Symmetry Breaking in Local Search for Unsatisfiability

Fadi Aloul
Dept of Computer Engineering,
American University of Sharjah,
UAE
faloul@aus.edu

Inês Lynce
IST/INESC-ID,
Technical University
of Lisbon, Portugal
ines@sat.inesc-id.pt

Steven Prestwich
Cork Constraint Computation
Centre, Dept of Computer Science,
University College, Cork, Ireland
s.prestwich@cs.ucc.ie

Abstract

Symmetry breaking can greatly speed up the search for solutions and proofs of unsatisfiability, but has been shown to have a bad effect on local search for solutions. Recently a new form of local search has been used to prove unsatisfiability of SAT problems, based on randomised resolution with greedy heuristics. An interesting question is: does symmetry breaking speed up this form of local search? We present experimental evidence that it does, making randomised refutation a promising new application of symmetry breaking.

1 Introduction

Complete SAT algorithms may be based on resolution or backtracking. Resolution provides a complete proof system by refutation [Robinson, 1965]. The first resolution algorithm was the Davis-Putnam (DP) procedure [Davis and Putnam, 1960] which was then modified to the Davis-Putnam-Logemann-Loveland (DPLL) backtracking algorithm [Davis *et al.*, 1962]. Because of its high space complexity, resolution is often seen as impractical for real-world problems, but there are problems on which general resolution proofs are exponentially smaller than DPLL proofs [Ben-Sasson *et al.*, 2004]. Incomplete SAT algorithms are usually based on local search following early work by [Gu, 1992; Selman *et al.*, 1992]. On some large satisfiable problems, local search finds a solution much more quickly than complete algorithms. Genetic and other evolutionary algorithms have also been applied to SAT but do not yet rival local search. A new form of local search was recently described that is able to prove *unsatisfiability*: the RANGER [Prestwich and Lynce, 2006] and GUNSAT [Audemard and Simon, 2007] algorithms apply resolution to SAT instances in a randomised way, using greedy heuristics and other techniques to speed up the search, in the hope of deriving the empty clause.

Symmetry-breaking has proved to be very effective when combined with complete solvers, by reducing the size of the search space. Probably the simplest and most popular approach is to add constraints to the problem formulation, so that each equivalence class of solutions to the original problem corresponds to a single solution in the new problem. A formal framework for this approach is given in [Puget, 1993]. In constraint programming, symmetry breaking is usually applied manually by the modeller in an instance-independent way, whereas in SAT it is usually applied in an automated, instance-dependent way via generic tools that detect graph automorphisms [Aloul *et al.*, 2003; Crawford *et al.*, 1996]. On some problems the instance-dependent approach is best [Ramani *et al.*, 2006] while on others the opposite holds [Lynce and Marques-Silva, 2007]. Symmetry breaking can also be used to reduce the size of a resolution proof, for example short inductionless proofs of the pigeon-hole principle can be constructed using symmetry [Krishnamurthy, 1985].

Symmetry breaking has also been used in genetic algorithms (though not for SAT to the best of our knowledge). A symmetric optimisation problem has multiple optimum solutions that are symmetrically equivalent, and applying recombination to them may yield offspring with very poor fitness. Symmetry breaking in this context involves designing more complex genetic operators and problem models [Galinier and Hao, 1999], or using clustering techniques [Pelikan and Goldberg, 2000]. However, the use of symmetry-breaking constraints seems to have a bad effect on local search (randomised, non-population-based) algorithms [Prestwich, 2003]. This is true even if we ignore any runtime overheads due to symmetry breaking constraints, and measure only search steps. The reasons for this phenomenon are not completely understood, but detailed experiments show that symmetry breaking constraints transform symmetric solutions into deep local minima, thus decreasing the solution density and increasing the number of local minima, and also reduce the relative sizes of basins of attraction of global minima [Prestwich and Roli, 2005].

