Community-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. Unsatisfiability-based algorithms for Maximum Satisfiability (MaxSAT) have been shown to be very effective in solving several classes of problem instances. These algorithms rely on successive calls to a SAT solver, where an unsatisfiable subformula is identified at each iteration. However, in some cases, the SAT solver returns unnecessarily large subformulas. In this paper a new technique is proposed to partition the MaxSAT formula in order to identify smaller unsatisfiable subformulas at each call of the SAT solver. Preliminary experimental results analyze the effect of partitioning the MaxSAT formula into communities. This technique is shown to significantly improve the unsatisfiability-based algorithm for different benchmark sets.

1 Introduction

Problem partitioning is a well-known technique used for general problem solving and it has already been proposed for Boolean optimization [1,2] formulations. The main goal of partitioning is to identify easier to solve subproblems such that it will help to solve the overall problem.

In recent years, several algorithms and solvers have been proposed for Maximum Satisfiability (MaxSAT). In particular, unsatisfiability-based MaxSAT solvers [3,4,5,6,7] have been shown to be very effective in tackling real-world problems [8]. These solvers are based on iteratively calling a SAT solver enhanced with the ability of providing a certificate of unsatisfiability. However, one drawback of these algorithms results from the SAT solver returning unnecessary large unsatisfiable subformulas as certificates. Instead of dealing initially with the whole formula, we start with a smaller formula that is extended at each iteration of the algorithm. The goal is to initially have smaller formulas that enable the SAT solver to provide smaller certificates of unsatisfiability.

In this paper we propose a new method for formula partitioning in an unsatisfiability-based algorithm for partial MaxSAT. A graph representation of the formula is used and graph communities are identified based on a modularity measure, thus allowing to build partitions of soft clauses in the MaxSAT formula. The paper is organized as follows. The next section introduces MaxSAT and briefly reviews the main approaches for MaxSAT solving. In section 3 an unsatisfiability-based algorithm with partitioning

* This work was partially supported by FCT under research projects iExplain (PTDC/EIA-CCO/102077/2008) and ASPEN (PTDC/EIA-CCO/110921/2009), and INESC-ID multiannual funding through the PIDDAC program funds (PEst-OE/EEI/LA0021/2011).

of soft clauses is described. Section 4 proposes partition methods for partial MaxSAT, in particular a new approach based on the identification of graph communities. Experimental results are presented in section 5 and the paper concludes in section 6.

2 Preliminaries

The Maximum Satisfiability (MaxSAT) problem is an optimization version of the Propositional Satisfiability (SAT) problem which consists in finding an assignment to the variables of the CNF formula such that the number of unsatisfied (satisfied) clauses is minimized (maximized). 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 a partial MaxSAT formula $\varphi = \varphi_h \cup \varphi_s$, some clauses are declared as hard (φ_h), while the rest are declared as soft (φ_s). The objective in partial MaxSAT is to find an assignment to formula variables such that all hard clauses φ_h are satisfied, while minimizing the number of unsatisfied soft clauses in φ_s. Finally, in the weighted versions of MaxSAT, soft clauses can have weights greater than or equal to 1 and the objective is to satisfy all hard clauses while minimizing the total weight of unsatisfied soft clauses.

2.1 MaxSAT Algorithms

In the last decade, several new techniques and algorithms for MaxSAT have been proposed [9], resulting in significant improvements in MaxSAT solvers. More recently, new algorithms have been devised that are more effective for solving industrial benchmark instances[1], namely linear search on the objective value of the MaxSAT instance and unsatisfiability-based algorithms.

In the linear search approach, a new relaxation variable is initially added to each soft clause and the resulting formula is solved by a SAT solver. Whenever a solution is found, a new constraint on the relaxation variables is added such that solutions with a higher or equal value are excluded. This new constraint is usually translated into a set of propositional clauses so that a SAT solver can handle the resulting formula [10]. Otherwise, a pseudo-Boolean solver must be used. The algorithm stops when the resulting formula becomes unsatisfied.

