To be submitted to
DIMACS Workshop on Constraint Programming and Large Scale Discrete Optimisation,
Rutgers, New Jersey, USA, September 1998

Guided Local Search Joins the Elite in Discrete Optimisation

Christos Voudouris
Intelligent Systems Research Group,
Advanced Research & Technology Dept.
BT Laboratories, British Telecommunications plc.
United Kingdom
Email: chrisv@info.bt.co.uk

Edward Tsang
Department of Computer Science, University of Essex
United Kingdom
Email: edward@essex.ac.uk

August 1998

Abstract

Developed from constraint satisfaction as well as operations research ideas, Guided Local Search (GLS) and Fast Local Search are novel meta-heuristic search methods for constraint satisfaction and optimisation. GLS sits on top of other local-search algorithms. The basic principle of GLS is to penalise features exhibited by the candidate solution when a local-search algorithm settles in a local optimum. Using penalties is an idea used in operations research before. The novelty in GLS is in the way that features are selected and penalised. FLS is a way of reducing the size of the neighbourhood. GLS and FLS together have been applied to a non-trivial number of satisfiability and optimisation problems and achieved remarkable result. One of their most outstanding achievements is in the well-studied travelling salesman problem, in which they obtained results as good as, if not better than the state-of-the-art algorithms. In this paper, we shall outline these algorithms and describe some of their discrete optimisation applications.

1. Introduction

Constraint satisfaction [Tsang 1993, Freuder & Mackworth 1994, Marriott & Stuckey 1998] is a very general problem that is required in many real life problems. Due to its generality, much research effort has been spent in this area in recent years. This has led to technological break-through as well as commercial exploitation. Sound commercially-available systems have been built, e.g. see ILOG Solver [Puget 1995], CHIP [Simonis 1995], ECLiPSe [Lever el. al. 1995] and Prolog IV [Colmerauer 1990]. Constraint programming is now a multi-

million Pounds business; see [Cras 1993] [Wallace 1996] [Zweben & Fox 1994] for some of their applications.

Most real life constraint optimisation problems are too complex for systematic search methods. In recent years, stochastic methods have received great attention [Freuder *et. al*. 1995]. This paper describes the results of a research programme which has led to successful applications of stochastic constraint satisfaction techniques to optimisation.

2. Constraint satisfaction related to discrete optimisation

Many problems involve constraint satisfaction. A constraint satisfaction problem (CSP) comprises three elements:

(Z, D, C)

where Z is a finite set of variables; D is a function that maps every variable *x* in Z to a set of objects (of any type), which is called the *domain* of *x*. Most research in constraint satisfaction deal with discrete and finite domains. Z is a set of constraints, which may take any form, which restricts the values that variables may take simultaneously. The task is to assign one value to each variable satisfying all the constraints [Tsang 1993]. Constraint satisfaction is a general problem which has been applied to a wide variety of problems [Freuder & Mackworth 1994, Wallace 1996].

In many constraint satisfaction problems, some solutions are "better" than others, where "better" is defined by some domain-dependent objective functions. The task in such problems is to find the optimal (minimum or maximum) solution. In other problems, constraints are classified as *hard* and *soft* constraints. Hard constraints are not to be violated in any case. Soft constraints can be violated at certain costs. In some problems, assigning different values to different variables involve different utilities. The task is to minimise the cost of maximise utilities of the solution. Finite constraint satisfaction problems that involve optimisation are basically discrete optimisation problems traditionally studied in Operations Research. There

has been cross-fertilisation between the two fields. The work described in this paper started from constraint satisfaction and benefited from ideas in operations research.

3. Background: Hill-climbing

3.1. Basic Principles of Hill-climbing

Due to their combinatorial explosion nature, many real life constraint optimisation problems are hard to solve using complete methods such as *branch & bound* [Hall 1971, Reingold *et. al*. 1977]. One way to contain the combinatorial explosion problem is to sacrifice completeness. Some of the best known methods that use this strategy are local search methods, the basic form of which are often referred to as hill-climbing.

To perform hill-climbing, one must define the following:

(a) a representation for candidate solutions;
(b) an objective function: given any candidate solution, this function returns a numerical value. The problem is seen as an optimisation problem according to this objective function (which is to be minimized or maximized);
(c) a neighbourhood function that maps every candidate solution *x* (often called a *state*) to a set of other candidate solutions (which are called *neighbours* of *x*).

