Breaking Symmetries in SAT Matrix Models

Inês Lynce[1] and Joao Marques-Silva[2]

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

Abstract. Symmetry occurs naturally in many computational problems. The use of symmetry breaking techniques for solving search problems reduces the search space and therefore is expected to reduce the search time. Recent advances in breaking symmetries in SAT models are mainly focused on the identification of permutable variables via graph automorphism. These symmetries are denoted as instance-dependent, and although shown to be effective for different problem instances, the advantages of their generalised use in SAT are far from clear. Indeed, in many cases symmetry breaking predicates can introduce significant computational overhead, rendering ineffective the use of symmetry breaking. In contrast, in other domains, symmetry breaking is usually achieved by identifying instance-independent symmetries, often with promising experimental results. This paper studies the use of instance-independent symmetry breaking predicates in SAT. A concrete application is considered, and techniques for symmetry breaking in matrix models from CP are used. Our results indicate that instance-independent symmetry breaking predicates for matrix models can be significantly more effective than instance-dependent symmetry breaking predicates.

1 Introduction

In the recent past, symmetry breaking has been proposed as a technique that may be essential for solving hard computational problems. Indeed, successful results have been reported in different areas, including satisfiability (SAT), constraint programming (CP), planning and model checking. Nonetheless, whereas in most areas symmetries are broken according to specific properties of each problem instance, in Boolean satisfiability a more generic approach is often followed [1]. Instead of breaking symmetries when modelling a problem instance with SAT, generic symmetry breaking tools read a CNF formula and output the given formula extended with symmetry breaking clauses, which result from a graph automorphism analysis.

State-of-the-art SAT solvers are currently able to deal with very large formulae and to perform hundreds of thousands of propagations per second. Hence, one may think that augmenting the formula with symmetry breaking clauses in a preprocessing step does not represent a significant overhead to a SAT solver.

However, this is not the case for preprocessing techniques in general. Only specific techniques applied to specific problems have been shown to be effective.

On the other hand, mainly due to the effectiveness of SAT solvers learning techniques, modelling has not been much developed in SAT, at least when compared with other areas such as CP. Jointly with dynamic heuristics, learning is able to extend the formula in such a way that strategic resolution steps are performed. So, it is a reasonable approach to let the SAT solver learn *intelligently* rather than telling in advance what it should be able to learn during search. However, learning can hardly replace symmetry breaking predicates. For example, symmetry breaking predicates may reduce the number of solutions and learning does not. This paper compares the use of generalised CNF-based symmetry breaking predicates, also known as *instance-dependent predicates*, with the use of specific symmetry breaking predicates, i.e. *instance-independent predicates*, in the context of SAT matrix models[3].

2 Symmetry Breaking in SAT

The first complete framework suggesting a symmetry extraction mechanism for satisfiability based on a reduction to graph automorphism was proposed in [2]. This approach has been recently adapted and made practical for satisfiability in `shatter` [1]. For single variable permutations, `shatter` generates CNF formulae linear in the number of variables. In addition, `shatter` proposes a number of optimisations to the implementation of the graph automorphism algorithm. (Observe, however, that graph automorphism is believed not to be in P, even though it is not known whether it is NP-complete.)

The same authors have compared the efficiency of breaking *instance-dependent* symmetries against the efficiency of breaking *instance-independent* symmetries [7]. For the concrete problem of exact graph colouring, the use of *instance-dependent* symmetries is significantly more efficient. Instance-dependent symmetries are identified *automatically* via graph automorphism, whereas instance-independent symmetries are specific to the problem and are usually identified *manually* at the time the encoding is done. Before the existence of an efficient tool such as `shatter`, the generation of effective instance-independent symmetries was studied for several classes of combinatorial objects [8]. However, this approach was not evaluated against a generic one. Moreover, the use of symmetry breaking predicates in local search consistently has a negative effect in local search algorithms [6]. Interestingly, this observation has motivated an opposite strategy when applying local search: *maximising* symmetry in the SAT model.

3 Symmetry Breaking in Matrix Models

Symmetry in matrix models is usually broken by using lexicographic constraints [3]. If permutations in rows and/or columns can be made without affecting the ex-

[3] The paper follows the classification of predicates proposed in [1].

