On Partitioning for Maximum Satisfiability

Ruben Martins[1] and **Vasco Manquinho**[1] and **Inês Lynce** [1]

Abstract. Partitioning formulas is motivated by the expectation to identify easy to solve subformulas, even though at the cost of having more formulas to solve. In this paper we suggest to apply partitioning to Maximum Satisfiability (MaxSAT), the optimization version of the well-known Satisfiability (SAT) problem. The use of partitions can be naturally combined with unsatisfiability-based algorithms for MaxSAT that are built upon successive calls to a SAT solver, where each call identifies an unsatisfiable subformula. One of the drawbacks of these algorithms results from the SAT solver returning large unsatisfiable subformulas. However, when using partitions, the solver is more likely to identify smaller unsatisfiable subformulas. Experimental results show that the use of partitions in MaxSAT significantly improves the performance of unsatisfiability-based algorithms.

1 Maximum Satisfiability

Maximum Satisfiability (MaxSAT) can be seen as an optimization version of Boolean Satisfiability (SAT) which consists in finding an assignment to the variables such that it minimizes (maximizes) the number of unsatisfied (satisfied) clauses. MaxSAT has several variants and can be generalized to the weighted partial MaxSAT problem. In this problem, some clauses are declared as hard, while the rest are declared as soft. The objective is to find an assignment to the variables such that all hard clauses are satisfied, while minimizing the total weight of unsatisfied soft clauses. For a more detailed introduction to MaxSAT we point the reader to the literature [5].

Unsatisfiability-based algorithms for weighted partial MaxSAT [6, 1, 2, 3] iteratively identify and relax unsatisfiable subformulas. In this paper we propose to improve these algorithms by using a new technique based on partitioning soft clauses. Instead of using the initial weighted partial MaxSAT formula to search for unsatisfiable subformulas, this paper proposes to start with a smaller formula that only contains a partition of the soft clauses. At each iteration, the formula is constrained by adding one more partition of soft clauses. This procedure is repeated until all partitions are added to the formula. The motivation for this technique is twofold. First, at each iteration we are solving formulas that are less constrained than the initial formula. Although the number of iterations may be larger than when not using partitions, each iteration is expected to require less time. As a result, more iterations may not imply using more computational time at the end. Second, by splitting soft clauses into partitions, the search is focused on a given subset of soft clauses. This can lead to finding smaller unsatisfiable subformulas that are less likely to be found if we consider the whole set of soft clauses.

[1] INESC-ID/IST, TU Lisbon, Portugal, {ruben, vmm, ines}@sat.inesc-id.pt. This work was partially supported by FCT under research projects PTDC/EIACCO/102077/2008 and PTDC/EIA-CCO/110921/2009, and INESC-ID multiannual funding through the PIDDAC program funds.

Algorithm 1: Unsatisfiability-based algorithm for weighted partial MaxSAT enhanced with partitioning of soft clauses

```
Input: $\varphi = \varphi_h \cup \varphi_s$
Output: satisfiable assignment to $\varphi$ or UNSAT
1  $\gamma \leftarrow \langle \gamma_1, \ldots, \gamma_n \rangle \leftarrow \texttt{partitionSoft}(\varphi_s)$
2  $\varphi_W \leftarrow \varphi_h$
3  while true do
4      $\varphi_W \leftarrow \varphi_W \cup \texttt{first}(\gamma)$
5      $\gamma \leftarrow \gamma \setminus \texttt{first}(\gamma)$
6      $(\mathsf{st}, \varphi_C) \leftarrow \texttt{SAT}(\varphi_W)$
7      if st = UNSAT then
8          $\mathsf{min_C} \leftarrow \texttt{min}\{\texttt{weight}(\omega) \mid \omega \in \varphi_C \wedge \texttt{soft}(\omega)\}$
9          $V_R \leftarrow \emptyset$
10         foreach $\omega \in \varphi_C \wedge \texttt{soft}(\omega)$ do
11             $V_R \leftarrow V_R \cup \{r\}$        // r is a new variable
12             $\omega_R \leftarrow \omega \cup \{r\}$              // relax soft
13             $\texttt{weight}(\omega_R) \leftarrow \mathsf{min_C}$
14             if $\texttt{weight}(\omega) > \mathsf{min_C}$ then
15                 $\varphi_W \leftarrow \varphi_W \cup \{\omega_R\}$    // duplicate soft
16                 $\texttt{weight}(\omega) \leftarrow \texttt{weight}(\omega) - \mathsf{min_C}$
17             else $\varphi_W \leftarrow \varphi_W \setminus \{\omega\} \cup \{\omega_R\}$
18         if $V_R = \emptyset$ then return UNSAT
19         else $\varphi_W \leftarrow \varphi_W \cup \{\texttt{CNF}(\sum_{r \in V_R} r = 1)\}$
20     else
21         if $\gamma = \emptyset$ then return satisfiable assignment to $\varphi_W$
```

