Local Search for Very Large SAT Problems

Steven Prestwich[1] and Colin Quirke[2]

[1] Cork Constraint Computation Centre
Department of Computer Science, University College, Cork, Ireland
`s.prestwich@cs.ucc.ie`
[2] Boole Centre for Research in Informatics,
University College, Cork, Ireland
`c.quirke@4c.ucc.ie`

Abstract. The Walksat local search algorithm has previously been extended to handle quantification over variables. This greatly reduces model sizes, but in order to guide greedy moves the algorithm still maintains a set of violated clauses. For very large problems, or at the start of a search, this can cause memory problems. We design a new local search algorithm that does not maintain this set and is therefore applicable to larger SAT problems. We show that this algorithm is nevertheless greedy in a probabilistic sense, and that it has good performance on some SAT problems. We also describe a prototype lifted version of the algorithm, and show that advanced constraint programming techniques pay off when searching for violated clauses.

Introduction

To handle SAT problems with many clauses, an interesting and powerful technique is *lifting* [4, 12]: compressing a large set of clauses into a single formula, then redesigning a search algorithm (backtracking or local search) to operate on these formulae. This is impractical for random problems which (almost by definition) cannot be compressed, but for real problems very significant compression may be achieved. The problem of locating clauses with certain properties (for example the property of violation), which must be solved by any search algorithm, is an NP-complete problem [4]. This *subsearch* problem is therefore an interesting candidate for solution by constraint-based methods, and we aim to apply more powerful methods than previously used.

Previous lifted SAT algorithms [4, 12] have been direct adaptations of existing search algorithms, which is a natural approach. However, it can cause certain implementation difficulties. In particular, local search algorithms for SAT typically maintain a set of currently violated clauses. This is used to guide *greedy* local moves, which maximally reduce the number of violations. But for very large problems the set of violated clauses may also be large. Though it often remains manageably small it is not guaranteed to do so for all problems, especially in early phases of the search when it is far from any solution. In fact it is reported in [4] that a lifted solver ran out of memory on a lifted model corresponding to billions of clauses.

To avoid this problem we design a new local search algorithm specifically for lifting. We would ideally like a local search algorithm for SAT that is reasonably efficient on real problems, yet does not count the number of violated clauses. This would allow us to implement a lifted local search algorithm without maintaining an actual set of violated clauses. However, it is not obvious how to guide local search without this information. Even early local search algorithms such as GSAT [18] were greedy. The more modern Walksat [17] and its variants all count violated clauses in some way. SDF [16] and DLM [19] adjust clause weights and do not simply count violated clauses, but still maintain the set of violations. Analogous techniques are used in combinatorial optimization: simulated annealing [8] uses the change in solution quality to decide whether to accept a random move or not, and genetic algorithms [5] are biased towards fitter organisms. To perform local search without measuring the quality of the current or neighbouring states seems almost perverse, but would have a practical advantage for lifting.

A new local algorithm

Some non-greedy local search algorithms have already been explored. The importance of greediness was been shown to be debatable in [3]. In their experiments a "timid" (not very greedy) hill-climbing algorithm gave similar results to a more greedy algorithm. Schöning's algorithm [15] randomly selects a violated clause, then flips the value of a randomly chosen variable in that clause. This is similar to an algorithm of Papadimitriou [11] for 2-SAT, except that after $3n$ flips it restarts from a random total assignment, where n is the number of SAT variables. No attempt is made to count violated clauses and the algorithm is shown to have an expected runtime within a polynomial factor of $O(2(1-k)^n)$ for a k-SAT problem. This algorithm was designed with theoretical analysis in mind and no empirical results seem to have been reported. However, it is equivalent (modulo restarts) to Walksat with the noise parameter set to 1 and freebie moves suppressed (*freebie moves* create no new violations). For many problems choosing an appropriate noise level is critically important for finding a solution in a reasonable time, and freebie moves are known to improve performance, so Schöning's algorithm is unlikely to be a good general-purpose local search algorithm.

