Symmetry Breaking for Maximum Satisfiability[⋆]

Joao Marques-Silva[1], Inês Lynce[2], and Vasco Manquinho[2]

[1] School of Electronics and Computer Science, University of Southampton, UK
jpms@ecs.soton.ac.uk
[2] IST/INESC-ID, Technical University of Lisbon, Portugal
{ines,vmm}@sat.inesc-id.pt

Abstract. Symmetries are intrinsic to many combinatorial problems including Boolean Satisfiability (SAT) and Constraint Programming (CP). In SAT, the identification of symmetry breaking predicates (SBPs) is a well-known, often effective, technique for solving hard problems. The identification of SBPs in SAT has been the subject of significant improvements in recent years, resulting in more compact SBPs and more effective algorithms. The identification of SBPs has also been applied to pseudo-Boolean (PB) constraints, showing that symmetry breaking can also be an effective technique for PB constraints. This paper extends further the application of SBPs, and shows that SBPs can be identified and used in Maximum Satisfiability (MaxSAT), as well as in its most well-known variants, including partial MaxSAT, weighted MaxSAT and weighted partial MaxSAT. As with SAT and PB, symmetry breaking predicates for MaxSAT and variants are shown to be effective for a representative number of problem domains, allowing solving problem instances that current state of the art MaxSAT solvers could not otherwise solve.

1 Introduction

Symmetry breaking is a widely used technique for solving combinatorial problems. Symmetries have been used with great success in Satisfiability (SAT) [6, 1], and are regarded as an essential technique for solving specific classes of problem instances. Symmetries have also been widely used for solving constraint satisfaction problems (CSPs) [8]. More recent work has shown how to apply symmetry breaking in pseudo-Boolean (PB) constraints [2] and also in soft constraints [18]. It should be noted that symmetry breaking is viewed as an effective problem solving technique, either for SAT, PB or CP, that is often used as an alternative technique, to be applied when default algorithms are unable to solve a given problem instance.

In recent years there has been a growing interest in algorithms for MaxSAT and variants [12, 13, 20, 10, 11, 14], in part because of the wide range of potential applications. MaxSAT and variants represent a more general framework than either SAT or PB, and so can naturally be used in many practical applications. The interest in MaxSAT and variants motivated the development of a new generation of MaxSAT algorithms, remarkably more efficient than early MaxSAT algorithms [19, 4]. Despite the observed improvements, there are many problems still too complex for MaxSAT algorithms to

[⋆] This paper is also available as reference [15].

solve [3]. Natural lines of research for improving MaxSAT algorithms include studying techniques known to be effective for either SAT, PB or CP. One concrete example is symmetry breaking. Despite its success in SAT, PB and CP, the usefulness of symmetry breaking for MaxSAT and variants has not been thoroughly studied.

This paper addresses the problem of using symmetry breaking in MaxSAT and in its most well-known variants, partial MaxSAT, weighted MaxSAT and weighted partial MaxSAT. The work extends past recent work on computing symmetries for SAT [1] and PB constraints [2] by computing automorphism on colored graphs obtained from CNF or PB formulas, and by showing how symmetry breaking predicates [6, 1] can be exploited. The experimental results show that symmetry breaking is an effective technique for MaxSAT and variants, allowing solving problem instances that state of the art MaxSAT solvers could not otherwise solve.

The paper is organized as follows. The next section introduces the notation used throughout the paper, provides a brief overview of MaxSAT and variants, and also summarizes the work on symmetry breaking for SAT and PB constraints. Afterwards, the paper describes how to apply symmetry breaking in MaxSAT and variants. Experimental results, obtained on representative problem instances from the MaxSAT evaluation [3] and also from practical applications [1], demonstrate that symmetry breaking allows solving problem instances that could not be solved by *any* of the available state of the art MaxSAT solvers. The paper concludes by summarizing related work, by overviewing the main contributions, and by outlining directions for future work.

2 Preliminaries

This section introduces the notation used through the paper, as well as the MaxSAT problem and its variants. An overview of symmetry identification and symmetry breaking is also presented.