Hill-climbing works as follows: starting from a candidate solution, which may be randomly or heuristically generated, the search moves to a neighbour which is 'better' according to the objective function (in a minimization problem, a better neighbour is one which is mapped to a lower value by the objective function). The search terminates if no better neighbour can be found, or resources run out. The whole process can be repeated from different starting points.

One of the main problems with hill-climbing is that it may settle in local optima — states that are better than all their neighbours but not necessarily the best possible solution. To overcome that, methods such as *Simulated Annealing* [Aarts & Korst 1989, Davis 1987, Otten & van Ginneken 1977] and *Tabu Search* [Glover *et. al*. 1989, 1990, 1993] have been proposed.

3.2. Example of hill-climbing: the travelling salesman problem

The *travelling salesman problem* (TSP) is a well-known optimisation problem. Given a number of cities and the distances between them, the task is to find a tour that visits all cities. The objective is to find the tour with the shortest distance travelled.

One way to hill-climb on a TSP with *n* cities is to represent a candidate solution by a sequence of *n* variables where variable *i* represents the *i*-th city to be visited in the tour. For example, a tour through 10 cities may be:

1 4 5 8 2 7 6 9 3 10

The objective function is the total distances to be travelled in a given tour. One simple but reasonably effective neighbourhood function is *2-Opt*. Effectively what it does is to pick a subsequence and reverse the order of the cities. For example, applying 2-Opt to the above tour, may get:

1 4 5 9 6 7 2 8 3 10

In other words, the subsequence 8-2-7-6-9 is reversed. This tour qualifies to be accepted by hill-climbing if the total travelling distance incurred is shorter than that in the previous tour. When many neighbours are 'better' than the previous tour, one may choose a random one. Alternatively, heuristics may be applied to select among the qualified neighbours. For example the *steepest descent* heuristic will select the neighbour that incurs the least travelling distance.

4. Fast Local Search (FLS)

One factor which limits the efficiency of a hill-climbing algorithm is the size of the neighbourhood. If there are many neighbours to consider, then if the search takes many steps to reach a local optima, and/or each evaluation of the objective function requires a nontrivial

amount of computation, then the search could be very costly. Bentley [1992] presented the *approximate 2-Opt* method to reduce the neighbourhood of 2-Opt in the TSP. We generalised this method to a method that we call *Fast Local Search* (FLS). The intention is to, guided by heuristics, ignore neighbours that are unlikely to lead to fruitful hill-climbs in order to improve the efficiency of a search.

Here we shall use the TSP to show how FLS can be applied to the 2-Opt neighbourhood function. An *activation bit* is associated to each transition position in the tour (e.g. transition between the third and fourth positions in the tour). All activation bits are switched on at start. Only positions with an *on* activation bit will be examined to see if it can make an improvement move. If no improvement is possible, then this bit is switched off. It will only be switched on again under two conditions:

(1) if a 2-Opt step (initiated by another position) is made which inverts a subsequence which ends in this position. For example, if the subsequence between the fourth and the sixth cities were reversed, then the activation bit for both transitions third-to-fourth and sixth-to-seventh will be switched on.
(2) if this transition is a feature that is penalised (to be explained when we introduce GLS later).

To generalise this to neighbourhood functions other than the 2-Opt, one may define features for candidate solutions. Selecting such features in an application is not difficult because the objective function is often made up of a number of features in the candidate solutions.

By reducing the size of the neighbourhood, one may significantly reduce the amount of computation involved in each hill-climbing step. The hope is to enable more hill-climbing steps in a fixed amount of time. The danger of ignoring certain neighbours is that some improvements may be missed. The hope is that the gain out-weighs the loss. We found that FLS combined extremely well with GLS.

5. Guided Local Search (GLS)

Guided local search (GLS) is a meta-heuristic algorithm which aim is (like simulated annealing and tabu search) to help hill-climbing to escape local optima. The basic idea is to augment the objective function with penalties, which directs the search away from local optimum. GLS was built upon our experience in a connectionist method called GENET (which stands for "Generic Network") [Wang & Tsang 1991, Tsang & Wang 1992, Davenport et. al. 1994] as well as penalty ideas from operations research [Koopman 1957, Stone 1983, Luenberger 1984].

GLS is an algorithm for modifying the behaviour of hill-climbing. To apply GLS, one has to define *features* for the candidate solutions. For example, in the travelling salesman problem, a feature could be "whether the candidate tour travels immediately from city A to city B".