We decided to test the algorithm empirically on random 3-SAT problems, in which each clause contains three literals, and each literal is negated with probability 0.5. The hard problems are known to be in the *crossover* region where 50% of problems are satisfiable, which occurs when the ratio of clauses c to variables v is approximately $c/v = 4.258 + 58.26v^{-5/3}$. This formula was carefully fitted to data up to 400 variables in [1]. We tested Schöning's algorithm on these problems using various restart thresholds, including ∞ and its recommended value of $3n$. We applied it to 1000 problems of various sizes and took the 25th percentile of the execution time to find a solution. Because exactly 50% of problems from the crossover region are satisfiable, the 25th percentile result for all problems is the median result for the solvable problems. (This method is faster than filtering out unsolvable problems via a SAT backtracker.) The results are shown in Figure 1, along with results for Walksat using noise 0.5 (which is roughly optimal for these problems) and optimal restart intervals (results taken from [13], no result for $n = 75$ available). Entries marked "—" denote more than 3,000,000 flips. Schöning's algorithm clearly scales poorly compared to Walksat, despite its good theoretical properties. Surprisingly, its best results were obtained with infinite restart intervals.

		Schöning's algorithm + various intervals				
n	WSAT	$3n$	$10n$	$30n$	$100n$	∞
25	94	656	471	418	422	432
50	414	26672	9866	11200	7688	8142
75	?	847788	229154	268674	242604	145927
100	2123	—	—	—	—	—

Fig. 1. Flip results for Schöning's algorithm with various restart intervals

Despite this unpromising start we tried various modifications in the hope of improving performance. We restricted ourselves to heuristics based purely on locating a single violated clause. In other words, we do not try to measure the quality of the search state, nor the relative quality of the current state and the state after a candidate flip. Instead we look for new heuristics. A clue lies in the fact that Schöning's algorithm is similar to Walksat with maximum noise, thus making no greedy moves. If we can find a way of making the search more greedy then this might move its performance closer to that of Walksat. But how can we make it greedy if we do not count violated clauses? First we define a weak form of greediness:

Definition. *A variable selection heuristic is probabilistically greedy (PG) if non-greedy local moves are more likely to be immediately reversed than greedy local moves.*

A *greedy move* is defined here as one that decreases the number of violated clauses, while *immediately reversed* means that the next local move flips the same variable back to its previous truth assignment. Our aim is to find PG heuristics. Some local search algorithms, such as the Novelty variant of Walksat [10], have a bias towards flipping the least-recently flipped variable. The TABU variant also prevents the reversal of recent flips. We decided to try flipping the *most* recently flipped variable, a heuristic we shall call MRF (most recently flipped). Note that the choice of violated clause is random, so the most recently flipped variable in the selected clause is not necessarily the most recently flipped variable of all. It turns out that a large class of heuristics including MRF is PG:

Theorem 1. *Any heuristic H that flips a variable in a randomly chosen violated clause is PG.*

Proof. Suppose that at flip i H chooses variable v. Firstly, if flipping v increases the number of violated clauses then all the newly-violated clauses must contain v, so the proportion of violated clauses containing v increases. Therefore the probability of H selecting v at flip $i+1$ is higher than it was at flip i. Secondly, if flipping v decreases the number of violated clauses then, by the same argument, the probability of H selecting v at flip $i+1$ is lower than it was at flip i. Thus a greedy move is less likely to be reversed than a non-greedy one. □

The advantage of MRF is that it maximizes the probability that a non-greedy move will be reversed. Suppose that such a move flips variable v, creating new violated clauses S_v. The next move randomly selects a violated clause C. MRF reverses the move if and only if $C \in S_v$, whereas any other variable selection heuristic (for example random selection) only reverses it some of the time. In particular, flipping the least recently flipped variable will never immediately reverse a non-greedy flip. MRF is likely to pay off most when the search is close to a solution, because typically there will be few violated clauses, so after any non-greedy move most of the violated clauses are in S_v and the move is very likely to be reversed.

Another property of some local search algorithms is *probabilistic asymptotic completeness* (PAC) [6]. If a local search algorithm is PAC then its probability of solving a (satisfiable) problem tends to 1 as its runtime tends to ∞. This property is not guaranteed to yield a good local search algorithm, but there is empirical evidence that it is an important ingredient [6]. Moreover, a PAC algorithm never needs (in principle) to randomly restart, because it will eventually escape from any local minimum. Avoiding the need for restarts reduces the number of runtime parameters that require tuning. If an algorithm is not PAC then it is *essentially incomplete* and requires restarts to make it PAC. It is still an open question whether Walksat is PAC though it is suspected to be, and has been shown to be for 2-SAT [2]. Schöning's algorithm uses restarts and is therefore PAC but our algorithm is not. We therefore modify it by adding a noise parameter, giving the algorithm shown in Figure 2, which we call PG-SAT.

```
assign all variables to random truth values
let $S$ be the set of violated clauses
while $S \neq \{\}$
    randomly choose a clause $C \in S$
    with probability $p$
        flip a randomly chosen variable in $C$
    else with probability $1 - p$
        flip the most recently flipped variable in $C$
    update $S$
```