2 Partitioning Soft Clauses

Algorithm 1 illustrates an unsatisfiability-based algorithm for weighted partial MaxSAT [6, 1] enhanced with partitioning of soft clauses. The differences between algorithm 1 and the original unsatisfiability-based algorithm for weighted partial MaxSAT are highlighted. The algorithm takes as input a weighted partial MaxSAT formula φ that is composed by a set of hard clauses φ_h and a set of soft clauses φ_s. It begins by partitioning the soft clauses and placing them in an ordered list γ. At each iteration, a SAT solver is applied to the working formula φ_W. Initially, φ_W corresponds to φ_h. At each iteration, φ_W is augmented with the first partition from list γ (line 4). Next, the added partition is removed from γ (line 5). A SAT solver is then applied to φ_W returning a pair (st, φ_C) where st denotes the outcome of the solver: SAT or UNSAT. If the outcome is UNSAT, then φ_C contains the unsatisfiable subformula identified by the SAT solver. In this latter case, the unsatisfiable subformula is relaxed as in the original algorithm [6, 1]. On the other hand, if the solver outcome is SAT and there are no more partitions of soft clauses in γ, then the solver found an optimal solution to the original weighted partial MaxSAT formula. However, if γ is not empty, then φ_W is extended with a new partition from γ (line 4) and the algorithm proceeds.

Weight-based Partitioning. The most natural form of partitioning soft clauses is to use their weight. With this technique, soft clauses with the same weight belong to the same partition. These soft clauses are more likely to be related to each other than to the remaining ones. Note that if we sort the partitions of soft clauses from the largest to the smallest weight, then we can improve algorithm 1. Consider a weighted partial MaxSAT formula with only 2 weights associated with the soft clauses, 1 and 100. Moreover, consider also that a soft clause ω with weight 100 must be relaxed in the optimal solution. In the first iteration the unsatisfiable subformula given by the SAT solver contains clause ω and soft clauses with weight 1. Therefore, the weight of ω is decreased by 1 and a relaxed copy of ω is created. In the worst case, this procedure can be repeated up to 100 times in order to completely relax ω. Now, consider the scenario where the soft clauses have been partitioned by weight. If an unsatisfiable subformula with ω is found, then only one iteration is required to relax ω since all soft clauses that belong to the unsatisfiable subformula have weight 100. An important optimization when using weight-based partitioning is to dynamically put the soft clauses that are duplicated into the partition having soft clauses with the same weight. This procedure may dynamically create new partitions. (For the sake of simplicity, this optimization is not shown in algorithm 1.)

Even though the proposed approach is novel, there has been some related work on using weights to guide the search. `MSUncore` with lexicographical optimization [7] is dedicated to solving problem instances where the optimality criterion is lexicographic. Soft clauses are grouped by their weight to iteratively find an optimal solution to each criterion. The version of `WPM1` [1] from the MaxSAT 2011 evaluation considers the weights of soft clauses to find unsatisfiable subformulas with larger weights first [2].

Graph-based Partitioning. For some problem instances, the weights of the soft clauses may not form natural partitions. For example, if a formula has all soft clauses with different weights, then each partition has one soft clause. Therefore, for these cases the formula should be partitioned using other techniques. A possible alternative is graph-based partitioning, namely hypergraph partitioning. A hypergraph is a generalization of a graph where an edge can connect any number of vertices. To build a hypergraph from a weighted partial MaxSAT formula, the soft and hard clauses of the formula are considered as the vertices of the hypergraph. Each edge of the hypergraph represents a variable of the formula and connects all clauses (vertices) which contain that variable. This representation resembles the hypergraph obtained from a SAT formula [8]. In our experimental evaluation, graph-based partitioning is used instead of weight-based partitioning when the number of partitions is large (> 300) and the average number of soft clauses in each partition is small (< 3). The tool `hmetis` [4] was used to partition the hypergraph into 16 partitions. For each partition, only the soft clauses are considered.