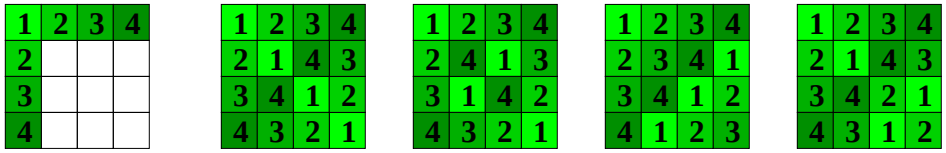

Fig. 1. A 4x4 Latin square with the first row and column fixed and its 4 solutions.

istence of solutions, then an ordering should be fixed to eliminate these symmetries. Although different orderings may be used, lexicographic ordering is considered to be the most intuitive. The resulting predicates are not guaranteed to eliminate all symmetries, since the problem instance may contain other symmetries. Also, ordering constraints do not break all symmetries when matrices have both row and column symmetries [3]. Nevertheless, symmetries in matrix models have the advantages of being easily identified and broken at a small cost.

Example 1. Consider a 4x4 Latin square, i.e. a 4x4 matrix to be filled with 4 different symbols in such a way that each symbol occurs exactly once in each row and exactly once in each column. This problem has 576 solutions. Clearly, most symmetries can be easily eliminated by forcing the first row and the first column to be lexicographically ordered. Nonetheless, these constraints do not prevent this problem from having more than one solution: there are still 4 possible solutions. Figure 1 illustrates a 4x4 Latin square after adding the lexicographic constraints and the four possible solutions. `Shatter` is able to identify further symmetries such that only two of these solutions can be found.

We now focus on the SHIPs SAT model [4, 5]. SHIPs is a SAT-based approach for solving the problem of haplotype inference by pure parsimony (HIPP). (A detailed description of SHIPs can be found in [4, 5].) Given a set $\mathcal{G}$ of n genotypes, each of length m, the haplotype inference problem consists in finding a set $\mathcal{H}$ of $2 \cdot n$ haplotypes, not necessarily different, such that for each genotype $g_i \in \mathcal{G}$ there is at least one pair of haplotypes (h_j, h_k), with h_j and $h_k \in \mathcal{H}$ such that the pair (h_j, h_k) explains g_i. The pure parsimony approach finds a solution that minimises the total number of distinct haplotypes used.

The organisation of the SHIPs algorithm considers increasing values r of candidate haplotypes, with $1 \leq r \leq 2 \cdot n$, such that a solution is found when r haplotypes suffice to explain the n genotypes. The SHIPs model [4, 5] can be described by the matrix formulation $G = S^a \cdot H \oplus S^b \cdot H$, where G is a $n \times m$ matrix describing the genotypes, H is a $r \times m$ matrix of haplotype variables, S^a and S^b are $n \times r$ matrices of selector variables, and $\oplus$ is the explanation operation. One of the contributions of the SHIPs model are the techniques for breaking key symmetries in the problem formulation. If matrix H is interpreted as a vector of strings of size m, $H = [h_1 h_2 \dots h_r]^T$, then we can impose the condition $h_1 < h_2 < \dots < h_r$, i.e. the haplotypes are lexicographically sorted. An additional form of symmetry is due to the S variables. If $S^a = [s_1^a \dots s_n^a]^T$ and $S^b = [s_1^b \dots s_n^b]^T$, then we can impose the condition $s_i^a \leq s_i^b$, $1 \leq i \leq n$,