Fig. 2. The PG-SAT local search algorithm

Theorem 2. *PG-SAT with $p > 0$ is PAC.*

Proof. Consider a search state A and a solution S. If $A = S$ then we have found a solution. Otherwise choose any violated clause C. At least one of the variables in C currently has the opposite truth value to its value under S (if not then C is not violated because S is a solution). Let v be one of these variables. If $p > 0$ then PG-SAT has a chance (a non-zero probability) of selecting

v and flipping it, thus reducing the Hamming distance between A and S. Therefore PG-SAT has a chance of moving directly to any solution from any search state. □

(Note that this proof does not assume any clause selection strategy, so PG-SAT is PAC even under non-random clause selection; however, our PG proof assumes random clause selection.) We tested PG-SAT with noise 0.2 on random 3-SAT problems, with results shown in Figure 3. Note that random problems cannot benefit from lifting because lifting requires some form of regularity in the clauses, but random 3-SAT is a standard benchmark for search algorithms. The results show that PG-SAT does not use many more flips than Walksat (with roughly optimal runtime parameters) for these problems. This supports the thesis of [3] that extreme greediness is not a critical factor in local search, and that weaker forms are sufficient.

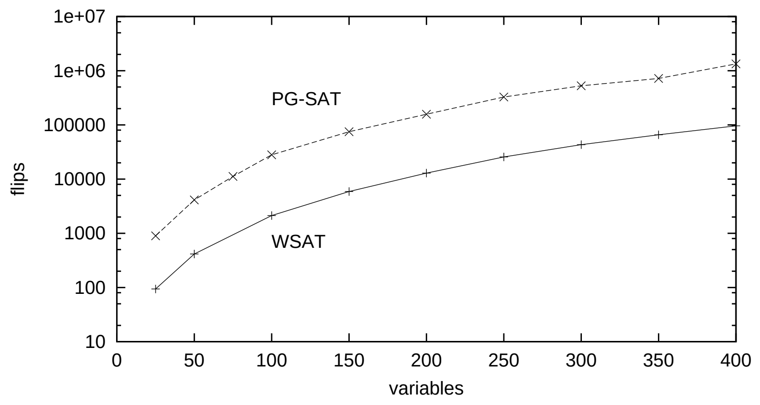

Fig. 3. Results on random 3-SAT

Next we compared PG-SAT with Walksat on some other SAT benchmarks (results omitted for space reasons). PG-SAT seems to be relatively worse on some problems than on random 3-SAT, and on some problems such as planning it was much worse. This may be because greedy moves are more important for handling variable dependencies arising in structured problems, and that PG-SAT's weaker form of greed is less effective than truly greedy search. But this is still a prototype algorithm and we hope to find improved heuristics for structured problems in future work. We have experimented with several variants, including an adaptive noise strategy that boosted noise if too many flips are reversed immediately, but this did not noticeably improve the results, nor did random restarts.

Lifting the algorithm

An advantage of PG-SAT over Walksat (and most other local search algorithms) is that it requires fewer subsearch problems. In fact it has just one: to randomly select a violated clause, and if there are none then report this fact. This simplicity makes a lifted version easier to implement. We implemented a prototype lifted PG-SAT (LPG-SAT) in C++ using the EFC constraint library [7], which provides powerful constraint programming techniques such as forward checking (FC), generalized arc-consistency (GAC) [9], conflict-directed backjumping (CBJ) [14] and dynamic variable ordering. The lifted algorithms in [12] did not use constraint propagation; one in [4] did but used static variable ordering. Our algorithm therefore uses more constraint-based techniques than previous lifted algorithms.

