

Guided Local Search Joins the Elite in Discrete Optimisation

Christos Voudouris and Edward Tsang

ABSTRACT. Developed from Constraint Satisfaction as well as Operations Research ideas, Guided Local Search (GLS) and Fast Local Search (FLS) 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 results. 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 the GLS and FLS algorithms and describe some of their discrete optimisation applications.

1. Introduction

Constraint satisfaction [Tsa93, FM94, MS98] is a very general problem that can be used in many real life applications. Due to its generality, much research effort has been spent in this area in recent years. This has led to technological breakthroughs as well as commercial exploitation. Sound commercially available systems have been built, e.g. see ILOG Solver [Pug95], CHIP [Sim95], ECLiPSe [LWR95] and Prolog IV [Col90]. Constraint programming is now a multi-million pound business; see [Cra93, Wal96, ZF94] for some of their applications.

Many real life constraint optimisation problems are hard to solve using systematic search methods. In recent years, stochastic methods have received great attention [FDG⁺95]. 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 deals 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 [Tsa93]. Constraint satisfaction is a general problem, which has been applied to a wide variety of domains [FM94, Wal96].

In many constraint satisfaction problems, some solutions are "better" than others, where "better" is defined by some domain-dependent objective function. 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 may result in different utilities. The task is to minimise the cost of soft constraints or maximise the utilities of assignments in the solution while satisfying all the hard constraints. 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* [LW66, Hal71, RND77]. 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 is 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 minimised or maximised);
- (c) a *neighbourhood function* which maps every candidate solution x (often called a *state*) to a set of other candidate solutions (which are called *neighbours* of x). In other words, the domain of the neighbourhood function is the set of all candidate solutions, S . Its range is the powerset of S .

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 minimisation 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 (e.g. limited time budget). 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 - solutions that are better than all their neighbours but not necessarily the best possible for the problem. To overcome that, methods such as *Simulated Annealing*

[AK89, Dav87, OvG89] and *Tabu Search* [Glo89, Glo90, GL97] 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 (i.e. closed path) which visits each city exactly once and it is of minimum length.

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 distance to be travelled in a given tour. One simple but reasonably effective neighbourhood function is that defined by the *2-Opt* move [Joh90]. Effectively what the 2-Opt move does it to pick a sub-sequence and reverse the order of the cities. For example by applying 2-Opt between the 4th to 8th city in the above tour, the result will be the following:

1 4 5 9 6 7 2 8 3 10

In other words, the sub-sequence 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 optimum, and/or each evaluation of the objective function requires a nontrivial amount of computation, then the search could be very costly. [Ben92] presented the *approximate 2-Opt* method to reduce the neighbourhood size of 2-Opt in the TSP. We generalised this method to a method that we call *Fast Local Search* (FLS). The intention is to use heuristics that enable hill climbing to ignore neighbours that are unlikely to lead to fruitful hill-climbs in order to improve the efficiency of the search.

Here we shall use the TSP to demonstrate how FLS applies to the 2-Opt neighbourhood. An *activation bit* is associated to each city in the tour. All activation bits are switched on at the start of the search process. Only cities with an *on* activation bit will be examined to determine if an improvement move can be made. When examining a city, we look for 2-Opt moves which remove one of the edges adjacent to this city. If no improving move is found, then the city's bit is switched off. It will only be switched on again under two conditions:

- (1) if a 2-Opt move is made which results in changing an edge adjacent to the city.
- (2) if there is a feature that is penalised (to be explained when we introduce GLS later).

In the general case, FLS works in the following way. A large neighbourhood is broken down into a number of small sub-neighbourhoods. In the example above, each sub-neighbourhood was defined by the set of possible 2-Opt moves which remove one of the two edges adjacent to a city. An *activation bit* is attached to each one of the sub-neighbourhoods. In the beginning of the search, all sub-neighbourhoods are active. If a sub-neighbourhood is examined and does not contain any improving moves then it becomes inactive. Otherwise, it remains active and the improving move found is performed. Depending on the move performed, a number of other sub-neighbourhoods are also activated. In particular, we activate all the sub-neighbourhoods where we expect other improving moves to appear as a result of the changes made by the move just performed. As the solution improves the process dies out with fewer and fewer sub-neighbourhoods being active until all the sub-neighbourhood bits turn to 0.

The overall procedure could be many times faster than conventional local search. The bit setting/clearing scheme encourages chains of moves that improve specific parts of the overall solution. As the solution becomes locally better the process is scaling down, examining fewer moves and saving enormous amounts of time, which would otherwise be spent, on examining predominantly bad moves.

