Guided Local Search: An Illustrative Example in Function Optimisation

Christos Voudouris

The Guided Local Search method has been successfully applied to a number of hard combinatorial optimisation problems from the well-known TSP and QAP to real world problems such as Frequency Assignment and Workforce Scheduling.
In this paper, we are demonstrating that the potential applications of GLS are not limited to optimisation problems of discrete nature but also to difficult continuous optimisation problems. Continuous optimisation problems arise in many engineering disciplines (such as electrical and mechanical engineering) in the context of analysis, design or simulation tasks. The problem examined gives an illustrative example of the behaviour of GLS, providing insights on the mechanisms of the algorithm.

1. Introduction

Guided Local Search originally proposed by Voudouris and Tsang [29] is a general optimisation technique suitable for a wide range of combinatorial optimisation problems. Successful applications of the technique so far include practical problems such as Frequency Allocation [29], Workforce Scheduling [28] and Vehicle Routing [2, 18] and also classic problems such as the Travelling Salesman Problem (TSP) and the Quadratic Assignment Problem (QAP) [30]. Guided Local Search (GLS) belongs to a class of techniques known as *Metaheuristics* [22, 23, 25]. Prominent members of this class include *Tabu Search* [7-12], *Simulated Annealing* [1, 5, 19, 21], *GRASP* [6], *Genetic Algorithms* [4, 14, 24], Scatter Search [8] and others. Metaheuristics aim at enhancing the performance of heuristic methods in solving large and difficult combinatorial optimisation problems.
In the case of GLS, the main focus is on the exploitation of problem and search-related information to effectively guide local search heuristics in the vast search spaces of NP-hard optimisation problems. This is achieved by augmenting the objective function of the problem to be minimised with a set of penalty terms which are dynamically manipulated during the search process to steer the heuristic to be guided. Higher goals, such as the distribution of the search effort to the areas of the search space according to the promise of these areas to contain high quality solutions, can be expressed and pursued.
GLS is closely related to the *Frequency-Based Memory* approaches introduced in Tabu Search [9, 13], extending these approaches to take into account the quality of structural parts of the solution and also react to feedback from the local optimisation heuristic under guidance. In this paper, we provide an illustrative example of how GLS works by explaining its use for solving an a non-convex optimisation problem.

2. Nonconvex Optimisation and Global Optimisation Methods

Continuous optimisation problems arise in many engineering disciplines (such as electrical and mechanical engineering) in the context of analysis, design or simulation tasks. Particularly difficult problems are those with non-linear multi-extremal cost functions (that is functions with many local minima). These problems, also known as nonconvex optimisation problems [15], are difficult to solve using deterministic gradient-based algorithms used extensively elsewhere in continuous optimisation. Gradient algorithms can be easily trapped in the many local minima of the cost function, so failing to reach the global minimum.
Global Optimisation (GO) methods which seek the global minimum are utilised to solve such problems. The most simple global optimisation algorithm is to run a gradient algorithm many times and from different starting points in the hope that the global

minimum will be amongst the local minima obtained over the many runs. Example of such algorithm is the variation of the Sequential Unconstrained Minimisation Technique suggested in [15]. Many other GO algorithms exist which make use of gradient techniques or derive directly from general search methods such as Genetic Algorithms [16], Simulated Annealing [17], Function Smoothing [27], Orthogonal Arrays with the GRG algorithm [20] to name but a few.

3. Local Search for Continuous Optimisation Problems

Recently and mainly driven by the use of Genetic Algorithms [4, 14, 16] in combinatorial optimisation, GO methods have been developed which deal with nonconvex optimisation as a combinatorial optimisation task. The idea is to convert the continuous problem to a discrete one by encoding the real variables of the cost function as binary strings.

In the case of binary encoding, a binary string value is interpreted to represent an integer in base-2 notation. The mapping of the binary string to a real variable works as follows. The binary string value is first converted to the corresponding integer. This integer is then scaled by the appropriate coefficient to give a real value in the desired range (i.e. domain of variable) [4]. One binary string is used for each problem variable and combinatorial search is utilised to find these binary string configurations which after decoding result in the optimal value for the real-valued cost function. Increasing the number of bits used for representing each variable increases the accuracy of the solution but also results in an increase of the combinatorial search space.