2.1 Maximum Satisfiability

The paper assumes the usual definitions for SAT. A propositional formula is represented in *Conjunctive Normal Form* (CNF). A CNF formula φ consists of a conjunction of clauses, where each clause ω is a disjunction of literals, and a literal l is either a propositional variable x or its complement $\bar{x}$. Variables can be assigned a propositional value, either 0 or 1. A literal $l_j = x_j$ assumes value 1 if $x_j = 1$ and assumes value 0 if $x_j = 0$. Conversely, literal $l_j = \bar{x}_j$ assumes value 1 if $x_j = 0$ and value 0 when $x_j = 1$. For each assignment of values to the variables, the value of formula φ is computed with the rules of propositional logic. A clause is said to be *satisfied* if at least one of its literals assumes value 1. If all literals of a clause assume value 0, then the clause is *unsatisfied*. The propositional satisfiability (SAT) problem consists in deciding whether there exists an assignment to the variables such that φ is satisfied.

Given a propositional formula φ, the MaxSAT problem is defined as finding an assignment to variables in φ such that the number of satisfied clauses is maximized. (MaxSAT can also be defined as finding an assignment that minimizes the number of unsatisfied clauses.) Well-known variants of MaxSAT include partial MaxSAT, weighted MaxSAT and weighted partial MaxSAT.

For partial MaxSAT, a propositional formula φ is described by the conjunction of two CNF formulas φ_s and φ_h, where φ_s represents the *soft* clauses and φ_h represents the *hard* clauses. The partial MaxSAT problem over a propositional formula $\varphi = \varphi_h \wedge \varphi_s$ consists in finding an assignment to the problem variables such that all hard clauses (φ_h) are satisfied and the number of satisfied soft clauses (φ_s) is maximized.

For *weighted* MaxSAT, each clause in the CNF formula is associated to a non-negative weight. A weighted clause is a pair (ω, c) where ω is a classical clause and c is a natural number corresponding to the cost of unsatisfying ω. Given a weighted CNF formula φ, the *weighted* MaxSAT problem consists in finding an assignment to problem variables such that the total weight of the unsatisfied clauses is minimized, which implies that the total weight of the satisfied clauses is maximized. For the *weighted partial* MaxSAT problem, the formula is the conjunction of a weighted CNF formula (soft clauses) and a classical CNF formula (hard clauses). The weighted partial MaxSAT problem consists in finding an assignment to the variables such that all hard clauses are satisfied and the total weight of satisfied soft clauses is maximized. Observe that, for both partial MaxSAT and weighted partial MaxSAT, hard clauses can be represented as weighted clauses. For these clauses one can consider that the weight is greater than the sum of the weights of the soft clauses.

MaxSAT and variants find a wide range of practical applications, that include scheduling, routing, bioinformatics, and design automation. Moreover, MaxSAT can be used for solving pseudo-Boolean optimization [11]. The practical applications of MaxSAT motivated recent interest in developing more efficient algorithms. The most efficient algorithms for MaxSAT and variants are based on branch and bound search, using dedicated bounding and inference techniques [12, 13, 10, 11]. Lower bounding techniques include for example the use of unit propagation for identifying necessarily unsatisfied clauses, whereas inference techniques can be viewed as restricted forms of resolution, with the objective of simplifying the problem instance to solve.

2.2 Symmetry Breaking

Given a problem instance, a symmetry is an operation that preserves the constraints, and therefore also preserves the solutions [5]. For a set of symmetric states, it is possible to obtain the whole set of states from any of the states. Hence, symmetry breaking predicates may eliminate all but one of the equivalent states. Symmetry breaking is expected to speed up the search as the search space gets reduced. For specific problems where symmetries may be easily found this reduction may be significant. Nonetheless, the elimination of symmetries necessarily introduces overhead that is expected to be negligible when compared with the benefits it may provide.

The elimination of symmetries has been extensively studied in CP and SAT [16, 6]. The most well-know strategy for eliminating symmetries in SAT consists in adding symmetry breaking predicates (SBPs) to the CNF formula [6]. SBPs are added to the formula before the search starts. The symmetries may be identified for each specific problem, and in that case it is required that the symmetries in the problem are identified when creating the encoding. Alternatively, one may give a formula to a specialized tool for detecting all the symmetries [1]. The resulting SBPs are intended to merge symmetries in equivalent classes. In case all symmetries are broken, only one assignment, instead of n assignments, may satisfy a set of constraints, being n the number of elements in a given equivalent class.