GLS associates *costs* and *penalty* to each feature. The costs should normally take their values from the objective function. For example, in the travelling salesman problem, the cost of the above feature is the distance between cities A and B. The penalties are initialized to 0 and will only be increased when the local search reaches local optimum. This will be elaborated below.

Given an objective function g that maps every candidate solution s to a numerical value, we define a function h which will be used by hill-climbing (replacing g).

$$h(s) = g(s) + \lambda \times \Sigma(p_i \times I_i(s)) \quad (1)$$

where s is a candidate solution, λ is a parameter to the GLS algorithm, i ranges over the features, p_i is the penalty for feature i (all p_i's are initialized to 0) and I_i is an indication of whether s exhibits feature i:

$$I_i(s) = 1 \text{ if } s \text{ exhibits feature } i\text{; 0 otherwise.} \quad (2)$$

When the local search settles on a local optimum, the penalty of some of the features associated to this local optimum is increased (to be explained below). This has the effect of

changing the objective function (which defines the "*landscape*" of the local search) and driving the search towards other candidate solutions. The key to the effectiveness of GLS is in the way that penalties are imposed. It is worth pointing out that a slight variation in the way that penalties are managed could make all the difference to the effectiveness of a local search.

Our intention is to penalize "bad features", or features which "matter most", when a local search settles in a local optima. The feature that has high cost affects the overall cost more. Another factor that should be considered is the current penalty value of that feature. We define the utility of penalizing feature *i*, $util_i$, under a local optimum *s**, as follows:

$$util_i(s^*) = I_i(s^*) \times c_i / (1 + p_i) \quad (3)$$

where c_i is the cost and p_i is the current panelty value of feature i. In other words, if a feature is not exhibited in the local optimum, then the utility of penalizing it is 0. The higher the cost of this feature (c_i), the greater the utility of penalizing it. Besides, the more times that it has been penalized, the lower the utility of penalizing it again.

In a local optimum, the feature(s) with the greatest *util* value will be *penalized*. This is done by incrementing its penalty value by 1:

$$p_i = p_i + 1 \quad (4)$$

By taking cost and the current penalty into consideration in selecting the feature to penalize, we are distributing the search effort in the search space. Candidate solutions which exhibit "good features", i.e. features involving lower cost, will be given more effort in the search, but penalties help to prevent all effort be directed to the best features. The idea of distributing search effort, which plays an important role in the success of GLS, is borrowed from Operations Research, e.g. see Koopman [1957] and Stone [1983]. Following we shall describe the general GLS procedure:

Procedure **GLS** (input: an objective function *g;* a local search strategy ***L***; features and their costs; parameter λ)

1. Generate a starting candidate solution randomly or heuristically;
1. Initialize all the penalty values (p_i) to 0;
2. Repeat the following until a termination condition (e.g. a maximum number of iterations or time limit) has been reached:
   3.1. Perform local search (using ***L***) according to the function *h* (which is g plus the penalty values, as defined in (1) above) until a local optimum *M* has been reached;
   3.2. For each feature *i* which is exhibited in *M* compute $util_i = c_i / (1 + p_i)$
   3.3. Penalize every feature *i* such that $util_i$ is maximum: $p_i = p_i + 1$;
4. Return the best candidate solution found so far according to the objective function *g*.

6. Applications of GLS and FLS in discrete optimisation problems

GLS and FLS have been applied to a number of discrete optimisation problems. They have been applied to *radio link frequency assignment problem* (RLFAP) [Bouju *et. al.* 1995] and British Telecom's *work force scheduling* problem (WFS) [Baker 1993, Azarmi & Abdul-Hameed 1995]. In the RLFAP, the task is to assign available frequencies to communication channels satisfying constraints that prevent interference. In some RLFAPs, the goal is to minimize the number of frequencies used. GLS+FLS reported the best results when it was published [Voudouris & Tsang 1996].

In British Telecom's WFS, the task is to assign technicians from various bases to serve various jobs, which may include customer requests and repairs, at various locations. Customer requirements and working hours restrict the times that certain jobs can be served by certain technicians. The objective is to minimize a function which take into consideration the travelling cost, overtime cost and unserved jobs. In the WFS, GLS+FLS still holds the best-published results in the benchmark problem known to the authors [Tsang & Voudouris 1997].