A different approach is trying to satisfy all hard and soft clauses using a SAT solver enhanced with the ability to produce certificates of unsatisfiability [3]. At each call of the SAT solver, an unsatisfiable subformula is identified and relaxed by adding a new relaxation variable to each soft clause in the unsatisfiable subformula. Additionally, a new constraint is added such that at most one of the new relaxation variables can be assigned value true. Again, in order to continue using a SAT solver, this new constraint must be encoded into a set of propositional clauses. Next, the SAT solver is called with the resulting formula. The algorithm stops when the formula becomes satisfiable. Several extensions of this approach have been proposed for solving MaxSAT and its variants [5,4,6,7].

[1] See results from MaxSAT Evaluations at `http://maxsat.ia.udl.cat/`

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

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

```
1  (st, φ_C) ← SAT(φ_h)        // check if the MaxSAT formula is UNSAT
2  if st = UNSAT then
3      return UNSAT
4  γ ← ⟨γ_1, ..., γ_n⟩ ← partitionSoft(φ_s)
5  φ_W ← φ_h
6  while true do
7      φ_W ← φ_W ∪ first(γ)
8      γ ← γ \ first(γ)
9      (st, φ_C) ← SAT(φ_W)
10     while st = UNSAT do
11         V_R ← ∅
12         foreach ω ∈ (φ_C ∩ φ_s) do
13             V_R ← V_R ∪ {r}                    // r is a new variable
14             ω_R ← ω ∪ {r}                      // relax soft clause
15             φ_W ← φ_W \ {ω} ∪ {ω_R}
16         φ_W ← φ_W ∪ {CNF(Σ_{r∈V_R} r = 1)}
17         (st, φ_C) ← SAT(φ_W)
18     if γ = ∅ then
19         return satisfiable assignment to φ_W
```

3 Partition-based MaxSAT Algorithm

In this section we review an unsatisfiability-based MaxSAT algorithm that takes advantage of partitioning. The original algorithm [2] was proposed for weighted variants of MaxSAT where partitions are built considering the different weights of soft clauses.

Algorithm 1 starts by checking if the MaxSAT instance φ is satisfiable by calling a SAT solver only with hard clauses φ_h. Next, the set of soft clauses φ_s is split into a list of partitions (line 4) such that each soft clause is assigned to one partition. Initially, the working formula only considers the hard clauses φ_h (line 5). At each iteration, a partition γ_i of soft clauses is added to the working formula (line 7) and removed from the partition list γ (line 8). 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. While the outcome is UNSAT, φ_C contains the unsatisfiable subformula identified by the SAT solver and the unsatisfiable subformula is relaxed as in the original algorithm [3]. Next, the SAT solver is applied to the modified working formula (line 17). After a given number of relaxations, the working formula becomes satisfiable[2] and a new partition of soft clauses is added to the working formula. If there are no more partitions in γ, then the solver found an optimal solution to the original MaxSAT formula (line 19).

[2] Notice that initially we already confirmed that the MaxSAT formula is not unsatisfiable due to the hard clauses. Since at each iteration at least one soft clause is relaxed, the working formula at some point becomes satisfiable.

In Algorithm 1, a partition method must be used to split the set of soft clauses. For weighted variants of MaxSAT, it was already shown that splitting the soft clauses according to its weight allows the solver to be much more effective [11,2]. However, for partial MaxSAT this partition method cannot be used since all soft clauses have weight 1. In the next section we present two graph-based methods for partitioning of soft clauses for partial MaxSAT.

4 Partial MaxSAT Partitioning

It has already been shown that partitioning can greatly boost the performance of unsatisfiability-based solvers for weighted MaxSAT [11,2]. However, using the weight of soft clauses for partitioning is not useful for partial MaxSAT. As a result, other methods based on the structure of the formula must be used. Next, we briefly review the hypergraph partitioning method first proposed for weighted MaxSAT. Afterwards, we propose a new partition method for MaxSAT based on a graph representation of the MaxSAT formula. The new partition method uses the graph representation to identify communities using a modularity measure. The methods described in this section represent different implementations of the `partitionSoft` procedure in line 4 of Algorithm 1.