Although binary encoding schemes were principally developed for Genetic Algorithms, they have also been used in the context of local search [3, 31]. To explain how local search operates in this case, let us consider the problem with two variables $x \in A \subset \Re$ and $y \in B \subset \Re$ and a function $f(x, y)$ to be minimised in $A \times B \subset \Re^2$. A local search move flips the value of a bit in the binary string representing the solution (comprises the binary strings of the function's variables). In the x-y plane, bit flips translate to "jumps" in either the *x* or *y* direction. The more significant the bit changed, the larger the step of the "jump" performed. Local search starting from a random binary string examines all possible bit flips and performs that which results in the maximum reduction in cost (minimisation case). The new solution if better replaces the old solution and the procedure continues from there on until a solution is reached for which no further improvement is possible.

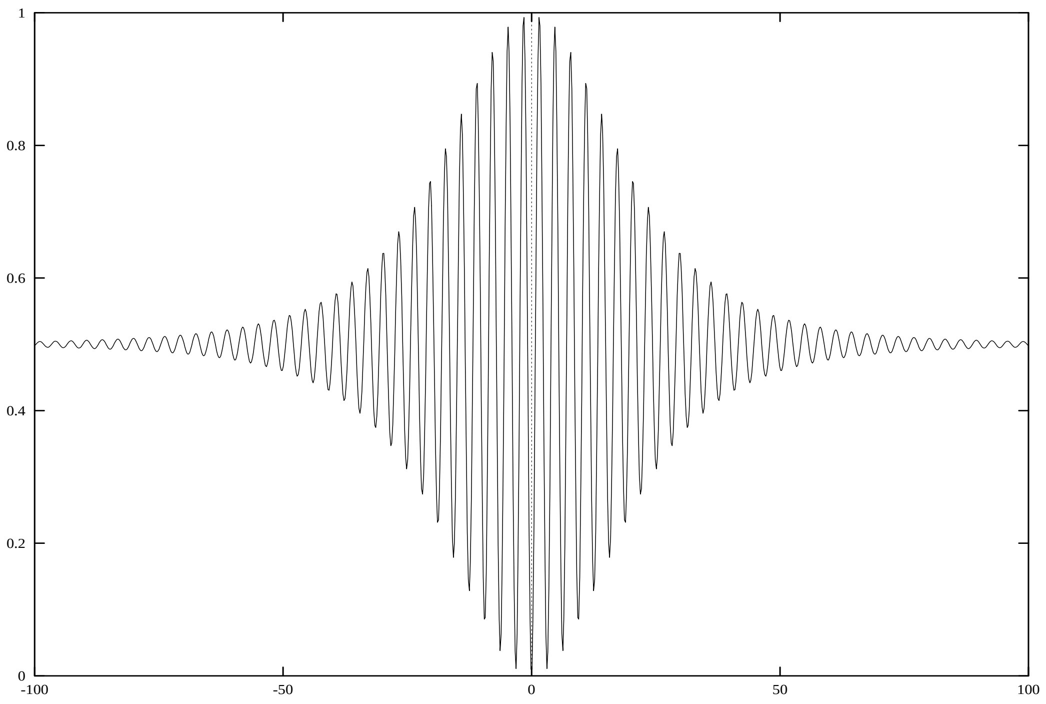

Figure 1. Cross section of F6 function.

4. The Sine Envelope Sine Wave (F6) Function

As mentioned in the section 2, nonconvex optimisation refers to non-linear multi-extremal cost functions. An example of such a function, mentioned many times in the literature, is the sine envelope sine wave function also known as F6 [3, 4, 31]:

$$F6(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{\left[1.0 + 0.001\cdot\left(x^2+y^2\right)\right]^2} \quad (1)$$

minimised in the domain [-100,100]×[-100,100]. F6 has been suggested as a benchmark for Genetic Algorithms [26].
A cross section of the function is shown in Figure 1. The global minimum of F6 is located at (0,0) where the function takes the value 0. The basin of the global minimum is very narrow and therefore difficult to reach unless a lucky start is made from within the domain of attraction of the global minimum. The many local minima of the function are arranged in concentric cycles around the global minimum forming an ideal trap for hill-climbing based techniques. In F6, local gradients provide limited (if any) information on the location of the global minimum. Guided Local Search may be exploited to help local search to escape from local minima and moreover distribute its search efforts in the search space.