Given the above background, what effect should we expect symmetry breaking clauses to have on local search for unsatisfiability as in RANGER and GUNSAT? On one hand, symmetry breaking clauses usually have a bad effect on local search; on the other hand, we might expect them to speed up proof of unsatisfiability even if that proof is generated non-systematically. This paper investigates this question: Section 2 provides background on this new class of local search algorithms, Section 3 reports the results of our experiments, and Section 4 concludes the paper.

2 Local search for unsatisfiability

Local and backtrack search have complementary strengths and weaknesses. Local search has superior scalability on many large problems, but it cannot (in its usual form) prove unsatisfiability. Backtrack search and resolution-based algorithms are (usually) complete, and backtrack search's use of unit propagation, clause learning, dedicated data structures and other methods enables it to outperform local search on some highly-structured problems. This complementarity has inspired research on hybrid approaches such as the use of unit propagation in local search, and more flexible backtracking strategies.

An interesting question is: can local search be applied to *unsatisfiable* problems? Such a method might be able to refute (prove unsatisfiable) SAT problems that defy complete algorithms. The first such algorithm that we know of is RANGER [Prestwich and Lynce, 2006], which explores a space of multisets of resolvents using general resolution, and aims to derive the empty clause non-systematically but greedily. It will eventually refute any unsatisfiable instance while using only bounded memory (by exploiting a recent theoretical result of [Esteban and Torán, 2001]). It can refute some problems more quickly than current DPLL and systematic resolution algorithms, though on most benchmarks it is currently uncompetitive.

The RANGER architecture is shown in Figure 1. It has six parameters: the formula ϕ, three probabilities p_i, p_t, p_g, the width w and the size k of the formula ϕ_i. RANGER begins with any sub-multiset $\phi_1 \subseteq \phi$ (we interpret ϕ, ϕ_i as multisets of clauses). It then performs iterations i, each either replacing a ϕ_i clause by a ϕ clause (with probability p_i) or resolving two ϕ_i clauses and placing the result r into ϕ_i. In the latter case, if r is tautologous or contains more than w literals then it is discarded and $\phi_{i+1} = \phi_i$. Otherwise a ϕ_i clause must be removed to make room for r: either (with probability p_g) the removed clause is the longer of the two parents of r (breaking ties randomly), or it is randomly chosen. In the former case, if r is longer than the parent then r is discarded and $\phi_{i+1} = \phi_i$. At the end of the iteration, any satisfiability-preserving transformation may (with probability p_t) be applied to ϕ, ϕ_{i+1} or both. If the empty clause has been derived then the algorithm terminates with the message "unsatisfiable". Otherwise the algorithm might not terminate, but a time-out condition (omitted here for brevity) may be added.

Local search algorithms usually use *greedy* local moves that reduce the value of an objective function, and *plateau traversal* moves that leave it unchanged. However, they must also allow non-greedy moves in order to escape from local minima. This is often controlled by a parameter known as *noise* (or *temperature* in simulated annealing). RANGER's goal is to derive the empty clause, and a necessary condition for this to occur is that ϕ_i contains at least some small clauses. We call a local move *greedy* if it does not increase the number of literals in ϕ_i. This is guaranteed on line 10, so increasing p_g increases the greediness of the search, reducing the proliferation of large resolvents.

RANGER has a useful convergence property: for any unsatisfiable SAT problem with n variables and m clauses,

```
1  RANGER(φ, p_i, p_t, p_g, w, k):
2    i ← 1 and φ_1 ← {any k clauses from φ}
3    while φ_i does not contain the empty clause
4      with probability p_i
5        replace a random φ_i clause by a
         random φ clause
6      otherwise
7        resolve random φ_i clauses c, c' giving r
8        if r is non-tautologous and |r| ≤ w
9          with probability p_g
10           if |r| ≤ max(|c|, |c'|) replace the
             longer of c, c' by r
11         otherwise
12           replace a random φ_i clause by r
13     with probability p_t
14       apply any satisfiability-preserving
         transformation to φ, φ_i
15     i ← i + 1 and φ_{i+1} ← {the new formula}
16   return UNSATISFIABLE
```

Figure 1: The RANGER architecture

RANGER finds a refutation if $p_i > 0, p_i, p_t, p_g < 1, w = n$ and $k \geq n + 1$ (for a proof see [Prestwich and Lynce, 2006]). The space complexity of RANGER is $O(n + m + kw)$. To guarantee convergence we require $w = n$ and $k \geq n + 1$ so the complexity becomes at least $O(m + n^2)$. In practice we may require k to be several times larger, but a smaller value of w is often sufficient.