4.1 Hypergraph Partitioning

Hypergraph partitioning has already been applied to SAT [12], as well as to weighted MaxSAT solving [2]. A hypergraph is a generalization of a graph where an edge, also called hyperedge, can connect any number of vertices. In our case, for each soft and hard clause there is a corresponding vertex in the hypergraph. Moreover, for each formula variable x_j there is an hyperedge connecting all vertices that represent soft or hard clauses containing variable x_j.

After building the hypergraph, the tool `hmetis` [13] is used as a black box to identify the partitions. In the experiments, `hmetis` is configured to identify 16 partitions in each problem instance [2]. Afterwards, for each partition only the soft clauses are considered. As a result, partitions containing only hard clauses are removed.

4.2 Community-based Partitioning

The identification of communities in SAT instances has been previously proposed [14] and it was shown to be effective in characterizing industrial SAT instances. For that, SAT instances are first represented as undirected weighted graphs and partitions of vertices (communities) are identified using a modularity measure. In this paper we use both graph representations described in [14] for SAT instances, namely the Variable Incidence Graph (VIG) and the Clause-Variable Incidence Graph (CVIG) model. In addition, a different weighting function is proposed in this paper.

Graph Representations

We start by defining an incidence function I on the formula variables x_j in the soft clauses φ_s as follows:

$$I(x_j) = 1 + \sum_{x_j \in \omega \ \wedge \ \omega \in \varphi_s} \frac{1}{|\omega|} \tag{1}$$

Notice that $I(x_j) = 1$ if variable x_j does not occur in any soft clause.

In the Variable Incidence Graph (VIG) model, a graph G is built such that for each variable x_j in the problem instance there is a corresponding vertex in G. Moreover, if x_j and x_k belong to the same clause (hard or soft), then there is an edge between the vertices corresponding to these variables with the following weight:

$$w(x_j, x_k) = \sum_{x_j, x_k \in \omega \ \wedge \ \omega \in \varphi} \frac{I(x_j) \cdot I(x_k)}{\binom{|\omega|}{2}} \tag{2}$$

Observe that if we consider $I(x_j) = 1$ for all variables, then this weight function corresponds to the one proposed in [14], where all clauses are equally relevant. However, for MaxSAT one has to consider both soft and hard clauses. In our graph representation, more weight is given to clauses that establish edges between variables that occur in soft clauses. The motivation is to fortify the relationship between variables that occur in soft clauses.

In the Clause-Variable Incidence Graph (CVIG) model, for each variable x_j and for each clause $\omega_i \in \varphi$, there is a corresponding vertex in graph G. In this model, edges only connect vertices representing a variable and a clause where the variable occurs. Hence, if a variable x_j occurs in clause ω_i, then there is an edge between those vertices with weight:

$$w(x_j, \omega_i) = \frac{I(x_j)}{|\omega_i|} \tag{3}$$

Community Identification

After building a graph representation for the problem instance, we are interested in making explicit the hidden structure of the MaxSAT formula by identifying partitions in the graph. Clearly one can devise many different ways of partitioning. Therefore, it is necessary to evaluate the quality of a given set of partitions.

In recent years, the use of modularity measures became common for the identification of communities in graphs [15,16,17]. The main goal of the modularity measure is to evaluate the quality of the communities in a graph where vertices inside a community should be densely connected, while vertices assigned to different communities should be sparsely connected. Let $G = (V, w)$ denote a complete undirected weighted graph where V is the set of vertices and $w : V \times V \rightarrow \mathbb{R}$ is a weight function for each pair of vertices. If an edge does not occur in G, then it has weight 0. Let $C = \{C_1, C_2, \ldots, C_n\}$ denote a set of communities such that every vertex $u \in V$ is assigned to one and only one community in C. Hence, the modularity value Q of the set of communities C in

graph G can be defined as follows [16]:

$$Q = \sum_{C_k \in C} \left[\frac{\sum\limits_{i,j \in C_k} w(i,j)}{m} - \left(\frac{\sum\limits_{i \in C_k} \sum\limits_{j \in V} w(i,j)}{2m} \right)^2 \right] \tag{4}$$

where $m = \sum_{i,j \in V} w(i,j)$ denotes the sum of the weights of all the edges in G.