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 enables (like simulated annealing and tabu search) hill-climbing to escape local optima [Vou97]. The basic idea is to augment the objective function with penalties, which direct the search away from local optima. GLS was built upon our experience in a connectionist method called GENET (which stands for "Generic Network") [WT91, TW92, DTWZ94] as well as penalty and search theory ideas from Operations Research [Koo57, Sto83, Lue84]. In the following, we present the basic GLS algorithm for the minimisation case but it is not difficult to see how it can be applied to maximisation problems too.

GLS is an algorithm for modifying the behaviour of local search. 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" (i.e. an edge).

GLS associates to each feature a *cost* and *penalty* parameter. 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 (i.e. edge length). The penalties are initialised to 0 and will only be increased when local search reaches a local minimum. 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).

$$(5.1) \quad h(s) = g(s) + \lambda \times \sum (p_i \times I_i(s))$$

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 initialised to 0) and I_i

is an indication of whether s exhibits feature i :

$$(5.2) \quad I_i(s) = 1 \text{ if } s \text{ exhibits feature } i; 0 \text{ otherwise.}$$

When the local search settles in a local minimum, the penalty of some of the features associated to this local minimum is increased (to be explained below). This has the effect of changing the objective function (which defines the "*landscape*" of 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.

Our intention is to penalise "unfavourable features" when a local search settles in a local minimum. 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 penalising feature i , $util_i$, under a local minimum s , as follows:

$$(5.3) \quad util_i(s) = I_i(s) \times c_i / (1 + p_i)$$

where c_i is the cost and p_i is the current penalty value of feature i . If a feature is not exhibited in the local minimum, then the utility of penalising it is 0. The higher the cost of a feature (c_i), the greater the utility of penalising it. Besides, the more times that it has been penalised, the lower the utility of penalising it again. In a local minimum, the feature(s) with the greatest $util$ value will be *penalised*. This is done by incrementing its penalty value by 1:

$$(5.4) \quad p_i = p_i + 1$$

By taking cost and the current penalty into consideration in selecting the feature to penalise, 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 since good features will be penalised less frequently. On the other hand, candidate solutions which exhibit bad features, i.e. features involving higher cost, will be given less effort in the search since bad features will be penalised more frequently. The idea of distributing search effort, which plays an important role in the success of GLS was inspired by ideas in Operations Research and in particular from the Theory of Search area, e.g. see [Koo57] and [Sto83]. 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;
- (2) Initialise all the penalty values (p_i) to 0;
- (3) Repeat the following until a termination condition (e.g. a maximum number of iterations or time limit) has been reached:
 - (a) Perform local search (using L) according to the function h (which is g plus the penalty values, as defined in 5.1 above) until a local minimum M has been reached;
 - (b) For each feature i which is exhibited in M compute $util_i = c_i / (1 + p_i)$
 - (c) Penalise 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 .

It is worth pointing out that a variation in the way that penalties are managed could make all the difference to the effectiveness of guided local search. For example, selecting the features to penalise at random, applying excessive penalties, or using a different utility function than the one described above may result in much inferior results as our own experience suggests.

Combining GLS with FLS presented in the previous section is straightforward. The key idea is to associate features to sub-neighbourhoods. The associations to be made are such that for each feature we know which sub-neighbourhoods contain moves that remove the feature from the solution. In the beginning, FLS is left to reach a local minimum (i.e. all the bits turn to 0). GLS then penalises one or more features as explained above. The sub-neighbourhoods, which are associated to the penalised features and only these, have their bits set to 1 and FLS is run again to reach a new local minimum. This process is repeated until a termination condition for GLS is satisfied. After the first run of FLS, all subsequent runs examine moves, which are associated to the penalised features trying to remove them from the working solution. In the TSP example where edges are penalised, we will set to 1 the bits of the cities at the ends of the edge(s) penalised. As a result, FLS will focus on 2-Opt moves which try to remove the penalised edge(s) speeding up significantly the GLS procedure.

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 the *Radio Link Frequency Assignment Problem* (RLFAP) [THL95, MPR98] and British Telecom's *work force scheduling problem* (WFS) [Bak93, AAH95].

