GUIDED LOCAL SEARCH APPLIED TO THE SAT PROBLEM

Patrick Mills and Edward Tsang
{millph,edward}@essex.ac.uk
Department of Computer Science,
University of Essex,
Wivenhoe Park,
Colchester,
Essex CO4 3SQ,
United Kingdom.

Guided Local Search (GLS) has been shown to be successful in solving a number of practical real life problems, such as the travelling salesman problem, radio link frequency assignment problem and the vehicle routing problem. GLS is a penalty-based meta-heuristic, which works by augmenting the objective function of a local search algorithm with penalties, to help guide them out of local minima. Our aim is to show that adding GLS to Local Search algorithms generally enhances the performance of such algorithms, which do not include some similar meta-heuristic already. The SAT problem is a class of NP-complete problems. It is known to be important in mathematical logic, constraint satisfaction, VLSI engineering and computing theory. It has recently been the focus of much research on local search algorithms, for example, GSAT and WalkSAT. In this paper, we show progress in applying GLS to local search algorithms along similar lines to GSAT. Results so far show that GLS can reduce the amount of computational effort required to find a solution, when added to such local search algorithms and can also improve the success rate in finding solutions for the local search algorithms.

Keywords: Combinatorial Optimisation, Search Algorithms

1. INTRODUCTION

Guided Local Search (Voudouris 1997) has been applied to a number of real life problems, including BT's workforce scheduling problem (Tsang & Voudouris 1997), the travelling salesman problem (Voudouris & Tsang 1998), the vehicle routing problem (Kilby *et al* 1997), function optimisation (Voudouris 1998) and the radio link frequency assignment problem (Voudouris & Tsang 1998b).

GLS is a meta-heuristic, which sits on top of local search procedures for helping them escape from local minima. GLS is a generalisation of the GENET neural network (Devenport *et al* 1994, Davenport 1997), for solving constraint satisfaction and optimisation problems. Recently, Lau and Tsang (Lau & Tsang 1998a, Lau & Tsang 1998b) have shown how GLS can be sat on top of a specialised Genetic algorithm, which they call the Guided Genetic algorithm (GGA) for solving constraint satisfaction and optimisation problems, which they apply to the Processor Configuration Problem (Lau & Tsang 1998b) and the Generalised Assignment problem (Lau & Tsang 1998a).

The SAT problem is an important problem in mathematical logic, inference, machine learning, VLSI engineering, and computing theory (Gu 1993), and has been the focus of a lot of successful research into local search algorithms (see for example Gent & Walsh 1993, Selman & Kautz 1993a, Selman *et al* 1994, Shang & Wah 1998a). The SAT problem is a special case of a constraint satisfaction problem (CSP) where the variables take boolean values and the constraints are disjunctions (logical Or) of literals (variables or their negations).

In this paper, we show how GLS can be successfully added to local search algorithms, along the same lines to GSAT (Selman *et al* 1992) without restarts to successfully solve hard SAT problems. We give an evaluation of GLS's performance on hard benchmark problems from the DIMACS archive, comparing the results against those of other local search algorithms reported on these problems and the local search algorithms without GLS.

1.1 The SAT Problem

The SAT problem is an important class of constraint satisfaction problem (Tsang 1993), where the domain of a variable is always the set { false, true }, and each constraint is a logical disjunction. The SAT problem is important in solving many practical problems in mathematical logic, constraint satisfaction, VLSI engineering and computing theory (Gu 1993).

The SAT problem is defined below:

- A set of m (boolean) variables, $Z = \{x_1,x_2,..,x_n\}$, each of which may take the values true or false.
- A set of n clauses, $C = \{ C_1, C_2,.., C_n \}$, each of which is a disjunction of a set of literals (a variable or it's negation), e.g. $x_1 \vee \neg x_2 \vee x_3$.
- The goal of the SAT problem is to find an assignment of values to variables, if one exists, where all the clauses are satisfied (evaluate to true) or prove the problem is unsatisfiable, if no valid assignment exists (currently only complete algorithms can prove unsatisfiability).

Both complete and incomplete algorithms have been used to solve the SAT problem. Of the complete algorithms, one of the best known is the Davis-Putnam procedure (Davis & Putnam 1960), which is based on resolution. Of the incomplete local search algorithms, the best known is probably GSAT (first reported in (Selman *et al* 1992)), based on random restarts and steepest gradient descent and the related WalkSAT (Selman *et al* 1994) based on random walk with greedy variable selection heuristics.