Other approaches include remodeling the problem [17] and breaking symmetries during search [9]. Remodeling the problem implies creating a different encoding, e.g. obtained by defining a different set of variables, in order to create a problem with less symmetries or even none at all. Alternatively, the search procedure may be adapted for adding SBPs as the search proceeds to ensure that any assignment symmetric to one assignment already considered will not be explored in the future, or by performing checks that symmetric equivalent assignments have not yet been visited.

Currently available tools for detecting and breaking symmetries for a given formula are based on group theory. From each formula a group is extracted, where a group is a set of permutations. A permutation is a one-to-one correspondence between a set and itself. Each symmetry defines a permutation on a set of literals. In practice, each permutation is represented by a product of disjoint cycles. Each cycle $(l_1\, l_2\, \ldots\, l_m)$ with size m stands for the permutation that maps l_i on l_{i+1} (with $1 \leq i \leq m-1$) and l_m on l_1. Applying a permutation to a formula will produce exactly the same formula.

Example 1. Consider the following CNF formula:

$$\varphi = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2) \wedge (x_3 \vee x_2) \wedge (\bar{x}_3 \vee x_2)$$

The permutations identified for φ are $(x_3\ \bar{x}_3)$ and $(x_1\ x_3)(\bar{x}_1\ \bar{x}_3)$. (The permutation $(x_1\ \bar{x}_1)$ is implicit.) The formula resulting from the permutation $(x_3\ \bar{x}_3)$ is obtained by replacing every occurrence of x_3 by $\bar{x}_3$ and every occurrence of $\bar{x}_3$ by x_3. Clearly, the obtained formula is equal to the original formula. The same happens when applying the permutation $(x_1\ x_3)(\bar{x}_1\ \bar{x}_3)$: replacing x_1 by x_3, x_3 by x_1, $\bar{x}_1$ by $\bar{x}_3$ and $\bar{x}_3$ by $\bar{x}_1$ produces the same formula.

3 Symmetry Breaking for MaxSAT

This section describes how to apply symmetry breaking in MaxSAT. First, the construction process for the graph representing a CNF formula is briefly reviewed [6, 1], as it will be modified later in this section. Afterwards, plain MaxSAT is considered. The next step is to address partial, weighted and weighted partial MaxSAT.

3.1 From CNF Formulas to Colored Graphs

Symmetry breaking for MaxSAT and variants requires a few modifications to the approach used for SAT [6, 1]. This section summarizes the basic approach, which is then extended in the following sections.

Given a graph, the *graph automorphism* problem consists in finding isomorphic groups of edges and vertices with a one-to-one correspondence. In case of graphs with colored vertices, the correspondence is made between vertices with the same color. It is well-known that symmetries in SAT can be identified by reduction to a graph automorphism problem [6, 1]. The propositional formula is represented as an undirected

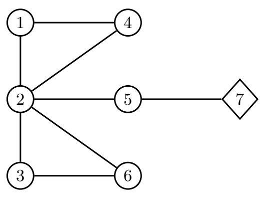

Fig. 1. Colored graph for Example 2

graph with colored vertices, such that the automorphism in the graph corresponds to a symmetry in the propositional formula.

Given a propositional formula φ, a colored undirected graph is created as follows:

- For each variable $x_j \in \varphi$ add two vertices to represent x_j and $\bar{x}_j$. All vertices associated with variables are colored with color 1;
- For each variable $x_j \in \varphi$ add an edge between the vertices representing x_j and $\bar{x}_j$;
- For each binary clause $\omega_i = (l_j \vee l_k) \in \varphi$, add an edge between the vertices representing l_j and l_k;
- For each non-binary clause $\omega_i \in \varphi$ create a vertex colored with 2;
- For each literal l_j in a non-binary clause ω_i, add an edge between the corresponding vertices.

Example 2. Figure 1 shows the colored undirected graph associated with the CNF formula of Example 1. Vertices with shape $\circ$ represent color 1 and vertices with shape $\diamond$ represent color 2. Vertex 1 corresponds to x_1, 2 to x_2, 3 to x_3, 4 to $\bar{x}_1$, 5 to $\bar{x}_2$, 6 to $\bar{x}_3$ and 7 to unit clause $(\bar{x}_2)$. Edges 1-2, 2-3, 2-4 and 2-6 represent binary clauses and edges 1-4, 2-5 and 3-6 link complemented literals.