5. Guided Local Search for Global Optimisation

GLS is iteratively posting constraints which modify the landscape and guide local search out of local minima and towards promising areas in the search space. Constraint posting in this problem could be based on information gathered during the search process. For example, if local search reaches a local minimum then an assumption can be made that the global minimum is unlikely to reside in the surrounding area. Constraints could then be introduced that exclude this area from being searched in future iterations. These constraints are essentially soft because we cannot be sufficiently confident that local search thoroughly searches the space around a solution when this solution is visited.
A set of features is defined that allow us to constrain solutions. A feature can be any solution property represented by an indicator function [30]. A simple setting for global optimisation is to divide the domains of variables into a number of non-overlapping and equally-sized intervals. Let us consider the variable $x \in (a,b]$. A set of features f_i, i=1, ...,n, can be defined by the intervals $(a_0=a,a_1]$, $(a_1, a_2]$, ..., $(a_{n-1}, a_n=b]$ as follows:

$$I_i(x) = \begin{cases} 1, & x \in \left(a_{i-1}, a_i\right] \\ 0, & otherwise \end{cases} \quad (2).$$

Each feature f_i is attached a penalty parameter p_i to allow GLS to penalise solutions that are characterised by the feature such that they can be avoided. The cost function is augmented with penalty terms to form the augmented cost function. This function replaces the original function and it is minimised instead. The augmented version of F6 is defined as follows:

$$H(x,y) = F6(x,y) + \lambda\cdot\left(\sum_{i=1}^{n} I_{xi}(x)\cdot p_{xi} + \sum_{j=1}^{m} I_{yj}(y)\cdot p_{yj}\right) (3),$$

where n the number of features defined over the domain of x, m is the number of features defined over the domain of y, and λ is the parameter that controls the relative importance of constraints with respect to the primary cost term (i.e. function to be minimised). Initially, all penalty parameters of features are set to 0 ($p_{xi} = 0$, $p_{yj} = 0$, $i = 1, ..., n$, $j = 1, ..., m$). Each time local search settles in a local minimum, we simply increment by one the penalty parameters of the features exhibited by the local minimum (only two at a time). This increases by 2*λ the cost of all solutions that lie in the intersection of the zones corresponding to the penalised features and by λ the cost of all solutions that lie in either one of these zones (see Figure 2). As a result, local search will primarily avoid the rectangular area with centre the local minimum and also to a lesser degree the two zones that run parallel to the co-ordinate axis as shown in Figure 2. This simple technique can be used to minimise arbitrary functions. In fact, there is nothing that binds the method to F6 which may not be used for other functions with two or more variables. In the following, we examine the results obtained for F6.

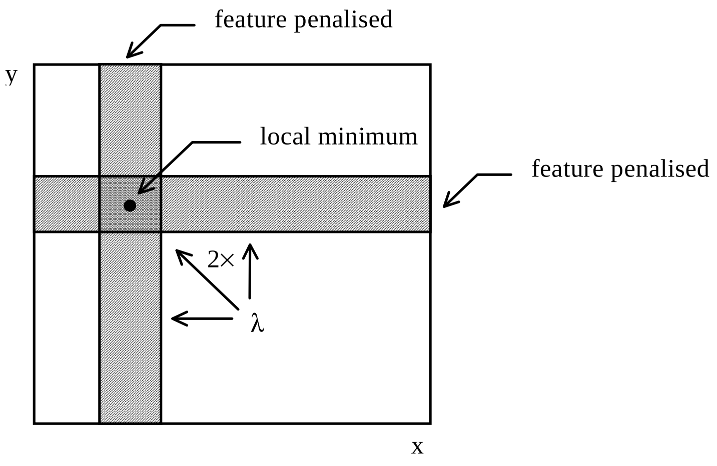

Figure 2. Changes in cost due to penalising the features exhibited by a local minimum.