Recently, (Shang & Wah 1998b, Wu & Wah, 1999) have applied DLM to the SAT problem with excellent results. DLM could be seen to be quite similar to GLS, although it does not use selective penalisation (see Section 1.3), and is based on theoretical mathematical foundations, whereas GLS is the result of many years of research based on empirical results. The two methods are related by the use of some kind of augmented objective function, with extra terms to guide the search out of local minima (Lagrangian multipliers in DLM, and penalties in GLS). So far both DLM (see Shang 1997 and Wu 1998 for a full list) and GLS (see Introduction) have been applied to a number of real-life problems.

1.2 Local Search for the SAT Problem

To cast the SAT problem as a local search problem, we simply want to minimise the number of unsatisfied clauses, which we will give as our basic objective function, *g*, given below.

$$g(x) = \#\{C_i \bullet C_i \text{ is unsatisfied}\} \tag{1}$$

The local search algorithm we use is based on ideas reported by Gent and Walsh in (Gent & Walsh 1993), who conduct extensive experiments on lots of different variations of the basic GSAT algorithm (Selman *et al* 1992). Gent and Walsh (1993) show how HSAT, which uses a history mechanism to choose different possible moves, outperforms other procedures on a random SAT problems, and is hence a good starting point for building our local search algorithm underneath GLS. We sort the variables into buckets of downward (if the variable is flipped the objective function will decrease) and sideways moves (if the variable is flipped the objective function remains the same). We then select the least recently flipped variable (this could be seen as form of tabu search, Glover 1989, 1990), which will result in a decrease in the objective function if one is available and flip it. Otherwise we select a so-called sideways move and flip that variable. We allow the maximum number of consecutive sideways moves to be a parameter, *smax* to the local search algorithm.

Pseudo code for our local search algorithm is given in Figure 1. In this paper *smax* is always set to 10, unless otherwise stated.

In our implementation, we incrementally update the changes in the objective function, which would result when a variable is flipped, and also the buckets storing the variables which if flipped result in downwards and sideways moves for the current search state, as this is more efficient than re-calculating every time.

Function LocalSATSearch(**s**, *g*, *smax*)
{
    While ($g(\mathbf{s}) > 0$)
    {
        Modify **s**, by flipping the least recently flipped variable, which will decrease the objective function $g(\mathbf{s})$, if one exists.

        Otherwise, if no variable exists, which will decrease the objective function $g(\mathbf{s})$, modify **s** by, flipping the least recently flipped variable, which does not increase or decrease the objective function.

        If the last *smax* moves were sideways and no downward move is available, return **s**.
    }

    Return **s**
}

Figure 1: Pseudo code for local search for the SAT problem

1.3 Problem Reduction

Some SAT problems have been very difficult for GLS to solve. We have found that using a standard technique, often used in complete algorithms for the SAT problem called Unit Propagation (Wu & Wah, 1999 used such a method for reducing SAT problems for their local search algorithm), we can reduce the number of variables in the SAT problem. For some hard SAT problems, the effect can be quite dramatic, whilst not taking much time to perform this reduction. The reduction is performed by taking advantage of the fact that some problems contain, so-called *Unit clauses* (clauses that contain only one variable). If a clause contains only one variable, obviously that variable must be set to satisfy that clause, if a feasible assignment to the problem is to be found. Once, this is done, then any clauses on the variable which was just fixed which are unsatisfied with respect to variable whose value has already been fixed and have only one unfixed variable left, can also be propagated in a similar way. In addition to this, if a clause contains a fixed variable, which satisfies that particular clause, then that clause will always be satisfied, and can thus be removed from the problem. In this way we reduce the some SAT problems to an equivalent problem, by removing variables and thus reducing the size of search space our local search algorithm will have to traverse.

1.4 Guided Local Search

Guided local search, (See Voudouris 1997 for a more detailed description) sits on top of a local search algorithm. Given a candidate solution **s** and a cost function *g*, a local search algorithm is expected to return a candidate solution **s'** according to its neighbourhood function. **s'** is hopefully better than **s** (i.e. hopefully $g(\mathbf{s}') < g(\mathbf{s})$ in a minimisation problem).