3.2 Plain Maximum Satisfiability

Let φ represent the CNF formula of a MaxSAT instance. Moreover, let φ_{sbp} be the CNF formula for the symmetry-breaking predicates obtained with a CNF symmetry tool (e.g. Shatter [3]). All clauses in φ are effectively *soft* clauses, for which the objective is to maximize the number of satisfied clauses. In contrast, the clauses in φ_{sbp} are *hard* clauses, which must necessarily be satisfied. As a result, the original MaxSAT problem is transformed into a partial MaxSAT problem, where φ denotes the soft clauses and φ_{sbp} denotes the hard clauses. The solution of the partial MaxSAT problem corresponds to the solution of the original MaxSAT problem.

Example 3. For the CNF formula of Example 1, the generated SBP predicates (by Shatter) are: $\varphi_{sbp} = (\bar{x}_3) \wedge (\bar{x}_1 \vee x_3)$ As result, the resulting instance of partial MaxSAT will be $\varphi' = (\varphi_h \wedge \varphi_s) = (\varphi_{sbp} \wedge \varphi)$. Moreover, $x_3 = 0$ and $x_1 = 0$ are necessary assignments, and so variables x_1 and x_3 can be ignored for maximizing the number of satisfied soft clauses.

[3] http://www.eecs.umich.edu/~faloul/Tools/shatter/

Observe that the hard clauses represented by φ_{sbp} do not change the solution of the original MaxSAT problem. Indeed, the construction of the symmetry breaking predicates guarantees that the maximum number of satisfied soft clauses remains unchanged by the addition of the hard clauses.

Proposition 1. *The maximum number of satisfied clauses for the MaxSAT problem φ and the partial MaxSAT problem $(\varphi \wedge \varphi_{sbp})$ are the same.*

Proof: *(Sketch) The proof follows from the fact that symmetries map models into models and non-models into non-models (see Proposition 2.1 in [6]). Consider the clauses as an ordered sequence $\langle\omega_1, \ldots, \omega_m\rangle$. Given a symmetry, a clause in position i will be mapped to a clause in another position j. Now, given any assignment, if the clause in position i is satisfied (unsatisfied), then by applying the symmetry, the clause in position j is now satisfied (unsatisfied). Thus the number of satisfied (unsatisfied) clauses is unchanged.* ∎

3.3 Partial and Weighted Maximum Satisfiability

For partial MaxSAT, the generation of SBPs needs to be modified. The graph representation of the CNF formula must take into account the existence of hard and soft clauses, which must be distinguished by a graph automorphism algorithm. Symmetric states for problem instances with hard and soft clauses establish a correspondence either between hard clauses or between soft clauses. In other words, when applying a permutation hard clauses can only be replaced by other hard clauses, and soft clauses by other soft clauses. In order to address this issue, the colored graph generation needs to be modified. In contrast to the MaxSAT case, binary clauses are not distinguished from other clauses, and are represented as vertices in the colored graph. Clauses can now have one of two colors. A vertex with color 2 is associated with each soft clause, and a vertex with color 3 is associated with each hard clause. This modification ensures that any identified automorphism guarantees that soft clauses correspond only to soft clauses, and hard clauses correspond only to hard clauses. Moreover, the procedure for the generation of SBPs from the groups found by a graph automorphism tool remains unchanged, and the SBPs can be added to the original instance as *new* hard clauses. The resulting instance is also an instance of partial MaxSAT. Correctness of this approach follows form the correctness of the plain MaxSAT case.