6. Experimentation with the F6 Function

Following Davis [4], we used 22-bits for representing each variable. An equal number of features was used to cover the domain of each variable (*n*=*m*). The algorithm was relatively insensitive to the parameter λ and performed well for values of λ greater than 0.2. The value 0.25 for λ was used in the tests. Experiments were performed for varying *n* (i.e. number of features per variable) to determine how this parameter affects GLS. The values tried for *n* were 5, 10, 15, 20, 50, and 100. Fifty runs from random solutions (random binary strings) were performed for each value of *n* considered with the iteration limit set to 10,000 local search improvement cycles. Table 1 illustrates the results obtained. The best setting proved to be *n*=*m*=5. Under this setting, the algorithm succeeded in finding the exact optimal solution (0,0) in 100% of 50 runs. Under all settings, the algorithms found the exact optimum many times.

This performance further improves if more time is given to the algorithm. For example, in the case (*n=m=100*) where most failures occurred (28 out of 50 runs), we performed the same experiment but this time allowed the algorithm to complete 100,000 local search iterations. The performance of GLS significantly improved and the algorithm found the exact optimum in 50 out of 50 runs (no failures).

The main observation made was that GLS performance degraded as the number of features used increased. More features meant more effort to leave a particular area but also more careful exploration. For this particular function, diversification of search to sample the whole search space proved important to find the global minimum quickly. The distribution of points visited for *n*=*m*=10 during 10,000 iterations of local search is shown in Figure 3. During the particular run that generated Figure 3, the optimal solution was found early and after 1965 iterations. Despite that, the algorithm was allowed to continue until 10,000 iterations were completed to get a better picture of the solutions visited by the algorithm. As one can see in Figure 3, the algorithm distributed its efforts over the whole of the search space but visited mainly local minima. That is why points in are arranged in concentric cycles around the point (0,0).

No. of features	n=m=5	n=m=10	n=m=15	n=m=20	n=m=50	n=m=100
Mean Cost	0.00E+00	4.55E-11	3.19E-10	2.73E-10	1.97E-04	3.21E-04
Best Solution	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Worst Solution	0.00E+00	2.28E-09	2.28E-09	2.28E-09	9.72E-03	9.72E-03
Mean Iterations	2287.32	2566.22	2954.08	3526.9	4132.66	3738.48
Mean Time	2.823333	3.150668	3.634334	4.382333	5.188333	4.654
Mean Funct. Eval.	104958.6	117778.8	135588	161878.4	189675.6	171578.5
Optimal Runs	50	49	43	44	31	22
Total runs	50	50	50	50	50	50

Table 1. GLS performance on F6 (Time in CPU seconds on a DEC Alpha 3000/600 175MHz).