Solution features are used to distinguish between solutions with different characteristics, so those undesirable characteristics can be penalised by GLS. The choice of solution features therefore depends on the type of problem, and also to a certain extent on the local search algorithm. Each feature, f_i defined must have the following components:

- An Indicator function, indicating whether the feature is present in the current solution **s** or not:

$$I_{fi}(\mathbf{s}) = \begin{cases} 1, & \text{if feature present in solution } \mathbf{s} \\ 0, & \text{otherwise} \end{cases}$$

- A cost function $c_{fi}(\mathbf{s})$, which gives the cost of having the feature present in the solution **s**.
- A penalty p_{fi}, initially set to 0, used to penalise occurrences of the feature, in local minima.

In the SAT problem, we define features as violated clauses (to be elaborated below).

When the Local Search algorithm returns a local minimum, **s**, which does not satisfy the termination criteria (e.g. all clauses satisfied), GLS penalises all the features present in that solution which have maximum utility, $Util(\mathbf{s},f_i)$ (as defined in below), by incrementing it's penalty.

$$Util(\mathbf{s}, f_i) = \frac{c_{fi}(\mathbf{s})}{1 + p_{fi}} \tag{2}$$

The idea is to penalise features with higher costs first, with the utility of doing so decreasing, the more times they are penalised.

GLS uses an augmented cost function (see below), to allow it to guide the Local Search algorithm out of the local minimum, by penalising features present in that local minimum. The idea is to make the local minimum more costly than the surrounding search space, where these features are not present, thus guiding the local search algorithm out of the local minimum.

$$h(\mathbf{s}) = g(\mathbf{s}) + \lambda \cdot \sum p_{fi} \cdot I_{fi}(\mathbf{s}) \tag{3}$$

The parameter λ may be used to alter the intensification of the search for solutions. A higher value for λ will result in a more diverse search, where plateaus and basins in the search are searched more coarsely; a low value will result in a more intensive search for the solution, where the plateaus and basins in the search landscape are searched with more care. In this paper λ is always set to 1, unless otherwise stated.

1.5 GLSSAT: Guided Local Search for the SAT Problem

To use Guided Local Search for a particular application, a set of features must be defined, and the local search algorithm must be called within the GLS procedure. In the SAT problem unsatisfied clauses seem to make a good feature to penalise, as they are solution features, which we wish to eliminate from the final solution, and moves made by the local search algorithm (flipping a variable), will usually result in a different set of violated clauses.

$$I_{ci}(\mathbf{s}) = \begin{cases} 1, & \text{if clause is unsatisfied by solution } \mathbf{s} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$C_{ci}(\mathbf{s}) = 1 \tag{5}$$

Using clauses as features is similar to using constraints as features as in the RLFAP problem, (Voudouris & Tsang, 1998b). We define the indicator and cost functions associated with clause features in equations (4) & (5). The feature is present if the clause is unsatisfied, and not present if it is satisfied. The cost of any unsatisfied clause is 1 for the standard SAT problem (since we have no preference for any particular unsatisfied clauses to be eliminated from the solutions, as we want to eliminate all of them). A related problem, the weighted MAX-SAT problem has weights associated with each clause, and the objective is to maximise the sum of the weights of satisfied clauses. A set of weighted MAX-SAT benchmark problems have been successfully solved using GLS, with the clause features cost equal to the weight associated with each clause, see (Mills & Tsang, 1999), with better results than any other algorithm reported in the literature on these benchmark problems.

As GLS usually uses a simple augmented evaluation function, as defined in equation (3), we need to augment the objective function for the local search algorithm in Figure 1, to take into account the penalty terms. This means passing the local search algorithm the augmented objective function h, instead of original objective function g, so that it searches for flips, which minimise h instead of minimising g.