The solution for weighted MaxSAT and for weighted partial MaxSAT is similar to the partial MaxSAT case, but now clauses with different weights are represented by vertices with different colors. This guarantees that the groups found by the graph automorphism tool take into consideration the weight of each clause. Let $\{c_1, c_2, \ldots, c_k\}$ denote the distinct clause weights in the CNF formula. Each clause of weight c_i is associated with a vertex of color $i + 1$ in the colored graph. In case there exist hard clauses, an additional color $k + 2$ is used, and so each hard clause is represented by a vertex with color $k + 2$ in the colored graph. Associating distinct clause weights with distinct colors guarantees that the graph automorphism algorithm can only make the correspondence between clauses with the same weight. Moreover, the identified SBPs result in new *hard* clauses that are added to the original problem. For either weighted MaxSAT

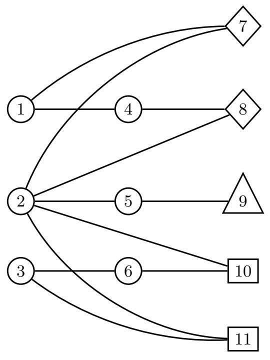

Fig. 2. Colored graph for Example 4

Table 1. Problem transformations due to SBPs

Original	With Symmetries
MS	PMS
PMS	PMS
WMS	WPMS
WPMS	WPMS

or weighted partial MaxSAT, the result is an instance of weighted partial MaxSAT. As before, correctness of this approach follows form the correctness of the plain MaxSAT case.

Example 4. Consider the following weighted partial MaxSAT instance:

$$\varphi = (x_1 \vee x_2, 1) \wedge (\bar{x}_1 \vee x_2, 1) \wedge (\bar{x}_2, 5) \wedge \\ (\bar{x}_3 \vee x_2, 9) \wedge (x_3 \vee x_2, 9)$$

for which the last two clauses are hard. Figure 2 shows the colored undirected graph associated with the formula. Clauses with different weights are represented with different colors (shown in the figure with different vertex shapes). A graph automorphism algorithm can then be used to generate the symmetry breaking predicates $\varphi_{sbp} = (\bar{x}_1) \wedge (\bar{x}_3)$, consisting of two hard clauses. As a result, the assignments $x_1 = 0$ and $x_3 = 0$ become necessary.

Table 1 summarizes the problem transformations described in this section, where MS represents plain MaxSAT, PMS represents partial MaxSAT, WMS represents weighted MaxSAT, and WPMS represents weighted partial MaxSAT. The use of SBPs introduces a number of hard clauses, and so the resulting problems are either partial MaxSAT or weighted partial MaxSAT.

4 Experimental Results

The experimental setup has been organized as follows. First, all the instances from the first and second MaxSAT evaluations (2006 and 2007) [3] were run. These results allowed selecting relevant benchmark families, for which symmetries occur and which require a non-negligible amount of time for being solved by both approaches (with or without SBPs). Afterwards, the instances for which both approaches aborted were removed from the tables of results. This resulted in selecting the `hamming` and the `MANN` instances for plain MaxSAT, the `ii32` and again the `MANN` instances for partial MaxSAT, the `c-fat500` instances for weighted MaxSAT and the `dir` and `log` instances for weighted partial MaxSAT.

Besides the instances that participated in the MaxSAT competition, we have included additional problem instances (`hole`, `Urq` and `chnl`). The `hole` instances refer to the well-known pigeon hole problem, the `Urq` instances represent randomized instances based on expander graphs and the `chnl` instances model the routing of wires in the channels of field-programmable integrated circuits. These instances refer to problems that can be naturally encoded as MaxSAT problems and are known to be highly symmetric [1]. The approach outlined above was also followed for selecting the instances to be included in the tables of results.

We have run different publicly available MaxSAT solvers, namely MINIMAXSAT [4], TOOLBAR [5] and MAXSATZ [6]. (MAXSATZ accepts only plain MaxSAT instances.) It has been observed that MINIMAXSAT behavior is similar to TOOLBAR and MAXSATZ, albeit being in general more robust. For this reason, the results focus on MINIMAXSAT.

Tables 2 and 3 provide the results obtained. Table 2 refers to plain MaxSAT instances and Table 3 refers to partial MaxSAT (PMS), weighted MaxSAT (WMS) and weighted partial MaxSAT (WPMS) instances. For each instance, the results shown include the number of clauses added as a result of SBPs (#ClsSbp), the time required for solving the original instances (OrigT), i.e. without SBPs, and the time required for breaking the symmetries plus the time required for solving the extended formula afterwards (SbpT). In practice, the time required for generating SBPs is negligible. The results were obtained on a Intel Xeon 5160 server (3.0GHz, 1333Mhz, 4MB) running Red Hat Enterprise Linux WS 4.