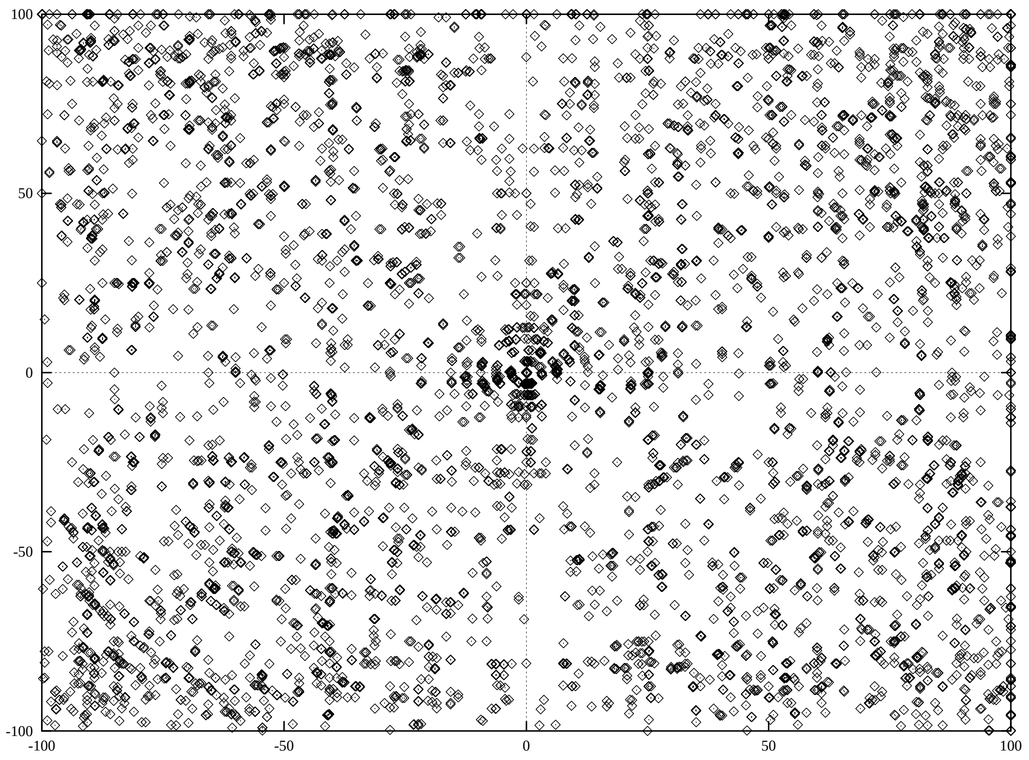

Figure 3. All the points visited during the first 10,000 iterations of local search.

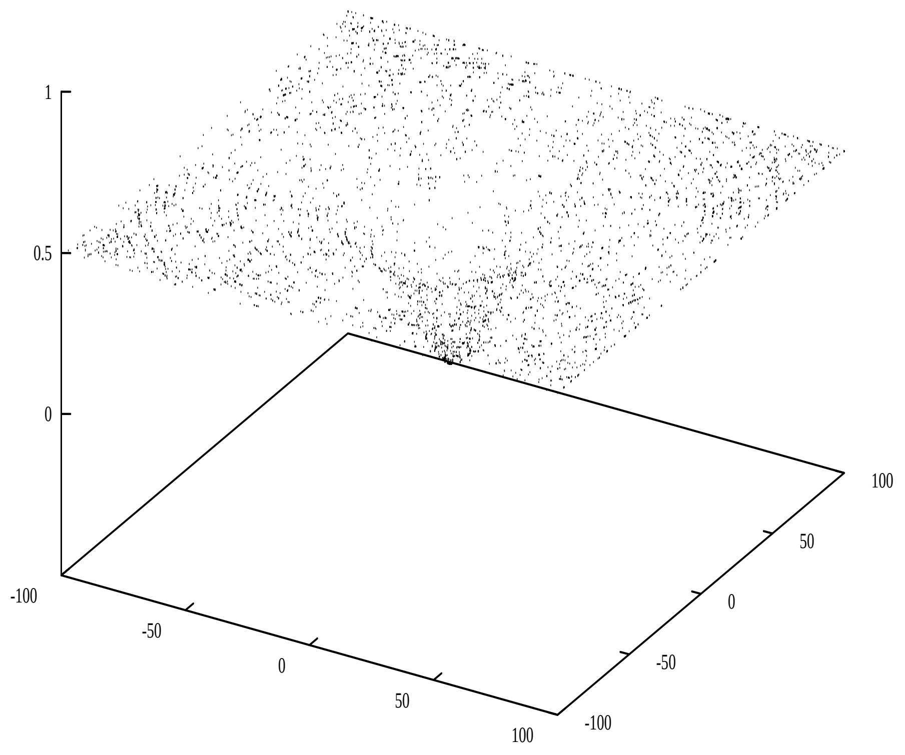

Figure 4. 3-D View of Figure 3.

This is more clearly demonstrated in Figure where a 3-D view of the visited points is shown. The shape formed is exactly the bottom part of F6 which suggests that the points are actually local minima in the great majority. Note here, that GLS is exploring binary space and not numeric space. In general, local minima and their attraction basins in the binary space are different from the local minima and their attraction basins appearing in the numeric space. Because of the symmetrical landscape, the binary encoding used and the structure of the GLS features, the majority of the solutions visited by GLS in the case of F6 have the property of being numeric local minima as illustrated in Figures 3 and 4. This is not necessarily the case for functions with non-symmetrical landscapes. In these cases, grey encodings (see [3] for example) and/or features of different structure may yield better performance than the encoding scheme and features used in this paper.