1.6 GLS Extensions: Bounded penalties

Whilst just adding GLS alone to the local search algorithm described earlier is usually enough to solve most problems in the DIMACS benchmark archive, typically in seconds, a few problems, remain very difficult for GLS to solve. We have found that on these problems (in particular the par16* problems, the penalties of GLS often grow out of control, resulting in GLS completely controlling the search, and progress becoming very slow). To resolve this problem, we multiply the penalties of all features by a constant *pdecay* when the maximum penalty of a feature becomes greater than a constant *pmax*. This allows us to limit the size of the penalties and stop them becoming too large. The value of *pdecay* should be close to 1.0 (we used 0.8 throughout this paper), so that the values of old penalties are not completely lost. The value of *pmax* should be large enough to allow GLS to escape from basins and plateaus present in the problem search landscape, but small enough to ensure that the size of the penalties is still kept under control. This scheme is similar to that employed by Frank (1997) who multiplies the weights in his algorithm by a decay constant after every local search move is performed and later by Shang & Wah (1998) who multiply their lagrangian multipliers by a fraction, every N iterations of their algorithm. Voudouris & Tsang (1998b) use a queue scheme to limit the exact number of penalties imposed in the Radio Link Frequency Problem in conjunction with GLS. Our scheme differs from these, however, in that it uses the maximum penalty imposed so far to decide when to decrease the penalties.

Pseudo code for the overall algorithm including GLS is given in Figure 3. Line 1 reduces the original SAT problem, by propagating all unit clauses, using unit propagation (as described in section 1.3). Line 2 sets all the penalties to 0, and line 3 picks a random assignment for the local search algorithm. Lines 6 and 7 call the local search algorithm, whilst lines 8 and 9 penalise the solution returned by the local search algorithm. Lines 10,11 and 12 keep the penalties within sensible bounds, as described above. The algorithm terminates (lines 14-16) when either all clauses are satisfied or some other termination criteria (such as maximum number of iterations executed or maximum CPU time used) are met.

```
Function GLSSAT(s, g, λ, smax, pmax, pdecay)
{
    perform unit propagation on all unit clauses, as described in section XXX
    for each f_i, p_fi = 0
    s = random assignment
    do
    {
        h = g augmented as in equation (3)
        s =LocalSATSearch(s, h, smax)
        For each feature, f_i with maximum utility, Util(s,f_i),
            p_fi = p_fi + 1
        If (maximum penalty > pmax)
            For each p_fi
                p_fi = p_fi * pdecay
    }
    while (h(s) > 0 or termination condition)

    return s
}
```

Figure 2: Psuedo code for GLSSAT

2. EXPERIMENTAL RESULTS

To test whether adding GLS and other heuristics for improving local search performance really gave an overall benefit to the algorithm, when applied to solving SAT problems, we ran several variants of our local search algorithm on a selection of various types of problems from the DIMACS benchmark archive (these problems are publicly available from ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf), averaged over 10 runs. For all experiments, *smax* was set to 2, λ was set to 1.0 and *pdecay* to 0.8 (All experiments were carried out on a Pentium II 300Mhz PC with 256MBs of RAM running Windows NT 4.0, and implemented in C++, compiled by MS Visual C++ 6.0).

2.1 Local search without GLS

When local search was run without GLS, with or without the history heuristic it had great difficulty (in fact in most cases it did not within 1000000 repairs from random starting points) in solving the majority of *easier* problems in the DIMACS benchmark archive. For this reason, there is little point in reporting these results here, since they only show that some form of meta-heuristic for escaping local minima and plateaus is necessary for solving these problems successfully in a reasonable time.

2.2 GLS plus local search with & without the history heuristic

In this section we evaluate the effect of adding the running local search, with GLS, choosing between moves randomly and choosing the least recent variable to flip, when offered a choice. This was to check that the history mechanism also improved the performance of a local search algorithm with GLS, as well as improving variants of GSAT. For these experiments *pmax* was set to infinity, since the problems we used for these tests, are easily soluble without penalty decay. We allowed both variants of our algorithm, a maximum of 100000 moves on the each problem to find a solution. For each problem we ran both variants 10 times, from random starting points. The average success rate, average CPU time and average number of moves to find a solution, for each class of problem are shown in Table 1.