The most significant results of GLS and FLS are probably in their application to the travelling salesman problem (TSP). The Lin-Kernighan algorithm (LK) is a specialised

algorithm for TSP that has long been perceived as the champion of this problem [Lin & Kernighan 1973, Martin & Otto 1996]. We tested GLS+FLS+2Opt against LK in a set of benchmark problems from the public TSP library [Reinelt 1991]. Given the same amount of time (we tested 5 cpu minutes and 30 cpu minutes on a DEC Alpha 3000/600), GLS+FLS+2Opt found better results than LK in average. GLS+FLS+2Opt also out-performed the best Simulated Annealing [Johnson], Tabu Search [Knox 1994] and Genetic Algorithm [Freisleben & Merz 1996] methods reported on TSP so far. One must be cautious when interpreting such empirical results as they could be affected by many factors, including implementation issues [Hooker 1995]. But given that the TSP is an extensively studied problem, it takes something special for an algorithm to out-perform the champions under any reasonable measure ("*find me the best results within a given amount of time*" must be a realistic requirement). It must be emphasized that LK is specialized for TSP but GLS and FLS are much simpler general-purpose algorithms. Details of GLS+FLS applied to the TSP can be found in [Voudouris & Tsang 1998].

GLS has also been applied to general function optimisation problems to illustrate that artificial features can be defined for problems in which the objective function suggests no obvious features. Results show that, as expected, GLS spreads its search effort across solution candidates depending on their quality (as measured by the objective function). Besides, GLS consistently found solutions in a landscape with many local sub-optimals [Voudouris & Tsang 1995].

7. Guided Genetic Algorithm: an extension of GLS

GLS is developed as a meta-heuristic algorithm. Apart from sitting it on top of local search algorithms, one can put it into *Genetic algorithms* (GAs). GAs borrow their ideas from evolution [Holland 1975, Davis 1987, 1991, Goldberg 1989]. The idea is to maintain a set of candidate solutions. Individuals are given different chances to produce offspring depending on their "*fitness*". Applied to optimisation, *fitness* is measured by the objective function. GAs

have been applied to constraint satisfaction [Eiben *et. al.* 1994, Ruttkay *et. al.* 1995] and demonstrated promising in discrete optimisation [Warwick & Tsang 1994, 1995].

Guided Genetic Algorithm (GGA) is a hybrid of GA and GLS. It can be seen as a GA with GLS to bring it out of local optimum: if no progress has been made after a number of iterations (this number is a parameter to GGA), GLS modifies the fitness function (which is the objective function) by means of penalties. GA will then use the modified fitness function in future generations. The penalties are also used to bias crossover and mutation in GA – genes that are involved in more penalties are made more susceptive to changes by these two GA operators. This allows GGA to be more focussed in its search.

On the other hand, GGA can roughly be seen as a number of GLS searches from different starting points running in parallel, exchanging material in a GA manner. The difference is that only one set of penalties is used in GGA whereas parallel GLS would have used one independent set of penalties per run. Besides, learning in GGA is more selective than GLS: the updating of penalties is only based on the best chromosome found at the point of penalization.

GGA has been found to be robust, in the sense that solutions found by GGA were as good as GLS (not surprising, as GGA was built upon GLS), but solution costs fall into a narrower range. GGA has been applied to the Processors Configuration Problem [Lau & Tsang 1997, 1998a], General Assignment Problem [Lau & Tsang 1998b] and the Radio Length Frequency Assignment Problem [Lau & Tsang, to appear] with excellent results. Details of GGA and its applications will be reported in another occasion.

Acknowledgements

The authors would like to thank Chang Wang, Jim Doran, Andrew Davenport, Kangmin Zhu, James Borrett, John Ford, Patrick Mills, Terry Warwick, Alvin Kwan, Richard Williams, T L Lau and Paul Scott for their contribution to this project. This project was partially sponsored by EPSRC funded projects GR/H75275 and GR/L20122.