The experimental results allow establishing the following conclusions:

– The inclusion of symmetry breaking is *essential* for solving a number of problem instances. We should note that *all* the plain MaxSAT instances in Table 2 for which MINIMAXSAT aborted, are also aborted by TOOLBAR and MAXSATZ. After adding SBPs all these instances become easy to solve by any of the solvers. For the aborted partial, weighted and weighted partial MaxSAT instances in Table 3 this is not always the case, since a few instances aborted by MINIMAXSAT could be solved by TOOLBAR without SBPs. However, the converse is also true, as there are instances that were initially aborted by TOOLBAR (although solved by MINIMAXSAT) that are solved by TOOLBAR after adding SBPs.

[4] http://www.lsi.upc.edu/∼fheras/docs/m.tar.gz
[5] http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/ToolBarIntro
[6] http://www.laria.u-picardie.fr/∼cli/maxsatz.tar.gz

Table 2. Results for MINIMAXSAT on plain MaxSAT instances

Name	#ClsSbp	OrigT	SbpT
hamming10-2	81	1000	0.19
hamming10-4	1	886.57	496.79
hamming6-4	437	0.17	0.15
hamming8-2	85	1000	0.21
hamming8-4	253	0.36	0.11
MANN_a27	85	1000	0.24
MANN_a45	79	1000	0.20
MANN_a81	79	1000	0.19
hole10	758	42.11	0.24
hole11	922	510.90	0.47
hole12	1102	1000	1.78
hole7	362	0.10	0.11
hole8	478	0.40	0.13
hole9	610	3.68	0.17
Urq3_5	29	83.33	0.27
Urq4_5	43	1000	50.88
chnl10_11	1954	1000	41.79
chnl10_12	2142	1000	328.12
chnl11_12	2370	1000	420.19

– For several instances, breaking only a few symmetries can make the difference. We have observed that in some cases the symmetries are broken with unit clauses.
– Adding SBPs is beneficial for most cases where symmetries exist. However, for a few examples, SBPs may degrade performance.
– There is no clear relation between the number of SBPs added and the impact on the search time.

Overall, the inclusion of SBPs should be considered when a hard problem instance is known to exhibit symmetries. This does not necessarily imply that after breaking symmetries the instance becomes trivial to solve, and there can be cases where the new clauses may degrade performance. However, in a significant number of cases, highly symmetric problems become much easier to solve after adding SBPs. In many of these cases the problem instances become *trivial* to solve.

5 Related Work

Symmetries are a well-known research topic, that serve to tackle complexity in many combinatorial problems. The first ideas on symmetry breaking were developed in the 90s [16, 6], by relating symmetries with the graph automorphism problem, and by proposing the first approach for generating symmetry breaking predicates. This work was later extended and optimized for propositional satisfiability [1].

Symmetries are an active research topic in CP [8]. Approaches for breaking symmetries include not only adding constraints before search [16] but also reformulation [17]

Table 3. Results for MINIMAXSAT on partial, weighted and weighted partial MaxSAT instances

Name	MStype	#ClsSbp	OrigT	SbpT
ii32e3	PMS	1756	94.40	37.63
ii32e4	PMS	2060	175.07	129.06
c-fat500-10	WMS	2	57.79	11.62
c-fat500-1	WMS	112	0.03	0.06
c-fat500-2	WMS	12	0.16	0.11
c-fat500-5	WMS	4	0.16	0.11
MANN_a27	WMS	1	1000	880.58
MANN_a45	WMS	1	1000	530.86
MANN_a81	WMS	1	1000	649.13
1502.dir	WPMS	1560	0.34	10.67
29.dir	WPMS	132	1000	28.09
54.dir	WPMS	98	4.14	0.32
8.dir	WPMS	58	0.03	0.05
1502.log	WPMS	812	0.76	0.71
29.log	WPMS	54	17.55	0.82
404.log	WPMS	124	1000	64.24
54.log	WPMS	48	2.37	0.16

and dynamic symmetry breaking methods [9]. Recent work has also shown the application of symmetries to soft CSPs [18].

The approach proposed in this paper for using symmetry breaking for MaxSAT and variants builds on earlier work on symmetry breaking for PB constraints [2]. Similarly to the work for PB constraints, symmetries are identified by constructing a colored graph, from which graph automorphisms are obtained, which are then used to generate the symmetry breaking predicates.