3 Experimental Results and Discussion

All experiments were run on the weighted partial MaxSAT instances from the crafted and industrial categories of the MaxSAT evaluation of 2011.The evaluation was performed on two AMD Opteron 6172 processors with a timeout of 1,200 seconds. Our new solver based on partitioning soft clauses (`PAR`) was built on top of `WBO` [6]. The performance of `PAR` has been compared against the following unsatisfiability-based algorithms: `MSUncore` [7] using lexicographical optimization (`MSU bmo`), `MSUncore` [3] using core-guided binary search with disjoint cores (`MSU bin-cd`), `WPM1` [3] [1], `WPM2` [3] [2], and `WBO` [6].

[2] Personal communication from the author.

Table 1. Number of instances solved by each solver.

Benchmark	#I	MSU bmo	MSU bin-cd	WPM1	WPM2	WBO	PAR
paths	86	0	0	33	0	0	7
scheduling	84	0	66	81	3	0	78
planning	56	26	53	54	39	31	50
warehouses	18	2	3	1	1	4	14
miplib	12	1	2	3	2	0	2
net	74	18	0	0	0	53	41
dir	21	5	13	5	9	5	7
log	21	5	11	6	8	6	6
pedigrees	100	86	26	87	44	73	80
timetabling	26	5	6	8	9	5	5
upgrade	100	100	98	100	98	100	100
Total	598	248	278	378	213	277	390

Table 1 shows the number of instances solved by each solver. `PAR` clearly outperforms `WBO` showing that partitioning soft clauses can significantly improve the performance of unsatisfiability-based algorithms. `PAR` is more efficient than `WBO` since it is able to find the optimal solution while making less calls to the SAT solver. On average, `WBO` performs 664 iterations, whereas `PAR` only needs 329. Moreover, `PAR` is also able to find smaller unsatisfiable subformulas than `WBO`. On average, `PAR` finds unsatisfiable subformulas with 57 soft clauses, whereas unsatisfiable subformulas in `WBO` have 72 soft clauses. The benchmarks `warehouses` and `net` were solved used graph-based partitioning. For these benchmarks, the use of weights for partitioning would create over 1,000 partitions, each of them containing on average slightly less than 2 soft clauses. The use of graph-based partitioning has mixed results. It improves the performance of the solver on the `warehouses` instances but it deteriorates the performance of the solver on the `net` instances. For the remaining benchmarks, weight-based partitioning was used. As a result, the solver's performance improved on several benchmarks, being most effective on the `scheduling` and `planning` instances. When compared to the remaining solvers, `PAR` is the most robust solver as it solves the largest number of instances.

As future work one may consider additional forms of graph partitioning. Moreover, graph partitioning can also be used to partition soft clauses in unweighted MaxSAT formulas.

REFERENCES

[1] C. Ansótegui, M. Bonet, and J. Levy, 'Solving (Weighted) Partial MaxSAT through Satisfiability Testing', in *International Conference on Theory and Applications of Satisfiability Testing*, (2009).

[2] C. Ansótegui, M. Bonet, and J. Levy, 'A New Algorithm for Weighted Partial MaxSAT', in *AAAI Conference on Artificial Intelligence*, (2010).

[3] F. Heras, A. Morgado, and J. Marques-Silva, 'Core-Guided Binary Search Algorithms for Maximum Satisfiability', in *AAAI Conference on Artificial Intelligence*, (2011).

[4] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, 'Multilevel hypergraph partitioning: Application in VLSI domain', in *IEEE Transactions on VLSI Systems*, volume 7, (1999).

[5] C. M. Li and F. Manyà, 'MaxSAT, Hard and Soft Constraints', in *Handbook of Satisfiability*, IOS Press, (2009).

[6] V. Manquinho, J. Marques-Silva, and J. Planes, 'Algorithms for Weighted Boolean Optimization', in *International Conference on Theory and Applications of Satisfiability Testing*, (2009).

[7] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce, 'Boolean Lexicographic Optimization: Algorithms & Applications', *Annals of Mathematics and Artificial Intelligence*, **62**(3-4), (2011).

[8] T. J. Park and A. V. Gelder, 'Partitioning Methods for Satisfiability Testing on Large Formulas', *Inf. and Computation*, **162**(1-2), (2000).

[3] Version from the MaxSAT 2011 evaluation.