Lines 13–14 provide an opportunity to apply helpful satisfiability-preserving transformations to ϕ or ϕ_i or both (if we do not aim for a pure resolution refutation). We apply the subsumption and pure literal rules in several ways. Using ϕ_i clauses to transform ϕ, a feature we shall call *feedback*, preserves useful improvements for the rest of the search. (We believe that for these particular transformations we can set $p_t = 1$ without losing completeness, but we defer the proof until a later paper.) Note that if ϕ is reduced then this will soon be reflected in the ϕ_i via line 5 of the algorithm.

A related algorithm is GUNSAT [Audemard and Simon, 2007] which has a similar architecture but interesting differences. For example, whereas RANGER aims for a high rate of rather unintelligent local moves, GUNSAT takes longer to make more intelligent moves based on a more complex objective function. GUNSAT also uses extended resolution while RANGER uses general resolution. The two algorithms have not yet been compared empirically.

3 Experiments

We now evaluate the effects of symmetry breaking on RANGER. The results are shown in Figure 2. All results are medians over 10 runs with a cutoff time of 1000 seconds. #Steps denotes the number of RANGER iterations, #Time the CPU time taken, #V and #C the number of variables and clauses (respectively) in the SAT instances. Experiments were performed on an Intel Xeon 3 GHz with 4GB RAM running Linux. The RANGER implementation is as described in [Prestwich and Lynce, 2006] with parameter settings k=10V, w=V, p_i=0.1, p_t=0.9 and p_g=0.95. The problem sets are as follows:

- `Chnl` problems represent large unsatisfiable instances that model the routing of X wires in the N channels of field-programmable integrated circuits. Assuming that each channel accepts up to one wire, since $X > N$ the instances are unsatisfiable [Aloul *et al.*, 2003].
- `Hole` represent the famous pigeon hole instances, where the goal is to place X pigeons in N holes.[1] Again each hole can hold up to one pigeon, and since $X > N$ the instances are unsatisfiable.
- `Pipe` represent difficult unsatisfiable instances that model the functional correctness requirements of modern out-of-order microprocessor CPUs. The instances were generated by Miroslav Velev [Velev and Bryant, 2001].
- `X` encodes verification problems of two exclusive-or chains. The instances were generated by Lintao Zhang and Sharad Malik.[2]
- `Urq` are unsatisfiable randomized instances based on expander graphs [Urquhart, 1987].
- The biological instances `b2ar`, `ace`, `non-uniform` and `hapmap` come from the haplotype inference problem. Given a set of genotypes, described using a string alphabet $\{0, 1, 2\}$ the main goal is to identify the minimum number of haplotypes, which are described over with a string over the alphabet $\{0, 1\}$ such that each genotype is explained by a pair of haplotypes. The CNF encoding for this problem is described in [Lynce and Marques-Silva, 2006] as well as the problem instances we have used.

Symmetry was broken by the Shatter system [Aloul *et al.*, 2006; 2003]. Symmetry breaking took only a small fraction of a second, which is not included in our results.

The FPGA and hole instances clearly show a huge improvement due to symmetry breaking. The others (of which five are denoted (1)...(5) for space reasons) were unrefuted within the time limit, with or without symmetry breaking. Thus we have no evidence that symmetry breaking harms RANGER performance, but some evidence that it improves performance. A possible explanation is that the symmetry breaking clauses allow smaller refutations, which are easier to discover than large refutations.