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 minimise the number of frequencies used or the maximum frequency used. GLS combined with FLS reported the best results when it was published [VT96]. New and significantly improved results were reported in a recent NATO Symposium on Frequency Assignment [VT98]. In particular, for the 25 publicly available RLFAP instances, GLS managed to find the best known solution in 18 of them, improved the best known solution in 5 and found a marginally inferior solution in only 2 of the instances. Note here, that the best known solutions have been discovered by different techniques. GLS achieved a better performance than the collection of these techniques by improving the cost of the previously best known solutions by an average 1.75%. Moreover the best GLS variant for the RLFAP was found able to achieve an average excess of 1.63% over the best known solutions when multiple trial runs were performed. This proves the robustness of the approach over the whole set of test problems when compared to other search techniques which only performed well on certain types of instances, see [VT98] for details.

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 minimise a function, which takes into consideration the travelling cost, overtime cost and penalties for unallocated jobs. In the WFS, GLS+FLS still holds the best-published results in the benchmark problem known to the authors [TV97].

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 and its variants have long been perceived as the champion heuristics for this problem [LK73, Joh90, MO96]. We tested GLS+FLS+2Opt against LK and variants of LK (which use the double bridge move) [Joh90] in a set of benchmark problems from the public TSP library [Rei91]. 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 in average better results than LK and its variants. GLS+FLS+2Opt also out-performed Simulated Annealing [JAMS89], Tabu Search [Kno94] and hybrid Genetic Algorithm [FM96] methods based on LK. One must be cautious when interpreting such empirical results as they could be affected by many factors, including implementation issues [Hoo95]. But given that the TSP is an extensively studied problem, it takes something special for an algorithm to have a performance comparable if not better to that of the champions. It must be emphasised that LK is specialised for TSP but GLS and FLS are much simpler general-purpose algorithms, which require only a fraction of the programming effort, required to implement the specialised methods. Details of GLS+FLS applied to the TSP can be found in [VT99].

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 optima [Vou98].

The research group at University of Essex are currently attempting to gain more understanding about when and where GLS works and why. Such understanding would enable us to improve GLS or develop specialised strategies for GLS to handle individual problems. On the way towards these goals, we have developed a specialised GLS algorithm called GLSSAT [MT99, MT00]. In solving SAT (satisfiability) problems, GLSSAT was comparable with the state-of-the-art algorithms such as WalkSAT [SKC94] in its performance. In solving MAX-SAT (optimisation) problems, GLSSAT comfortably out-performed DLM [WS97], MaxWalkSAT [JKS95] and GRASP [RF96].

Research in GLS and its predecessor GENET have been followed up by researchers outside our group. For example, GLS was applied to the vehicle routing problem [KPS99, BFK⁺97]. This work has been incorporated in *Dispatcher*; a package for vehicle routing developed by ILOG (<http://www.ilog.fr/html/products/>). [HM99] combined GLS with Memetic Algorithms and applied it to the TSP. [FPZ00] successfully applied GLS and also FLS to the bin-packing problem and conducted comparisons with complete and heuristic search methods. [CPvdV99] recently reported improved results for GLS on the TSP using local search methods based on Dynamic Programming. [BBD⁺95] applied GENET to radio link frequency assignment problem. Other applications of GENET include rail traffic control [JB97] and logic programming [LT95, ST98].

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) [Hol75,

Dav87, Dav91, Gol89]. The idea in GAs 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 [**ERR94, RER95**] and demonstrated promising results in discrete optimisation [**WT94, WT95**].

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 premature convergence (this situation resembles a local optimum situation for local search methods). In particular, 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 susceptible 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 penalisation.

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 [**Lau99**]. GGA has been applied to the Processors Configuration Problem [**LT97, LT98b**], General Assignment Problem [**LT98a**] and the Radio Link Frequency Assignment Problem [**LTar**] with excellent results. Details of GGA and its applications will be reported in another occasion.

8. 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. They would also like to thank the anonymous referees and the editor for their constructive comments and helpful suggestions. This project was partially sponsored by EPSRC funds GR/H75275, GR/L20122 and GR/M46297 and a Research Promotion Fund from University of Essex.

References

- [AAH95] N. Azarmi and W. Abdul-Hameed, *Workforce scheduling with constraint logic programming*, BT Technology Journal **13** (1995), no. 1, 81–94.
- [AK89] E. H. L. Aarts and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester, 1989.
- [Bak93] S. Baker, *Applying simulated annealing to the workforce management problem*, ISR Technical Report, BT Laboratories, Martlesham Heath, Ipswich, 1993.
- [BBD⁺95] A. Bouju, J. F. Boyce, C. H. D. Dimitropoulos, G. vom Scheidt, and J. G. Taylor, *Intelligent search for the radio link frequency assignment problem*, Proceedings of the International Conference on Digital Signal Processing (Cyprus), 1995.
- [Ben92] J. L. Bentley, *Fast algorithms for geometric traveling salesman problems*, ORSA Journal on Computing **4** (1992), 387–411.