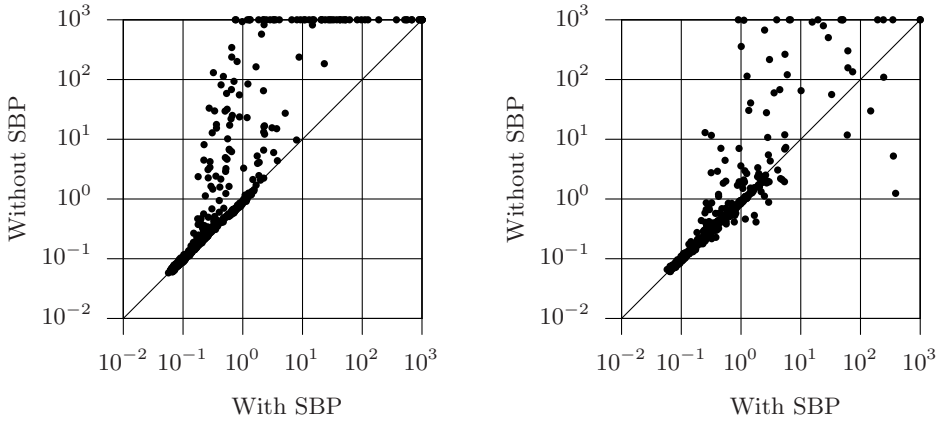

Fig. 2. CPU times with and without instance-independent symmetry breaking predicates (SBP) on unsatisfiable and satisfiable instances.

i.e. for each genotype i, the strings representing the selector variables a and the selector variables b are lexicographically ordered.

4 Experimental Results

This section provides empirical evidence that breaking instance-independent symmetry in SAT matrix models can be more effective than breaking instance-dependent symmetries. Different encodings for the SHIPs matrix model are evaluated. Also, due to the incremental approach implemented in SHIPs, both satisfiable and unsatisfiable problem instances are obtained. Consider a solution with size s: then iterations with $r < s$ represent unsatisfiable instances, and the iteration with $r = s$ represents a satisfiable instance. A set of 1183 problem instances obtained from `http://www.stats.ox.ac.uk/~marchini/phaseoff.html` and from [5] were evaluated. The results were obtained on an Intel Xeon 5160 (3.0GHz with 4GB of RAM) and a timeout of 1000s.

From an initial universe of 1183 instances, we removed 348 instances with equal computed lower and upper bounds [4]. Of the remaining instances, 134 are aborted when symmetry breaking is not used and 74 are aborted when symmetry breaking is used. Moreover, the run times with symmetry breaking are also consistently smaller. Figure 2 provides two plots comparing the effect of breaking instance-independent symmetries in terms of the total CPU time for unsatisfiable and satisfiable instances, respectively. Clearly, for unsatisfiable instances it is *always* useful to break symmetries, whereas for satisfiable instances it is useful in most cases. Next, we compare the use of `shatter` [1] on each set of unsatisfiable and satisfiable instances. `Shatter` may be applied either to the CNF formula resulting from the SHIPs model, for which instance-independent symmetry breaking predicates have been included, or to the plain model, for

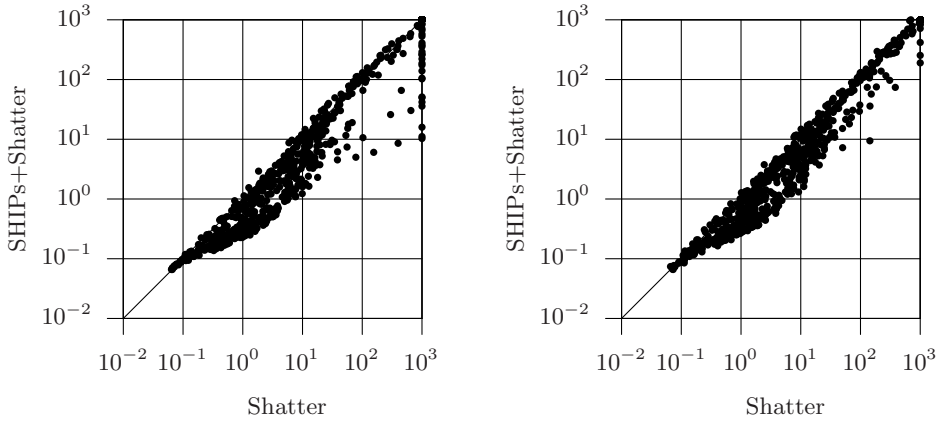

Fig. 3. Shatter vs SHIPS+Shatter on unsatisfiable and satisfiable instances.