References

1. Aarts E. & Korst J., Simulated Annealing and Boltzmann Machines, John Wiley & Sons, 1989
2. Aho A.V., Hopcroft J.E., & Ullman J.D., Data structures and algorithms, Addison-Wesley, 1983
3. Azarmi N. and Abdul-Hameed W., Workforce scheduling with constraint logic programming, British Telecom Technology Journal, Vol.13, No.1, January, 1995, 81-94
4. Baker S., Applying simulated annealing to the workforce management problem, ISR Technical Report, British Telecom Laboratories, Martlesham Heath, Ipswich, 1993
5. Bentley J.J., Fast algorithms for geometric traveling salesman problems, ORSA Journal on Computing, Vol.4, 1992, 387-411
6. Bouju, A., Boyce, J.F., Dimitropoulos, C.H.D., vom Scheidt, G. & Taylor, J.G., Intelligent search for the radio link frequency assignment problem, Proceedings of the International Conference on Digital Signal Processing, Cyprus 1995
7. Colmerauer, A., An introduction to Prolog III, CACM Vol.33, No7, July 1990, 69-90
8. Cras, J-Y., A review of industrial constraint solving tools, AI Perspective Series, AI Intelligence, Oxford, UK, 1993
9. Davenport A., Tsang E.P.K., Wang C.J. and Zhu K., GENET: a connectionist architecture for solving constraint satisfaction problems by iterative improvement, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 325-330
10. Davenport, A., Extensions and evaluation of GENET in constraint satisfaction, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, July 1997
11. Davis L. (ed.), Genetic algorithms and simulated annealing, Research notes in AI, Pitman/Morgan Kaufmann, 1987.
12. Davis L. (ed.), Handbook of genetic algorithms, Van Nostrand Reinhold, 1991
13. Eiben A.E., Raua P-E., and Ruttkay Zs., Solving constraint satisfaction problems

using genetic algorithms, Proc., 1st IEEE Conference on Evolutionary Computing, 1994, 543-547

14. Freisleben, B., and Merz, P., A Genetic Local Search Algorithm for Solving the Symmetric and Asymmetric TSP, Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 616-621, 1996

15. Freuder, E.C. & Mackworth, A., (ed.), Constraint-based reasoning, MIT Press, 1994

16. Freuder, E.C., Dechter, R., Ginsberg, M., Selman, B. & Tsang, E., Systematic versus stochastic constraint satisfaction, Panel Paper, in Mellish, C. (ed.), Proc., 14th International Joint Conference on AI, Montreal, Canada, August, 1995, 2027-2032F.

17. Glover, Tabu search Part I, ORSA Journal on Computing 1, 1989, 109-206

18. Glover F., Tabu search Part II, ORSA Journal on Computing 2, 1990, 4-32

19. Glover F., E. Taillard, and D. de Werra, A user's guide to tabu search, Annals of Operations Research, Vol.41, 1993, 3-28

20. Goldberg D.E., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Pub. Co., Inc., 1989

21. Hall P.A.V., Branch and bound and beyond, Proc. International Joint Conference on AI, 1971, 641-650

22. Holland J.H., Adaptation in natural and artificial systems, University of Michigan press, Ann Arbor, MI, 1975

23. Hooker, J.N., Testing heuristics: we have it all wrong, Journal of Heuristics, Vol.1, No.1, 1995, 33-42

24. Johnson, D., Local optimization and the traveling salesman problem, Proceedings of the 17th Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science, 443, 446-461, Springer-Verlag, 1990

25. Johnson, D., Aragon, C., McGeoch, L., and Schevon, C., Optimization by simulated annealing: an experimental evaluation, part I, graph partitioning, Operations Research, 37, 865-892, 1989

26. Knox, J., Tabu Search Performance on the Symmetric Traveling Salesman Problem, Computers Ops Res., Vol. 21, No. 8, pp. 867-876, 1994

27. Koopman B.O., The theory of search, part III, the optimum distribution of searching effort, Operations Research, Vol.5, 1957, 613-626

28. Lau, T.L. & Tsang, E.P.K., Solving the processor configuration problem with a mutation-based genetic algorithm, International Journal on Artificial Intelligence Tools (IJAIT), http://www.wspc.com.sg/journals/journals.html, World Scientific, Vol.6, No.4, December 1997, 567-585

29. Lau, T.L. & Tsang, E.P.K., Solving large processor configuration problems with the guided genetic algorithm, IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98), Taiwan, November 1998

30. Lau, T.L. & Tsang, E.P.K., The guided genetic algorithm and its application to the

general assignment problem, IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98), Taiwan, November 1998

31. Lau, T.L. & Tsang, E.P.K., Guided Genetic Algorithm and its application to the Radio Link Frequency Allocation Problem, manuscript, submitted to Journal of Constraints, Special Issue on Biocomputing, 1998

32. Lawler E.W. and Wood D.E., Branch-and-bound methods: a survey, Operations Research 14, 1966, 699-719

33. Lever J., Wallace M., and Richards B., Constraint logic programming for scheduling and planning, British Telecom Technology Journal, Vol.13, No.1., 1995, 73-80