One drawback of community identification using modularity measures is that finding a set of communities with an optimal modularity value is computationally hard [18]. As a result, several approximation algorithms have been proposed [19,20,21]. In this paper, the method proposed in [21] is used.

From Communities to Partitions

After identifying the communities in the graph, one must build the set of partitions to be used in Algorithm 1. When using the CVIG model, building partitions of soft clauses is straightforward since clauses are directly represented in the graph. For each community with vertices representing soft clauses, there is a corresponding partition containing the respective soft clauses in the community. After building the partitions, these are sorted by ascending size with respect to the number of soft clauses. Therefore, partitions with smaller size are considered first in Algorithm 1.

In the VIG model, only variables are represented in the graph. Therefore, given the set of communities C, we define that a soft clause ω belongs to the community C_k that maximizes $|C_k \cap \omega|$, i.e. C_k is the community with the most variables in ω. In case of a tie, ω is assigned to the community of the lowest index variable in ω. After assigning all soft clauses to communities, partitions to be used in Algorithm 1 are built as in the CVIG model.

5 Experimental Results

All experiments were run on the partial MaxSAT instances from the industrial category of the MaxSAT evaluation of 2012. The evaluation was performed on two AMD Opteron 6276 processors (2.3 GHz) running Fedora 18 with a timeout of 1,800 seconds and a memory limit of 16 GB.

The partitioning techniques described in the previous section were implemented on top of WBO [5]. Figure 1 compares the different partitioning solvers against the original WBO that does not use any partitioning techniques. The solver using hypergraph partitioning is denoted by *hyp*. *VIG* and *CVIG* correspond to the community-based partitioning using the VIG and CVIG graph representations described in section 4.2, respectively. To assess the quality of the new partitions, we have also implemented a random partitioning technique where each soft clause is placed randomly in one of the partitions. We denote this solver by *rdm*. Similarly to the hypergraph partitioning, 16 partitions are used. The clauses are distributed uniformly among the partitions.

The table on the left of Figure 1 shows the number of instances solved in each benchmark set by each solver. Randomly partitioning the soft clauses can have a detrimental effect on the performance of the solver for several classes of benchmarks. However, it can also significantly improve the performance of the solver on other classes of

Bench	#I	WBO	hyp	VIG	CVIG	rdm
aes	7	0	0	0	0	0
fir	59	40	26	28	29	18
simp	17	10	9	10	10	9
su	38	11	6	11	10	6
msp	64	4	5	4	6	1
mtg	40	18	39	38	36	19
syn	74	32	31	35	32	30
circuit	4	1	2	2	2	2
haplotype	6	5	5	5	5	5
nencdr	84	19	67	68	68	66
nlogencdr	84	24	71	70	75	68
routing	15	15	6	15	6	3
protein	12	1	2	1	1	2
Total	504	180	269	287	280	229

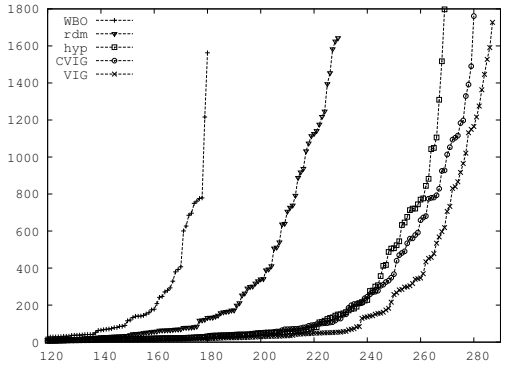

Fig. 1. Comparison between different partitioning solvers.

benchmarks, such as `nencdr` and `nlogencdr`. Nevertheless, other partition methods based on structural information of the formula are clearly better.

Community-based partitioning outperforms hypergraph partitioning. Note that hypergraph partitioning creates a fixed number of partitions. However, in community-based partitioning, the number of partitions is dynamic and depends on the structure of the formula. This may explain the effectiveness of community-based partitioning.