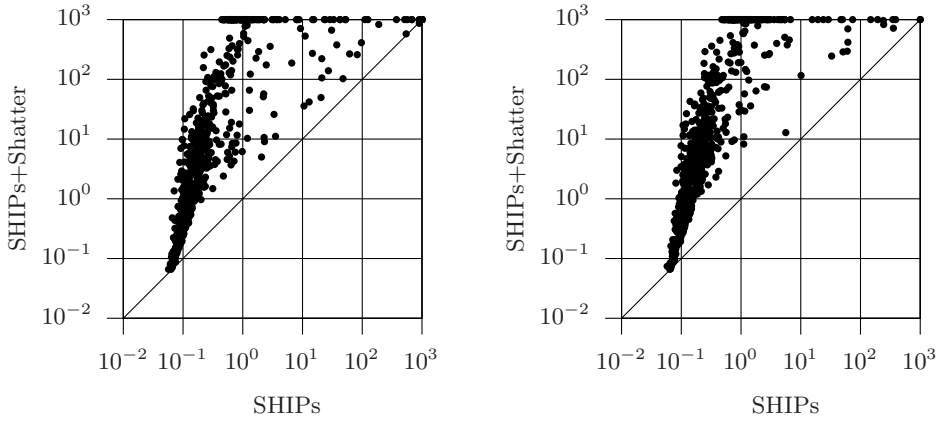

Fig. 4. SHIPs vs SHIPS+Shatter on unsatisfiable and satisfiable instances.

which no symmetries are broken. Figure 3 compares both approaches. Even though `shatter` performs better on the SHIPs model rather than on the plain model, the difference is not significant, in particular for satisfiable instances.

Finally, Figure 4 compares the use of instance-independent symmetry breaking predicates (i.e. SHIPs) with the use of both instance-independent and instance-dependent symmetry breaking predicates (i.e. SHIPs+Shatter) in terms of CPU time. The use of instance-independent symmetry breaking predicates is consistently more efficient than the use of both types of symmetry breaking predicates. Moreover, `Shatter` in unable to break all the symmetries in the allowed CPU time (1000s) for many instances. This is probably due to these instances having many symmetries, which can be easily identified beforehand.

One additional question is: "If there was an oracle giving the CNF formula computed by `shatter` what would be the SAT solver performance?" With this

purpose, the formula computed by `shatter` within 1000s was given to the SAT solver. Then we compared the time required by SHIPs with the time required by the SAT solver on the formula computed by `shatter`. If `shatter` is run on the plain model, i.e. without symmetry breaking predicates, then the SAT solver is able to solve more problem instances than using the plain model, but still less 45 instances than SHIPs. Also, the instances not solved by SHIPs are also not solved after using `shatter`. If `shatter` is run on the SHIPs model, which includes symmetry breaking predicates, then exactly the same instances are not solved. For the instances solved, the use of `shatter` yields a negligible speedup.

5 Conclusions and Future Work

Despite its impact in CP, symmetry breaking is seldom used in SAT. The main reason is that symmetry breaking can be time-consuming and not always effective in modern SAT solvers. This paper explores a different line of research, which has been quite successful in CP: instead of considering instance-dependent symmetry breaking, we propose problem-specific instance-independent symmetry breaking. Clearly, this necessarily depends on the application domain. The paper focus on symmetry breaking techniques for SAT matrix models, and more concretely for the HIPP problem. The experimental results show that more careful modelling of computational problems with SAT techniques, and exploring well-established symmetry breaking techniques, can be a quite effective approach, and can significantly outperform existing instance-dependent symmetry breaking approaches.

References

1. F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. Solving difficult instances in the presence of symmetry. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(9):1117–1137, 2003.
2. J. M. Crawford, M. L. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 1996.
3. P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *International Conference on Principles and Practice of Constraint Programming (CP)*, 2002.
4. I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, 2006.
5. I. Lynce and J. Marques-Silva. SAT in bioinformatics: Making the case with haplotype inference. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 2006.
6. S. Prestwich. First-solution search with symmetry breaking and implied constraints. In *CP Workshop on Modelling and Problem Formulation*, 2001.
7. A. Ramani, I. L. Markov, K. A. Sakallah, and F. A. Aloul. Breaking instance-independent symmetries in exact graph coloring. *Journal of Artificial Intelligence Research*, 26:289–322, 2006.
8. I. Shlyakhter. Generating effective symmetry-breaking predicates for search problems. In *LICS Workshop on Theory and Applications of Satisfiability Testing*, 2001.