings for cardinality constraints may change the performance of linear search algorithms [18]. Even though the encoding used in `LinearMS` is more compact, it may perform worse than the one implemented on top of `QMaxSAT`.

`ModelMS` and `PartMS` clearly improve our baseline solver. Moreover, the proposed algorithms not only solve more instances than `LinearMS` but also decrease the runtime needed. `ModelMS` is the best performing solver for this benchmark set since it was able to solve more 9 instances than `PartMS`. However, each solver is able to solve instances that are not solved by the other solvers. Therefore, for a given instance it is not clear which solver will perform the best. This provides a strong stimulus for further research in a hybrid algorithm between `LinearMS`,

ModelMS and PartMS, that during the search would be able to adjust its search algorithm according to the instance that is being solved.

Figure 2 shows a scatter plot with the runtime of ModelMS and PartMS. Each point in the plot corresponds to a problem instance, where the x-axis corresponds to the runtime required by ModelMS and the y-axis the runtime required by PartMS. Instances that are trivially solved by both approaches (in less than 1 second) are not shown in the plot. The scatter plot provides a better understanding of the comparison between two different solvers. Even though, ModelMS solves more instances than PartMS, it is slower than PartMS for most instances. Moreover, as mentioned before, there are several instances that are only solved by one of the solvers. ModelMS solves 32 instances that are not solved by PartMS, whereas PartMS solves 23 instances that were not solved by ModelMS.

5 Conclusions

Linear search algorithms for MaxSAT have shown to be particularly effective when the number of soft clauses is small. However, if the number of soft clauses increases, then the performance of linear search algorithms tends to deteriorate. This is due to the fact that the cardinality constraint is expressed over all relaxation variables.

In this paper, we have described two model-based algorithms that start by imposing a cardinality constraint over a subset of the relaxation variables. This can prune the search space significantly, which allows solving instances that could not be solved by linear search algorithms. Model-based algorithms have shown to be effective for problem instances where the number of soft clauses is large. However, for problems with a small number of soft clauses, the overhead induced by iteratively increasing the subset of used relaxation variables deteriorates the performance of the linear search algorithm.

As future work, we propose to build a hybrid solver that can dynamically adapt its search for a given instance. This is motivated by the fact that the virtual best solver clearly outperformed any singular solver. Moreover, the same approach can be extended for weighted MaxSAT problems by using appropriate CNF encodings.

Acknowledgements

This work was partially supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under projects PEst-OE/EEI/LA0021/2013, iExplain (PTDC/EIA-CCO/102077/2008) and ASPEN (PTDC/EIA-CCO/110921/-2009).

References

1. I. Abío, M. Deters, R. Nieuwenhuis, and P. J. Stuckey. Reducing Chaos in SAT-Like Search: Finding Solutions Close to a Given One. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 273–286, 2011.

2. I. Abío and P. J. Stuckey. Conflict Directed Lazy Decomposition. In *Principles and Practice of Constraint Programming*, pages 70–85, 2012.
3. 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.
4. C. Ansótegui, M. Bonet, and J. Levy. A New Algorithm for Weighted Partial MaxSAT. In *AAAI Conference on Artificial Intelligence*, pages 3–8, 2010.
5. 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.
6. R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Practical algorithms for unsatisfiability proof and core generation in SAT solvers. *AI Communications*, 23(2-3):145–157, 2010.
7. R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality Networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.
8. G. Audemard and L. Simon. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *International Joint Conferences on Artificial Intelligence*, pages 399–404, 2009.
9. O. Bailleux and Y. Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming*, pages 108–122, 2003.
10. F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.
11. F. Heras, A. Morgado, and J. Marques-Silva. Core-Guided Binary Search Algorithms for Maximum Satisfiability. In *AAAI Conference on Artificial Intelligence*, pages 36–41, 2011.
12. F. Heras, A. Morgado, J. Planes, and J. Marques-Silva. Iterative SAT Solving for Minimum Satisfiability. In *International Conference on Tools with Artificial Intelligence*, pages 922–927, 2012.
13. M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A Partial Max-SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 8:95–100, 2012.
14. D. Le Berre and A. Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):59–6, 2010.
15. C. M. Li, F. Manyà, and J. Planes. New Inference Rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, 2007.
16. H. Lin and K. Su. Exploiting Inference Rules to Compute Lower Bounds for MAX-SAT Solving. In *International Joint Conferences on Artificial Intelligence*, pages 2334–2339, 2007.
17. 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.
18. R. Martins, V. Manquinho, and I. Lynce. Exploiting Cardinality Encodings in Parallel Maximum Satisfiability. In *International Conference on Tools with Artificial Intelligence*, pages 313–320, 2011.
19. R. Martins, V. Manquinho, and I. Lynce. Parallel Search for Maximum Satisfiability. *AI Communications*, 25(2):75–95, 2012.
20. C. Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming*, pages 827–831, 2005.