However, it is interesting to note that the instances that RANGER could not refute contain few new variables when adding symmetry breaking (they are not required to model the *phase shift symmetries* of most of these instances), while those it did refute contain many new variables. This might indicate that the improvement was not completely due to symmetry breaking, but also to the use of additional variables, which might have a similar effect to the auxiliary variables introduced in *extended resolution*. (Extended resolution allows the definition of new SAT variables via the *extension rule*

[1] DIMACS Challenge benchmarks, 1996 ftp://Dimacs.rutgers.EDU/pub/challenge/sat/benchmarks/cnf

[2] SAT 2002 Competition http://www.satlive.org/SATCompetition/submittedbenchs.html

without symmetry breaking					
instance	#V	#C	#Lit	#Steps	#Time
chnl10_11	220	1122	2420	n/a	>1000
chnl10_12	240	1344	2880	n/a	>1000
chnl10_13	260	1586	3380	n/a	>1000
chnl11_12	264	1476	3168	n/a	>1000
chnl11_13	286	1742	3718	n/a	>1000
chnl11_20	440	4220	8800	n/a	>1000
hole7	56	204	448	n/a	>1000
hole8	72	297	648	n/a	>1000
hole9	90	415	900	n/a	>1000
hole10	110	561	1210	n/a	>1000
hole11	132	738	1584	n/a	>1000
hole12	156	949	2028	n/a	>1000
Urq3_5	46	470	2912	n/a	>1000
x1.1_16	46	122	364	n/a	>1000
2pipe	892	6695	18637	n/a	>1000
(1)	776	3725	10045	n/a	>1000
(2)	110	428	992	n/a	>1000
(3)	187	643	1584	n/a	>1000
(4)	156	522	1141	n/a	>1000
(5)	223	880	2363	n/a	>1000

with symmetry breaking					
instance	#V	#C	#Lit	#Steps	#Time
chnl10_11	728	3077	9103	2700218	11.605
chnl10_12	796	3487	10213	2999725	18.605
chnl10_13	864	3917	11363	3326896	27.67
chnl11_12	878	3847	11291	4417899	33.85
chnl11_13	953	4321	12561	5155214	50.905
chnl11_20	1478	8255	22683	948902	1.55
hole7	153	567	1663	241347	0.35
hole8	199	776	2261	352256	0.535
hole9	251	1026	2967	626528	1.015
hole10	309	1320	3787	948902	1.55
hole11	373	1661	4727	1153560	2.1
hole12	443	2052	5793	1784522	3.825
Urq3_5	46	500	2941	n/a	>1000
x1.1_16	48	142	385	n/a	>1000
2pipe	1246	8137	23571	n/a	>1000
(1)	780	3746	10073	n/a	>1000
(2)	120	449	1022	n/a	>1000
(3)	205	694	1670	n/a	>1000
(4)	160	531	1153	n/a	>1000
(5)	223	913	2395	n/a	>1000

Key:

(1) 1dlx_c_mc_ex_bp_f

(2) bio-b2ar-simp-b2ar_5.01

(3) bio-ace-simp-ace_5.07

(4) non-uniform-simp-nonunif-10_50.06

(5) hapmap-simp-test_chr21_HCB_30

Figure 2: Experiments on RANGER with and without symmetry breaking

[Tseitin, 1983]. It can lead to exponentially smaller proofs, but is used even less than general resolution because there are no known heuristics for generating the new variables.)

We hope to resolve this issue by further experimentation in future work, either by finding instances without auxiliary variables for which symmetry breaking helps randomised refutation, or by testing the effects of new auxiliary variables on the unrefuted instances. If the explanation turns out to be a form of extended resolution then we may enhance RANGER from general resolution to extended resolution, along the lines of GUNSAT.

4 Conclusion

This work is only at a preliminary stage, but already shows the promise of symmetry breaking in randomised refutation. Local search for unsatisfiability appears to be an exception to the rule that "symmetry breaking is bad for local search".

In retrospect this is perhaps unsurprising: if symmetry breaking allows smaller refutations then these may be easier to find by *any* resolution algorithm, whether systematic or randomised. Moreover, we have not actually applied symmetry breaking to the space explored by the local search algorithm: to do this we would have to restrict the search so that it excludes refutations that are symmetric in some sense to other refutations. This might indeed harm local search performance.