	GLSSAT without history mechanism			**GLSSAT with history mechanism**		
Problem group	**Average success rate (%)**	**Average CPU Seconds**	**Average number of moves**	**Average success rate (%)**	**Average CPU Seconds**	**Average number of moves**
ssa*	100.00%	0.10	7180.03	100.00%	0.10	6012.08
as*	100.00%	0.13	669.97	100.00%	0.12	532.73
ii*	100.00%	0.15	937.91	100.00%	0.22	1113.20
tm*	100.00%	0.20	652.25	100.00%	0.22	662.70
aim*	92.71%	0.22	16641.67	97.29%	0.17	12216.04
jnh*	100.00%	0.12	2868.33	100.00%	0.09	1839.06
par8*	85.00%	0.29	24634.38	90.00%	0.25	20711.94
Average	**96.82%**	**0.17**	**7654.93**	**98.18%**	**0.17**	**6155.39**

Table 1 : GLSSAT with and without the history mechanism

Obviously, as can be seen from the results, the history mechanism increases the success rate of GLSSAT, as well as reducing the average number of repairs required to find a solution. However, this is not the main reason why GLSSAT is so successful in solving these problems, since GLS with the local search algorithm choosing between possible moves at random performs very well as well.

	GLSSAT with history mechanism			**WalkSAT**		
Problem	**Average success rate (%)**	**Average CPU Seconds**	**Average number of moves**	**Average success rate (%)**	**Average CPU Seconds**	**Average number of moves**
ssa7552-038	100.00%	0.21	14269	70.00%	0.33	48215
ssa7552-158	100.00%	0.05	2440	100.00%	0.10	31473
ssa7552-159	100.00%	0.06	3144	100.00%	0.13	36598
ssa7552-160	100.00%	0.08	4196	100.00%	0.13	39552

Table 2: GLSSAT verses WalkSAT

To give an idea of GLSSAT's performance against other local search algorithms for the SAT problem, we compare GLSSAT against WalkSAT version 35, run on the same PC platform as GLSSAT, compiled with the same compiler, and run with maxflips (the number of maximum number of moves per run) set to 100000 and maxtries (the maximum number of runs) set to 10. This shows that GLSSAT performs better than WalkSAT in all cases, on these circuit diagnoses based SAT problems. These problems were first reported as solved by WalkSAT in Selman *et al* (1994), although we decided to rerun the experiments to give comparable average CPU times, and also so we could find out the average number of moves taken to find a solution for an implementation independent comparison.

2.3 Preliminary results with bounded penalties

In this section we give preliminary results of using bounded penalties, to help GLSSAT solve some of the easier par8* problems faster and more reliably, and solve some of the par16-*-c problems at all. We ran GLSSAT 10 times on each problem and allowed a maximum of 100,000,000 moves on the par16-*-c problems and 100,000 moves on the par 8-* problems. The parameters used for bounded penalties were *pmax* = 10 and *pdecay* = 0.8. The other parameters remained the same as for previous problems. These results are shown in Table 3.

	GLS with bounded penalties					GLS without bounded penalties				
		CPU time in seconds		Number of moves			CPU time in seconds		Number of moves	
Problem	Success rate (%)	Average	Standard deviation	Average	Standard deviation	Success rate (%)	Average	Standard deviation	Average	Standard deviation
par8-1-c	100%	0.04	0.06	2049	3473	100%	0.01	0.02	1033	1287
par8-2-c	100%	0.01	0.02	827	998	100%	0.03	0.05	1620	3305
par8-3-c	100%	0.06	0.10	3537	5441	90%	0.28	0.48	15897	26672
par8-4-c	100%	0.04	0.05	2293	2919	100%	0.05	0.08	3316	4839
par8-5-c	100%	0.16	0.23	7774	11326	90%	0.34	0.60	19162	34157
par8-1	100%	0.08	0.15	5239	9234	100%	0.20	0.44	13212	29944
par8-2	100%	0.23	0.30	13573	17719	100%	0.41	0.59	27793	39266
par8-3	100%	0.26	0.35	15560	21168	80%	0.28	0.49	17808	31899
par8-4	100%	0.26	0.31	15791	18864	90%	0.16	0.32	10483	20517
par8-5	100%	0.48	0.64	27656	37025	50%	0.15	0.24	9825	16476
par16-1-c	100%	183.88	249.67	10069383	13770707	NA	NA	NA	NA	NA
par16-2-c	100%	366.82	583.16	20408673	32353608	NA	NA	NA	NA	NA
par16-3-c	90%	580.16	778.64	32216329	43228577	NA	NA	NA	NA	NA
par16-4-c	100%	861.41	1293.49	25008850	38266987	NA	NA	NA	NA	NA
par16-5-c	100%	543.07	688.00	14206929	17993133	NA	NA	NA	NA	NA

Table 3: GLS with and without bounded penalties on parity problems

Generally, the results in table 3, show that on difficult problems, bounded penalties help to improve the reliability in solving problems, and also improve the time to find a solution on these problems. However, in a few cases, GLS with bounded penalties performs slightly worse (for example par8-1-c) with respect to the average number of moves required to find a solution.