The cactus plot of Figure 1 results from the running times of the different solvers. The x-axis shows the number of solved instances, whereas the y-axis shows the running time in seconds. We can distinguish between three classes of solvers: (i) solvers that do not use partitioning (WBO), (ii) solvers that use random partitioning (rdm) and (iii) solvers that use the structure of the formula to create the partitions (hyp, CVIG and VIG). Even random partitioning improves the overall performance of the solver. However, when the structure of the formula is considered, the performance of the solver is further improved.

Even though partitioning approaches outperform WBO on most benchmarks, there are some benchmarks where partitioning may lead to a detrimental effect on the performance of the solver. Our motivation for partitioning is to identify easier to solve subproblems. As a side effect, this may lead to finding smaller unsatisfiable subformulas at each call of the SAT solver. On average, WBO finds unsatisfiable subformulas with 110 soft clauses, whereas unsatisfiable subformulas in VIG have 66 soft clauses. The other solvers using partitions also behave similarly and find on average smaller unsatisfiable subformulas than WBO. This behavior is particularly visible on the `nencdr` and `nloencdr` benchmarks [22]. For these benchmarks, WBO finds on average unsatisfiable subformulas with 167 soft clauses, whereas unsatisfiable subformulas in VIG have only 47 soft clauses.

However, if the partitions are not adequate, then we may split *related* soft clauses between different partitions. This may prevent the solver from finding small unsatisfiable subformulas. For example, this behavior is observed in the `routing` benchmarks [23]. On average, WBO finds unsatisfiable subformulas with 7 soft clauses, whereas CVIG finds unsatisfiable subformulas with 45 soft clauses. Due to an inadequate partitioning,

CVIG is only able to solve 6 out of 15 instances of `routing`. Note that VIG can solve all `routing` instances since the partitions used allowed to find on average unsatisfiable subformulas with 9 soft clauses. Example 1 shows the impact that inadequate partitioning may have on the performance of the solver.

Example 1. Consider a partial MaxSAT formula $\varphi = \varphi_h \cup \varphi_s$. Assume that φ_h contains the hard clause $\omega_1 = (x_1 \vee x_2 \vee x_3)$ and that φ_s contains the soft clauses ω_2, ω_3 and ω_4, where $\omega_2 = (\bar{x}_1), \omega_3 = (\bar{x}_2)$ and $\omega_4 = (\bar{x}_3)$. If these soft clauses are placed in the same partition then we can find a trivial unsatisfiable subformula with the hard clause ω_1. However, consider the worst case scenario where ω_2, ω_3 and ω_4 are placed in different partitions with other soft clauses. Let us assume that we first try to find unsatisfiable subformulas in the partition that contains ω_2. In this case, we may find unsatisfiable subformulas formed by ω_2 with other soft clauses. Note that each time a new unsatisfiable subformula φ_C is found, all soft clauses in φ_C are relaxed. Therefore, after several iterations, when the working formula finally contains ω_2, ω_3 and ω_4, we may have already relaxed these soft clauses several times. If this is the case, then we will no longer be able to find the small unsatisfiable subformula that could be identified if no partitioning was used.

It was observed that the inadequate partitioning presented in Example 1 occurs frequently in some classes of benchmarks, such as `fir` and `routing`. Moreover, if a random partitioning is used, then this problem is even more accentuated. This may explain why random partitioning deteriorates the performance of the solver for several benchmark sets. On the other hand, this shows that an adequate partitioning of the formula is essential for the effectiveness of the solver. Future work will focus on improving partitioning techniques to further reduce the probability of inadequate partitioning.

WBO is a state-of-the-art solver for weighted partial MaxSAT but is not as effective for partial MaxSAT. Even though partitioning significantly improves the unsatisfiability-based algorithm of WBO, it is still not enough to match the performance of state-of-the-art solvers for partial MaxSAT. However, the partitioning approaches presented in this paper are not limited to the unsatisfiability-based algorithm of WBO but can also be extended to other unsatisfiability-based algorithms [4,7]. As future work, we propose to implement the partitioning techniques described in this paper on top of other unsatisfiability-based algorithms for partial MaxSAT.