- [BFK⁺97] B. D. Backer, V. Furnon, P. Kilby, P. Prosser, and P. Shaw, *Solving vehicle routing problems using constraint programming and metaheuristics*, Technical Report, GreenTrip Project, 1997, <http://www.cs.strath.ac.uk/~ps/GreenTrip/>.
- [Col90] A. Colmerauer, *An introduction to Prolog III*, Communications of the ACM **33** (1990), no. 7, 69–90.
- [CPvdV99] R. Congram, C. N. Potts, and S. L. van de Velde, *Dynasearch Algorithms for the Traveling Salesman Problem*, Travelling Salesman Problem Workshop (University of Southampton), 22 September 1999.
- [Cra93] J.-Y. Cras, *A review of industrial constraint solving tools*, AI Perspective Series, AI Intelligence, Oxford, UK, 1993.
- [Dav87] L. Davis (ed.), *Genetic algorithms and simulated annealing*, Research notes in AI, Pitman/Morgan Kaufmann, 1987.
- [Dav91] L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.
- [DTWZ94] A. Davenport, E. P. K. Tsang, C. J. Wang, and K. Zhu, *Genet: a connectionist architecture for solving constraint satisfaction problems by iterative improvement*, Proceedings of AAAI-94, 1994, pp. 325–330.
- [ERR94] A. E. Eiben, P.-E. Raue, and Zs. Ruttkay, *Solving constraint satisfaction problems using genetic algorithms*, Proceedings of 1st IEEE Conference on Evolutionary Computation, IEEE Press, 1994, pp. 542–547.
- [FDG⁺95] E. C. Freuder, R. Dechter, M. Ginsberg, B. Selman, and E. Tsang, *Systematic versus stochastic constraint satisfaction, Panel Paper*, Proceedings of 14th International Joint Conference on AI (C. Mellish, ed.), 1995, pp. 2027–2032.
- [FM94] E. C. Freuder and A. Mackworth (eds.), *Constraint-based reasoning*, MIT Press, 1994.
- [FM96] B. Freisleben and P. Merz, *A Genetic Local Search Algorithm for Solving the Symmetric and Asymmetric TSP*, Proceedings of IEEE International Conference on Evolutionary Computation (Nagoya, Japan), 1996, pp. 616–621.
- [FPZ00] O. Faroe, D. Pisinger, and M. Zachariasen, *Guided local search for the three-dimensional bin packing problem*, manuscript, 2000.
- [GL97] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
- [Glo89] F. Glover, *Tabu search Part I*, ORSA Journal on Computing **1** (1989), 190–206.
- [Glo90] F. Glover, *Tabu search Part II*, ORSA Journal on Computing **2** (1990), 4–32.
- [Gol89] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, 1989.
- [Hal71] P. A. V. Hall, *Branch-and-Bound and Beyond*, Proceedings of 2nd International Joint Conference on AI, 1971, pp. 641–650.
- [HM99] D. Holstein and P. Moscato, *Memetic Algorithms using Guided Local Search: A case study*, New Ideas in Optimisation (D. Corne, F. Glover, and M. Dorigo, eds.), McGraw-Hill, 1999, pp. 235–243.
- [Hol75] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan press, Ann Arbor, MI, 1975.
- [Hoo95] J. N. Hooker, *Testing heuristics: we have it all wrong*, Journal of Heuristics **1** (1995), no. 1, 33–42.
- [JAMS89] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon, *Optimization by simulated annealing: an experimental evaluation, part I, graph partitioning*, Operations Research **37** (1989), 865–892.
- [JB97] R. M. J. Jose and J. F. Boyce, *Application of connectionist local search to line management rail traffic control*, Proceedings of the Third International Conference on the Practical Application of Constraint Technology (London), 1997.
- [JKS95] Y. Jiang, H. Kautz, and B. Selman, *Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT*, 1st International Joint Workshop on Artificial Intelligence and Operations Research, 1995.
- [Joh90] D. Johnson, *Local Optimization and the Traveling Salesman Problem*, Proceedings of the 17th Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science, vol. 443, Springer-Verlag, 1990, pp. 446–461.
- [Kno94] J. Knox, *Tabu Search Performance on the Symmetric Traveling Salesman Problem*, Computers Operations Research **21** (1994), no. 8, 867–876.
- [Koo57] B. O. Koopman, *The Theory of Search, Part III: The Optimum Distribution of Searching Effort*, Operations Research **5** (1957), 613–626.