Warners and van Maaren (1998) have recently shown these problems to be easily soluble using a two phase algorithm, where they use linear programming to extract conjunctions of equivalencies from the original SAT problem, and then use a modified Davis-Puttnam algorithm to solve the resulting problem, so possibly modifying local search algorithms in this way in the future would be a better approach to solving these particular problems. These problem instances are still worth studying though, as there must be other cases where it is difficult (e.g. for much larger problems) to extract the conjunctions of equivalencies due to the size of the resulting LP problem. As far as local search algorithms are concerned, our results on these parity learning instances of SAT problems are not as good as those of Wu and Wah (1999), although our algorithm has far fewer parameters which require tuning than theirs and we could probably improve it's performance, if we tuned these parameters to these particular problems.

3. CONCLUSION

In this paper we have presented results showing the progress of applying GLS to the SAT problem. We have shown that using a history based mechanism similar to that described in Gent & Walsh (1993) we can also increase the performance of GLS, although only slightly, just as they do with their GSAT variants. In addition to this, we report work in progress that shows a new scheme for keeping the penalties under control on difficult problems can be used successfully with GLS, so that hard SAT problems, which could not be solved using GLS, without such a mechanism, can now be solved. Future work requires finding good heuristics for automatically setting parameters, which GLS and GLSSAT requires, and the possible extension of GLSSAT to include some mechanism for dealing with conjunctions of equivalencies within local search algorithms. Finally, we speculate that many other local search algorithms and problems will benefit from GLS in the future, where a suitable local search algorithm and set of features can be defined.

ACKNOWLEDGEMENTS

The authors would like to thank Dick Williams, John Ford, James Borrett, Christos Voudouris and Nathan Barnes for their helpful comments and suggestions and also Benjamin Wah for his comments on a related paper apply GLS to weighted MAX-SAT problems. This project is partly funded by EPSRC grant GR/L20122.

REFERENCES

Davis, M. and Putnam, H. (1960) A computing procedure for quantification theory. In **Journal of the ACM Vol. no. 7**, pp. 201-215

Frank, J. (1997) Learning Short-Term Weights for GSAT. In **Proceedings IJCAI '97**, Volume 1, pages 384-389, 1997.

Garey, M.R. and Johnson, D.S. (1979) Computers and intractibility, W.H.Freeman and Company.

Gent, I. and Walsh, T.(1993) Towards an Understanding of Hill-climbing Procedures for SAT. In **Proceedings of AAAI-93**.

Gent, I. and Walsh, T. (1995) Unsatisfied Variables in Local Search. In **Hybrid Problems, Hybrid Solutions**, ed. J. Hallam, IOS Press, Amsterdam, 1995, pp 73-85 (Proceedings of AISB-95).

Glover, F. (1989) Tabu search Part I. Operations Research Society of America (ORSA), **Journal on Computing, Vol. 1**, 109-206.

Glover, F. (1990) Tabu search Part II. Operations Research Society of America (ORSA), **Journal on Computing, Vol. 2**, 4-32.

Gu, J. (1993) Local search for Satisfiability (SAT) Problem. In **IEEE Transactions on Systems, Man and Cybernetics, Vol. 23**, No. 4, July/August 1993.

Gu, J. (1994) Global Optimization for Satisfiability (SAT) Problem. In **IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 3**, June 1994, pages 361-381.

Jiang Y., Kautz H., and Selman B. (1995) Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT. **1st International Joint Workshop on Artificial Intelligence and Operations Research**.

Kautz, H., McAllester, D. and Selman, B. (1997) Exploiting Variable Dependency in Local Search. DRAFT. Abstract appears in Abstracts of the Poster Sessions of **IJCAI-97**, Nagoya, Japan, 1997.

Kilby, P., Prosser, P. and Shaw, P. (1997) Guided Local Search for the Vehicle routing Problem. In **Proceedings of the 2nd International Conference on Metaheuristics**, July 1997.

Lau, T.L. and Tsang, E. P. K. (1998a) Solving the generalized assignment problem with the guided genetic algorithm. In Proceedings of **10th IEEE International Conference on Tools for Artificial Intelligence**, 1998.