6 Conclusions

Partitioning the soft clauses has shown to significantly improve the unsatisfiability-based algorithm of WBO for most classes of benchmarks. Moreover, if the structure of the formula is taken into consideration when creating the partitions we can further improve the effectiveness of the solver. This supports the idea that using the structure of the formula to guide the search improves the performance of the solver and provides a strong stimulus for future research.

As future work, we propose to extend our modularity-based partitioning for weighted MaxSAT. Furthermore, the partitioning approaches proposed in this paper are not limited to the WBO algorithm and will be used in other unsatisfiability-based algorithms.

References

1. Coudert, O.: On Solving Covering Problems. In: Proceedings of the Design Automation Conference. (June 1996) 197–202
2. Martins, R., Manquinho, V., Lynce, I.: On Partitioning for Maximum Satisfiability. In: European Conference on Artificial Intelligence. (2012) 913–914
3. Fu, Z., Malik, S.: On Solving the Partial MAX-SAT Problem. In: International Conference on Theory and Applications of Satisfiability Testing. (2006) 252–265
4. Ansótegui, C., Bonet, M., Levy, J.: Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In: International Conference on Theory and Applications of Satisfiability Testing. (2009) 427–440
5. Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for Weighted Boolean Optimization. In: International Conference on Theory and Applications of Satisfiability Testing. (2009) 495–508
6. Ansótegui, C., Bonet, M., Levy, J.: A New Algorithm for Weighted Partial MaxSAT. In: AAAI Conference on Artificial Intelligence. (2010) 3–8
7. Heras, F., Morgado, A., Marques-Silva, J.: Core-Guided Binary Search Algorithms for Maximum Satisfiability. In: AAAI Conference on Artificial Intelligence. (2011) 36–41
8. Janota, M., Lynce, I., Manquinho, V., Marques-Silva, J.: PackUp: Tools for Package Upgradability Solving. Journal on Satisfiability, Boolean Modeling and Computation **8**(1/2) (2012) 89–94
9. Li, C.M., Manyà, F.: MaxSAT, Hard and Soft Constraints. In: Handbook of Satisfiability. IOS Press (2009) 613–631
10. Koshimura, M., Zhang, T., Fujita, H., Hasegawa, R.: QMaxSAT: A Partial Max-SAT Solver. Journal on Satisfiability, Boolean Modeling and Computation **8** (2012) 95–100
11. Ansótegui, C., Bonet, M., Gabàs, J., Levy, J.: Improving SAT-Based Weighted MaxSAT Solvers. In: International Conference on Principles and Practice of Constraint Programming. (2012) 86–101
12. Park, T.J., Gelder, A.V.: Partitioning Methods for Satisfiability Testing on Large Formulas. Information and Computation **162**(1-2) (2000) 179184
13. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: Application in VLSI domain. In: IEEE Transactions on VLSI Systems. Volume 7. (1999) 69–79
14. Ansótegui, C., Giráldez-Cru, J., Levy, J.: The Community Structure of SAT Formulas. In: International Conference on Theory and Applications of Satisfiability Testing. (2012) 410–423
15. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12) (2002) 7821–7826
16. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69**(026113) (2004)
17. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences of the United States of America **101**(9) (2004) 2658–2663
18. Brandes, U., Delling, D., Gaertler, M., Goerke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: Maximizing modularity is hard. arXiv: physics, 0608255. (2006)
19. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E **70**(6) (2004) 066111
20. Pons, P., Latapy, M.: Computing Communities in Large Networks Using Random Walks. Journal of Graph Algorithms and Applications **10**(2) (2006) 191–218
21. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics **2008**(10) (2008) P10008

22. Morgado, A., Marques-Silva, J.: Combinatorial Optimization Solutions for the Maximum Quartet Consistency Problem. Fundamenta Informaticae **102**(3-4) (2010) 363–389
23. Aloul, F.A., Ramani, A., Markov, I.L., Sakallah, K.A.: PBS: A backtrack search pseudo Boolean solver. In: In Symposium on the theory and applications of satisfiability testing. (2002) 346–353