34. Lin, S., and Kernighan, B.W., "An effective heuristic algorithm for the traveling salesman problem", Operations Research, 21, 498-516, 1973

35. Luenberger, D., Linear and nonlinear programming, Addison-Wesley Publishing Co., Inc., 1984

36. Marriott, K. & Stuckey, P.J., Programming with constraints, an introduction, MIT Press, 1998

37. Martin, O., and Otto, S.W., "Combining Simulated Annealing with Local Search Heuristics", in: G. Laporte and I. H. Osman (eds.), Metaheuristics in Combinatorial Optimization, Annals of Operations Research, Vol. 63, 1996

38. Muller C., Magill E.H., and Smith D.G., Distributed genetic algorithms for resource allocation, Technical Report, Strathclyde University, Glasgow, 1993

39. Otten R.H.J.M. and van Ginneken L.P.P.P., The annealing algorithm, Kluwer Academic Publishers, 1989

40. Puget, J-F., Applications of constraint programming, in Montanari, U. & Rossi, F. (ed.), Proceedings, Principles and Practice of Constraint Programming (CP'95), Lecture Notes in Computer Science, Springer Verlag, Berlin, Heidelberg & New York, 1995, 647-650

41. Reinelt, G., A Traveling Salesman Problem Library, ORSA Journal on Computing, 3, 376-384, 1991

42. Reingold E. M., Nievergelt J., & Deo N., Combinatorial algorithms: theory and practice, Englewood Cliffs, N.J., Prentice hall, 1977

43. Ruttkay, Zs., Eiben, A.E. & Raue, P.E., Improving the performances of GAs on a GA-hard CSP, Proceedings, CP95 Workshop on Studying and Solving Really Hard Problems, 1995, 157-171

44. Simonis, H., The CHIP system and its applications, in Montanari, U. & Rossi, F. (ed.), Proceedings, Principles and Practice of Constraint Programming (CP'95), Lecture Notes in Computer Science, Springer Verlag, Berlin, Heidelberg & New York, 1995, 643-646

45. Stone L.D., The process of search planning: current approaches and continuing problems, Operations Research, Vol.31, 1983, 207-233

46. Tsang E.P.K., Scheduling techniques — a comparative study, British Telecom Technology Journal, Vol.13, No.1, 1995, 16-28

47. Tsang, E.P.K. & Voudouris, C., Fast local search and guided local search and their application to British Telecom's workforce scheduling problem, Operations Research Letters, Elsevier Science Publishers, Amsterdam, Vol.20, No.3, March 1997, 119-127

48. Tsang, E.P.K. & Wang, C.J., A generic neural network approach for constraint satisfaction problems, in Taylor, J.G. (ed.), Neural network applications, Springer-Verlag, 1992, 12-22

49. P. van Hentenryck, Constraint satisfaction in logic programming, MIT Press, 1989

50. Voudouris, C. & Tsang, E.P.K., Function optimization using guided local search, Technical Report CSM-249, Department of Computer Science, University of Essex, Colchester, UK, August, 1995

51. Voudouris, C. & Tsang, E.P.K., Partial constraint satisfaction problems and guided local search, Proc., Practical Application of Constraint Technology (PACT'96), London, April, 1996, 337-356

52. Voudouris, C., Guided Local Search for Combinatorial Optimisation Problems, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, July 1997

53. Voudouris C. & Tsang, E.P.K., Guided local search and its application to the traveling salesman problem, European Journal of Operational Resaerch, to appear (accepted for publication, March 1998)

54. Wang, C.J. & Tsang, E.P.K., Solving constraint satisfaction problems using neural-networks, Proceedings of IEE Second International Conference on Artificial Neural Networks, 295-299, 1991

55. Wallace, M., Practical applications of constraint programming, Journal of Constraints, Kluwer Academic Publishers, Boston, Vol.1, Nos.1&2, 1996, 139-168

56. Warwick T., and Tsang, E.P.K., Using a genetic algorithm to tackle the processors configuration problem, Proc., ACM Symposium on Applied Computing, 1994, 217-221

57. Warwick, T. & Tsang, E.P.K., Tackling car sequencing problems using a generic genetic algorithm, Evolutionary Computation, Vol.3, No.3, 1995, 267-298

58. Zweben, M. & Fox, M.S. (ed.), Intelligent scheduling, Morgan Kaufmann, San Francisco, 1994