Lau, T.L. and Tsang, E. P. K. (1998b) Guided Genetic algorithm and its application to the large processor configuration problem. In Proceedings of **10th IEEE International Conference on Tools for Artificial Intelligence**, 1998.

McAllester, D., Selman, B. and Kautz, H. (1997) Evidence for Invariants in Local Search. In **Proceedings AAAI-97**, Providence, RI, 1997.

Minton, S., Johnson M.D., Philips A.B. and Laird P. (1992) Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. In **Artificial Intelligence 58** (1992), pp161-205.

Mills, P. and Tsang, E. (1999), Solving the MAX-SAT problem using Guided Local Search. Technical Report CSM-327, University of Essex, Colchester, UK.

Morris, P. (1993) The breakout method for escaping from local minima. In **Proceedings of AAAI-93**, pp. 40-45. AAAI Press/The MIT Press.

Richards, T. and Richards, B. (1998) Non-systematic search and learning: an emperical study. In **Lecture Notes in Computer Science 1520**, Maher, M. and Puget J-F. (eds.). (Proceedings of 4th International Conference on Principles and Practice of Constraint Programming), Springer Verlag, pp370-384.

Selman, B., Levesque, H. and Mitchell, D. (1992) A New Method for Solving Hard Satisfiability Problems. In **Proceedings AAAI-92**, 1992.

Selman, B. and Kautz, H.(1993) An Empirical Study of Greedy Local Search for Satisfiability Testing. In **Proceedings AAAI-93**, 1993.

Selman, B. and Kautz, H. (1993) Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems. In **Proceedings IJCAI-93**, 1993.

Selman, B., Kautz, H. and Cohen, B. (1994) Noise Strategies for Improving Local Search. In **Proceedings AAAI-94**, 1994.

Selman, B., Kautz H. and McAllester, D. (1997) Ten Challenges in Propositional Reasoning and Search. In **Proceedings of the Fifteenth International Conference on Artificial Intelligence (IJCAI-97)**, NAGOYA, Aichi, Japan, 1997.

Shang, Y. (1997) Global Search Methods for Solving Nonlinear Optimization Problems. Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, August 1997.

Shang, Y. and Wah, B.W. (1998) A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems. In **Journal of Global Optimization**, Kluwer Academic Publishers, vol. 12, no. 1, Jan. 1998, pp. 61-99.

Tsang, E. (1993) Foundations of Constraint Satisfaction. Academic Press, 1993.

Tsang E. and Voudouris C. (1997) Fast local search and guided local search and their application to British Telecoms workforce scheduling problem. In **Operations Research Letters 20** (1997), 119-127.

Voudouris, C. (1997) Guided Local Search for Combinatorial Optimisation Problems, Ph.D. thesis. Department of Computer Science, University of Essex, 1997.

Voudouris, C. (1998) Guided Local Search - an illustrative example in function optimisation. In **BT Technology Journal, Vol. 16**, No. 3, July 1998, pp46-50.

Voudouris, C. and Tsang, E.P.K. (1998a) Guided local search and its application to the traveling salesman problem. In **European Journal of Operational Research, Vol.113, Issue 2**, November 1998, pp469-499.

Voudouris, C. and Tsang, E.P.K. (1998b) Solving the Radio Link Frequency Assignment Problem using Guided Local Search. In **Proceedings, NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace)**, Aalborg, Denmark, October 1998.

Wah, B.W. and Shang, Y. (1997) Discrete Lagrangian-Based Search for Solving MAX-SAT Problems. In **15th International Joint Conference on Artificial Intelligence**, 1997, pp378-383.

Warners, J.P. and van Maaren, H. (1998) A Two Phase Algorithm for Solving a Class of Hard Satisfiability Problems. Report SEN-R9802 (CWI). To appear in **Operations Research Letters 23** (1999), pp. 81-88.

Wu, Z. (1998) The Discrete Langrangian Theory and its Application to Solve Nonlinear Discrete Constrained Optimization Problems. M.Sc. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1998.

Wu, Z. and Wah, B.W. (1999) Trap Escaping Strategies in Discrete Langrangian Methods for Solving Hard Satisfiability and Maximum Satisfiability Problems. To appear in **Proceedings AAAI-99**.