- [KPS99] P. Kilby, P. Prosser, and P. Shaw, *Guided Local Search for the Vehicle Routing Problem with time windows*, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization (S. Voss, S. Martello, I. H. Osman, and C. Roucairol, eds.), Kluwer Academic Publishers, 1999, pp. 473–486.
- [Lau99] T. L. Lau, *Guided Genetic Algorithm*, Ph.D. thesis, Department of Computer Science, University of Essex, Colchester, UK, 1999.
- [LK73] S. Lin and B. W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem*, Operations Research **21** (1973), 498–516.
- [LT95] J. H. M. Lee and V. W. L. Tam, *A framework for integrating artificial neural networks and logic programming*, International Journal on Artificial Intelligence Tools **4** (1995), no. 1&2, 3–32.
- [LT97] T. L. Lau and E. P. K. Tsang, *Solving the processor configuration problem with a mutation-based genetic algorithm*, International Journal on Artificial Intelligence Tools (IJAIT) **6** (1997), no. 4, 567–585.
- [LT98a] T. L. Lau and E. P. K. Tsang, *The guided genetic algorithm and its application to the general assignment problem*, IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98) (Taiwan), 1998.
- [LT98b] T. L. Lau and E. P. K. Tsang, *Solving large processor configuration problems with the guided genetic algorithm*, IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98) (Taiwan), 1998.
- [LTar] T. L. Lau and E. P. K. Tsang, *Guided Genetic Algorithm and its application to the Radio Link Frequency Allocation Problem*, Journal of Constraints (to appear).
- [Lue84] D. Luenberger, *Linear and nonlinear programming*, Addison-Wesley, 1984.
- [LW66] E. W. Lawler and D. E. Wood, *Branch-and-bound methods: a survey*, Operations Research **14** (1966), 699–719.
- [LWR95] J. Lever, M. Wallace, and B. Richards, *Constraint logic programming for scheduling and planning*, BT Technology Journal **13** (1995), no. 1, 73–80.
- [MO96] O. Martin and S. W. Otto, *Combining Simulated Annealing with Local Search Heuristics*, Metaheuristics in Combinatorial Optimization (G. Laporte and I. H. Osman, eds.), Annals of Operations Research, vol. 63, 1996.
- [MPR98] R. A. Murphey, P. M. Pardalos, and M. G. C. Resende, *Frequency Assignment Problems*, Technical Report: 98.16.1, AT&T Labs, 1998, (to appear in Du, D-Z., Pardalos, P. (eds.), Handbook of Combinatorial Optimization, Supplement Volume A, Kluwer Academic Publishers, 1999).
- [MS98] K. Marriott and P. J. Stuckey, *Programming with constraints, an introduction*, MIT Press, 1998.
- [MT99] P. Mills and E. P. K. Tsang, *Guided local search applied to the satisfiability (SAT) problem*, Proceedings, 15th National Conference of the Australian Society for Operations Research (ASOR-99) (Queensland, Australia), July 1999, pp. 827–883.
- [MT00] P. Mills and E. P. K. Tsang, *Guided local search for solving SAT and weighted MAX-SAT problems*, Journal of Automatic Reasoning, Special Issue on Satisfiability Problems **24** (2000), 205–223.
- [OvG89] R. H. J. M. Otten and L. P. P. van Ginneken, *The Annealing Algorithm*, Kluwer Academic Publishers, Boston, MA, 1989.
- [Pug95] J-F. Puget, *Applications of Constraint Programming*, Proceedings, Principles and Practice of Constraint Programming (CP'95) (U. Montanari and F. Rossi, eds.), Lecture Notes in Computer Science (LNCS), vol. 976, Springer Verlag, Berlin, Heidelberg & New York, 1995, pp. 647–650.
- [Rei91] G. Reinelt, *A Traveling Salesman Problem Library*, ORSA Journal on Computing **3** (1991), 376–384.
- [RER95] Zs. Ruttikay, A. E. Eiben, and P. E. Raue, *Improving the performances of GAs on a GA-hard CSP*, Proceedings, CP95 Workshop on Studying and Solving Really Hard Problems, 1995, pp. 157–171.
- [RF96] M. G. C. Resende and T. A. Feo, *A GRASP for Satisfiability*, Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge (David S. Johnson and Michael A. Trick, eds.), DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, 1996, pp. 499–520.