We represent SAT models using formulae very similar to the *extended axioms* of [4]. We can define variables with finite integer domains and specify (binary or non-binary) constraints on them. Each variable may have any number of integer arguments so that families of variables can be defined, and constraints can be expressed on the arguments. As an example consider the problem of finding Ramsey numbers. The Ramsey number $R_{k,l}$ is the smallest number such that every graph with at least that number of vertices either contains a clique of size k or an independent set of size l. Ramsey proved that such such a number exists for every (k, l) pair, but computing them is very difficult. If a graph of n vertices can be constructed, by whatever means, with no k-cliques or l-independent sets, then this proves that $R_{k,l} > n$. The problem has a simple SAT encoding. Consider $R_{4,4}$ which is known to be 18. The problem of showing that $R_{4,4} > 17$ can be SAT-encoded as follows. Define Boolean variables $v_{i,j}$ where $1 \leq i < j \leq 17$ and $v_{i,j}{=}T$ if and only if vertices i and j are adjacent. There are two clause schemata:

$$1 \leq a < b < c < d \leq 17 \rightarrow v_{a,b} \vee v_{a,c} \vee v_{a,d} \vee v_{b,c} \vee v_{b,d} \vee v_{c,d}$$
$$1 \leq a < b < c < d \leq 17 \rightarrow \overline{v}_{a,b} \vee \overline{v}_{a,c} \vee \overline{v}_{a,d} \vee \overline{v}_{b,c} \vee \overline{v}_{b,d} \vee \overline{v}_{c,d}$$

where a, b, c, d are universally quantified. This model requires 136 variables and 4760 clauses, a small problem that can be solved in seconds by Walksat. However, the problems rapidly become far larger. The exact value of $R_{5,5}$ is unknown but lies between 43 and 49; the $R_{5,5} > 43$ problem requires about two million clauses, while others have much larger models.

On SAT problems other than Ramsey numbers we may need more than two schemata. If subsearch finds no solution (a violated clause) using a given clause schema then it does not contain a violated clause, so it tries the next schema, until either a violated clause is found or no schemata remain (in the latter case the SAT problem is solved). A complication is that clause schemata may encode different numbers of clauses. We therefore weight schemata according to the number of clauses that they encode, and when searching for a violated clause the schemata are selected in an order that is random but biased to select large schemata first. Without this bias, clauses occurring in smaller schemata are more likely to be selected. The bias helps to correct this, but because the clause selection is still basically random the algorithm is still PG. In future work we will investigate other biases, for example it might make sense to search for low-arity violated clauses first. To randomly select a violated clause we use random value orderings, and smallest-domain variable ordering with random tie-breaking. Both techniques are supported by EFC, as are constraints such as $<$ which can be used during subsearch. Note that we do not guarantee a uniform distribution of violated clauses, but that any violated clause may in principle be selected, which is sufficient to guarantee PAC.

One motivation for lifting is that there may exist models that would give better results with some search algorithm, but are never used because they have large space complexities. If space is not an issue then we are free to use such models. An example is the Golomb ruler problem: find an ordered sequence of integers (*marks*) $0 = x_1 < x_2 < \ldots < x_m = \ell$ such that the $m(m-1)/2$ differences $x_j - x_i$ $(j > i)$ are distinct, where ℓ is the permitted ruler length. Several models are given in [20], including a *binary/ternary* model with space complexity $O(m^3)$ and a *quaternary* model with $O(m^4)$. One would normally choose the former because of its smaller models, but the latter is simpler and may give better results with some search algorithms. In fact it turns out that with a constraint solver (Ilog Solver) the quaternary model gives inferior results, but in future work we will test it against the binary/ternary model with local search.

To evaluate the benefits of constraint programming techniques in lifted local search we compare the flip rates of Walksat against LPG-SAT with FC, GAC, FC with CBJ, and GAC with CBJ, on the quaternary Golomb ruler model. The results in Figure 4 show that all versions of LPG-SAT perform similarly for small problems, but that GAC has superior scaling as the problem size grows. More powerful constraint propagation leads to faster subsearch and therefore faster location of violated clauses. CBJ improved FC but not GAC. Walksat has a higher flip rate than both on smaller problems but does not scale as well as LPG-SAT[GAC]. We conclude that lifting with an appropriate constraint solver pays off on sufficiently large problems (crossover between Walksat and LPG-SAT[GAC] occurs at approximately 600,000 clauses). In future work we aim to improve both the implementation and the search heuristics.

Acknowledgement

This work was supported by the Boole Centre for Research in Informatics, University College, Cork, Ireland. Many thanks to George Katsirelos, Department of Computer Science, University of Toronto, for implementing new EFC features at our request.