In fact the usual arguments based on solution density and global basins of attraction do not hold when refuting an UNSAT problem. Adding symmetry breaking clauses to a SAT problem increases the number of possible resolution refutations, so there are more search states from which greedily applying resolution leads directly to the empty clause. In other words, in this context symmetry breaking *increases* the size of the basin of attraction of each solution (defined here as a search state containing the empty clause).

Acknowledgements

This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075, and partially supported by Fundação para a Ciência e Tecnologia under research projects POSC/EIA/61852/2004 and POSI/SRI/41926/01.

References

[Aloul *et al.*, 2003] F. Aloul, A. Ramani, I. Markov, and K. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. *IEEE Transactions on Computer Aided Design*, 22(9):1117–1137, 2003.

[Aloul *et al.*, 2006] F. Aloul, K. Sakallah, and I. Markov. Efficient symmetry breaking for boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, 2006.

[Audemard and Simon, 2007] G. Audemard and L. Simon. Gunsat: A greedy local search algorithm for unsatisfiability. In *20th International Joint Conference on Artificial Intelligence*, 2007. Poster.

[Ben-Sasson *et al.*, 2004] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near-optimal separation of treelike and general resolution. *Combinatorica*, 24(4):585–603, 2004.

[Crawford *et al.*, 1996] M. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry breaking predicates for search problems. In *5th International Conference on Principles of Knowledge Representation and Reasoning*, pages 148–159, 1996.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association of Computing Machinery*, 7(3), 1960.

[Davis *et al.*, 1962] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[Esteban and Torán, 2001] J. L. Esteban and J. Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001.

[Galinier and Hao, 1999] P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[Gu, 1992] Jun Gu. Efficient local search for very large-scale satisfiability problems. *Sigart Bulletin*, 3(1):8–12, 1992.

[Krishnamurthy, 1985] B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:327–337, 1985.

[Lynce and Marques-Silva, 2006] I. Lynce and J. P. Marques-Silva. Sat in bioinformatics: Making the case with haplotype inference. In *9th International Conference on Theory and Applications of Satisfiability Testing*, volume 4121 of *Lecture Notes in Computer Science*, pages 136–141. Springer, 2006.

[Lynce and Marques-Silva, 2007] I. Lynce and J. P. Marques-Silva. Breaking symmetries in sat matrix models. In *10th International Conference on Theory and Applications of Satisfiability Testing*, volume 4501 of *Lecture Notes in Computer Science*, pages 22–27. Springer, 2007.

[Pelikan and Goldberg, 2000] M. Pelikan and D. E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In *6th International Conference on Parallel Problem Solving from Nature*, 2000.

[Prestwich and Lynce, 2006] S. D. Prestwich and I. Lynce. Local search for unsatisfiability. In *9th International Conference on Theory and Applications of Satisfiability Testing*, volume 4121 of *Lecture Notes in Computer Science*, pages 283–296. Springer, 2006.

[Prestwich and Roli, 2005] S. D. Prestwich and A. Roli. Symmetry breaking and local search spaces. In *2nd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3524 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2005.

[Prestwich, 2003] S. D. Prestwich. Negative effects of modeling techniques on search performance. *Annals of Operations Research*, 118:137–150, 2003.

[Puget, 1993] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In *International Symposium on Methodologies for Intelligent Systems*, volume 689 of *Lecture Notes in Computer Science*, pages

350–361. Springer-Verlag, 1993. J. Komorowski, Z. W. Ras (eds.), Methodologies for Intelligent Systems.

[Ramani *et al.*, 2006] 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.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *10th National Conference on Artificial Intelligence*, pages 440–446. MIT Press, 1992.

[Tseitin, 1983] G. Tseitin. On the complexity of derivation in propositional calculus. *Automation of Reasoning: Classical Papers in Computational Logic*, 2:466–483, 1983.

[Urquhart, 1987] A. Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, 1987.

[Velev and Bryant, 2001] M. N. Velev and R. E. Bryant. Effective use of boolean satisfiability procedures in the formal verification of superscalar and vliw microprocessors. In *Design Automation Conference*, pages 226–231, 2001.