- [RND77] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial algorithms: theory and practice*, Prentice Hall, Englewood Cliffs, N.J., 1977.
- [Sim95] H. Simonis, *The chip system and its applications*, Proceedings, Principles and Practice of Constraint Programming (CP'95) (U. Montanari and F. Rossi, eds.), Lecture Notes in Computer Science (LNCS), vol. 976, Springer Verlag, Berlin, Heidelberg & New York, 1995, pp. 643–646.
- [SKC94] B. Selman, H. A. Kautz, and B. Cohen, *Noise strategies for improving local search*, Proceedings, 12th National Conference for Artificial Intelligence, 1994, pp. 337–343.
- [ST98] P. Stuckey and V. Tam, *Semantics for using stochastic constraint solvers in constraint logic programming*, Journal of Functional and Logic Programming (1998).
- [Sto83] L. D. Stone, *The Process of Search Planning: Current Approaches and Continuing Problems*, Operations Research **31** (1983), 207–233.
- [THL95] S. Tiourine, C. Hurkins, and J. K. Lenstra, *An overview of algorithmic approaches to frequency assignment problems*, EUCLID CALMA Project Overview Report, Delft University of Technology, The Netherlands, 1995.
- [Tsa93] E. P. K. Tsang, *Foundations of constraint satisfaction*, Academic Press, 1993.
- [TV97] E. P. K. Tsang and C. Voudouris, *Fast local search and guided local search and their application to British Telecom's workforce scheduling problem*, Operations Research Letters **20** (1997), no. 3, 119–127.
- [TW92] E. P. K. Tsang and C. J. Wang, *A generic neural network approach for constraint satisfaction problems*, Neural network applications (J. G. Taylor, ed.), Springer-Verlag, 1992, pp. 12–22.
- [Vou97] C. Voudouris, *Guided Local Search for Combinatorial Optimisation Problems*, Ph.D. thesis, Department of Computer Science, University of Essex, Colchester, UK, 1997.
- [Vou98] C. Voudouris, *Guided local search - an illustrative example in function optimisation*, BT Technology Journal **16** (1998), no. 3, 46–50.
- [VT96] C. Voudouris and E. P. K. Tsang, *Partial Constraint Satisfaction Problems and Guided Local Search*, Proceedings of the Second International Conference on the Practical Application of Constraint Technology (London, UK), 1996, pp. 337–356.
- [VT98] C. Voudouris and E. Tsang, *Solving the Radio Link Frequency Assignment Problems using Guided Local Search*, Proceedings of NATO symposium on Frequency Assignment, Sharing and Conservation in Systems (AEROSPACE), AGARD (Aalborg, Denmark), 1998, paper no. 14.
- [VT99] C. Voudouris and E. Tsang, *Guided local search and its application to the travelling salesman problem*, European Journal of Operational Research **113** (1999), no. 2, 469–499.
- [Wal96] M. Wallace, *Practical applications of constraint programming*, Journal of Constraints **1** (1996), no. 1&2, 139–168.
- [WS97] B. W. Wah and Y. Shang, *Discrete Lagrangian-Based Search for Solving MAX-SAT Problems*, Proceedings, 15th International Joint Conference on Artificial Intelligence, 1997, pp. 378–383.
- [WT91] C. J. Wang and E. P. K. Tsang, *Solving constraint satisfaction problems using neural networks*, Proceedings of IEE Second International Conference on Artificial Neural Networks, 1991, pp. 295–299.
- [WT94] T. Warwick and E. P. K. Tsang, *Using a genetic algorithm to tackle the processors configuration problem*, Proceedings, ACM Symposium on Applied Computing, 1994, pp. 217–221.
- [WT95] T. Warwick and E. P. K. Tsang, *Tackling car sequencing problems using a generic genetic algorithm*, Evolutionary Computation **3** (1995), no. 3, 267–298.
- [ZF94] M. Zweben and M. S. Fox (eds.), *Intelligent scheduling*, Morgan Kaufmann, San Francisco, 1994.

INTELLIGENT SYSTEMS RESEARCH GROUP, BT LABORATORIES, BRITISH TELECOMMUNICATIONS PLC., UNITED KINGDOM

E-mail address: chriv@info.bt.co.uk

DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF ESSEX, UNITED KINGDOM

E-mail address: edward@essex.ac.uk