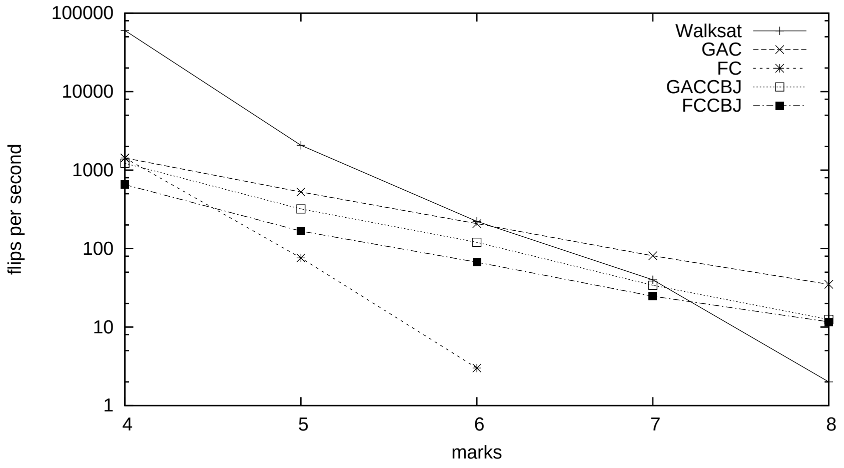

Fig. 4. Flip rate scaling on Golomb rulers

References

1. J. M. Crawford, L. D. Auton. Experimental Results on the Crossover Point in Random 3SAT. *Artificial Intelligence* vol. 81, nos. 1–2, 1996, pp. 31–57.
2. J. Culberson, I. P. Gent, H. H. Hoos. On the Probabilistic Approximate Completeness of WalkSAT for 2-SAT. Technical Report APES-15A-2000, Department of Computer Science, University of Strathclyde, 2000.
3. I. P. Gent, T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. *Eleventh National Conference on Artificial Intelligence*, AAAI Press/MIT Press, 1993, pp. 28–33.
4. Satisfiability Algorithms and Finite Quantification. M. L. Ginsberg, A. J. Parkes. *Seventh International Conference Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 2000, pp. 690–701.
5. J. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.
6. H. H. Hoos. On the Run-time Behaviour of Stochastic Local Search Algorithms for SAT. *Sixteenth National Conference on Artificial Intelligence*, AAAI Press, 1999, pp. 661–666.
7. G. Katsirelos. EFC, available at http://www.cs.toronto.edu/˜gkatsi/efc/efc.html.
8. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimization by Simulated Annealing. *Science* vol. 220, 1983, pp. 671–680.
9. A. K. Mackworth. On Reading Sketch Maps. *Fifth International Joint Conference on Artificial Intelligence*, Kaufmann, 1977, pp. 598–606.
10. D. McAllester, B. Selman, H. Kautz. Evidence for Invariants in Local Search. *Fourteenth National Conference on Artificial Intelligence*, AAAI Press, 1997, pp.
11. C. H. Papadimitriou. On Selecting a Satisfying Truth Assignment. *Thirty-Second Annual IEEE Symposium on Foundations of Computing*, 1991, pp. 163–169.
12. A. J. Parkes. Lifted Search Engines for Satisfiability. PhD dissertation, University of Oregon, 1999.
13. A. J. Parkes, J. P. Walser. Tuning Local Search for Satisfiability Testing. *Thirteenth National Conference on Artificial Intelligence*, AAAI Press, 1996, pp. 356–362.
14. P. Prosser. Hybrid Algorithms for the Constraint Satisfaction Problem. *Computational Intelligence* vol. 9 no. 3, 1993, pp. 268–299.
15. U. Schöning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. *Fortieth Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1999, pp. 410–414.
16. D. Schuurmans, F. Southey. Local Search Characteristics of Incomplete SAT Procedures. *Seventeenth National Conference on Artificial Intelligence*, AAAI Press, 2000, pp. 297–302.
17. B. Selman, H. Kautz, B. Cohen. Noise Strategies for Improving Local Search. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press, 1994, pp. 337–343.
18. B. Selman, H. Levesque, D. Mitchell. A New Method for Solving Hard Satisfiability Problems. *Proceedings of the Tenth National Conference on Artificial Intelligence*, MIT Press 1992, pp. 440–446.
19. Y. Shang, B. W. Wah. A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems. *Journal of Global Optimization* vol. 10, Kluwer 1997, pp. 1–40.
20. B. Smith, K. Stergiou, T. Walsh. Modeling the Golomb Ruler Problem. Research Report 1999.12, University of Leeds, England, June 1999 (presented at the IJCAI'99 Workshop on Non-binary Constraints).