6 Conclusions

This paper shows how symmetry breaking can be used in MaxSAT and in its most well-known variants, including partial MaxSAT, weighted MaxSAT, and weighted partial MaxSAT. Experimental results, obtained on representative instances from the MaxSAT evaluation [3] and practical instances [1], demonstrate that symmetry breaking allows solving problem instances that no state of the art MaxSAT solver could otherwise solve. For all problem instances considered, the computational effort of computing symmetries is negligible. Nevertheless, and as is the case with related work for SAT and PB constraints, symmetry breaking should be considered as an alternative problem solving technique, to be used when standard techniques are unable to solve a given problem instance.

The experimental results motivate additional work on symmetry breaking for MaxSAT. The construction of the colored graph may be improved by focusing on possible relations among the different clause weights. Moreover, the use of conditional symmetries could be considered [7, 18].

Acknowledgement. This work is partially supported by EU grants IST/033709 and ICT/217069, by EPSRC grant EP/E012973/1 and by FCT grants POSC/EIA/61852/2004 and PTDC/EIA/76572/2006.

References

1. F. Aloul, K. A. Sakallah, and I. Markov. Efficient symmetry breaking for boolean satisfiability. In *International Joint Conference on Artificial Intelligence*, pages 271–276, August 2003.
2. F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. ShatterPB: symmetry-breaking for pseudo-Boolean formulas. In *Asian and South-Pacific Design Automation Conference*, pages 883–886, 2004.
3. J. Argelich, C. M. Li, F. Manyà, and J. Planes. MaxSAT evaluation. www.maxsat07.udl.es, May 2007.
4. B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2(4):299–306, 1998.
5. D. A. Cohen, P. Jeavons, C. Jefferson, K. E. Petrie, and B. M. Smith. Symmetry definitions for constraint satisfaction problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 17–31, 2005.
6. J. M. Crawford, M. L. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 148–159, 1996.
7. I. P. Gent, T. Kelsey, S. Linton, I. McDonald, I. Miguel, and B. M. Smith. Conditional symmetry breaking. In *International Conference on Principles and Practice of Constraint Programming*, pages 256–270, 2005.
8. I. P. Gent, K. E. Petrie, and J.-F. Puget. *Handbook of Constraint Programming*, chapter Symmetry in Constraint Programming, pages 329–376. Elsevier, 2006.
9. I. P. Gent and B. M. Smith. Symmetry breaking in constraint programming. In *European Conference on Artificial Intelligence*, pages 599–603, 2000.
10. F. Heras and J. Larrosa. New inference rules for efficient Max-SAT solving. In *AAAI Conference on Artificial Intelligence*, 2006.
11. F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: a new weighted Max-SAT solver. In *International Conference on Theory and Applications of Satisfiability Testing*, May 2007.
12. C. M. Li, F. Manyà, and J. Planes. Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In *International Conference on Principles and Practice of Constraint Programming*, pages 403–414, 2005.
13. C. M. Li, F. Manyà, and J. Planes. Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In *AAAI Conference on Artificial Intelligence*, July 2006.
14. 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.
15. J. Marques-Silva, I. Lynce, and V. Manquinho. Symmetry breaking for maximum satisfiability. Technical Report RT/039/08-CDIL, INESC-ID, February 2008.
16. J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In *International Symposium on Methodologies for Intelligent Systems*, pages 350–361, 1993.
17. B. M. Smith. Reducing symmetry in a combinatorial design problem. Technical Report 2001.01, School of Computing, University of Leeds, January 2001. Presented at the CP-AI-OR Workshop, April 2001.
18. B. M. Smith, S. Bistarelli, and B. O'Sullivan. Constraint symmetry for the soft CSP. In *International Conference on Principles and Practice of Constraint Programming*, pages 872–879, September 2007.

19. R. Wallace and E. Freuder. Comparative studies of constraint satisfaction and Davis-Putnam algorithms for maximum satisfiability problems. In D. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 587–615. American Mathematical Society, 1996.
20. Z. Xing and W. Zhang. MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability. *Artificial Intelligence*, 164(1-2):47–80, 2005.