7. Conclusions

In this paper, we have shown that GLS has the potential to be utilised in the optimisation of real-valued functions with numerous local minima, which are considered to be difficult for gradient-based methods. The application of GLS to optimise the F6 function, a benchmark for Genetic Algorithms, has been examined. GLS repeatedly located the exact global optimum of the function. The paper also serves in demonstrating how artificial solution features can be created when no features can be deduced from the structure of the objective function, which adds support to our claim that GLS has wide applications.

References

[1] Aarts, E.H.L., and Korst, J.H.M., *Simulated Annealing and Boltzmann machines*, Wiley, Chichester, 1989.

[2] Backer, B.D., Furnon, V., Prosser, P., Kilby, P., and Shaw, P., "Solving vehicle routing problems using constraint programming and metaheuristics", Submitted to the *Journal of Heuristics* special issue on Constraint Programming, July 1997.

[3] Battiti, R., and Tecchioli, G., "The Reactive Tabu Search", *ORSA Journal on Computing*, Vol. 6, 126-140, 1994.

[4] Davis., L., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[5] Dowsland, A., "Simulated Annealing", in: Reeves, C. R. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 20-69, 1993.

[6] Feo, T.A., and Resende, M.G.C., "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization*, vol. 6, pp. 109-133, 1995.

[7] Glover, F., "Future paths for integer programming and links to artificial intelligence", *Computers Ops Res.*, 5, 533-549, 1986.

[8] Glover, F., "Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges", in: *Interfaces in Computer Science and Operations Research*, Barr, Helgason and Kennington eds., Kluwer Academic Publishers, 1996.

[9] Glover, F., "Tabu Search Fundamentals and Uses", Graduate School of Business, University of Colorado, Boulder, 1995.

[10] Glover, F., "Tabu search Part I", *ORSA Journal on Computing*, Vol. 1, 190-206, 1989.

[11] Glover, F., "Tabu search Part II", *ORSA Journal on Computing*, Vol. 2, 4-32, 1990.

[12] Glover, F., "Tabu Search: improved solution alternatives for real world problems", in: Birge & Murty (ed.), *Mathematical Programming: State of the Art*, 64-92, 1994.

[13] Glover, F., and Laguna, M., "Tabu Search", in: Reeves, C. R. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 71-141, 1993.

[14] Goldberg, D. E., *Genetic algorithms in search, optimisation, and machine learning*. Addison-Wesley, 1989.

[15] Hiller, F.S., and Lieberman, G., *Introduction to Operations Research*. Sixth edition, McGraw-Hill, New York, 1995.

[16] Holland, J.H., *Adaptation in natural and artificial systems*. University of Michigan press, Ann Arbor, MI, 1975.

[17] Ingber, A.L., "Very fast simulated re-annealing", Journal of Mathematical Computer Modelling, Vol. 12 No. 8, 967-973, 1989.

[18] Kilby, P., Prosser, P., and Shaw, P., "Guided local search for the vehicle routing problem", *Proceedings of the 2nd International Conference on Metaheuristics*, July 1997.

[19] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimisation by Simulated Annealing", *Science*, Vol. 220, 671-680, 1983.

[20] Kota, S., and Chiou, S., "Use of Orthogonal Arrays in Mechanism Synthesis", *Mechanical Machine Theory*, Vol. 28, 777-794, 1993.

[21] Laarhoven, P.J.M.V., and Aarts, E.H.L., *Simulated Annealing: Theory and Applications*, Kluwer, Dordrecht, 1988.

[22] Osman, I.H., ”An Introduction to Meta-Heuristics”, M. Lawrence and C. Wilson (eds.), in: *Operational Research Tutorial Papers*, Operational Research Society Press, Birmingham, UK, 92-122, 1995.
[23] Reeves, C.R. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 1993.
[24] Reeves, C.R., “Genetic Algorithms”, in: Reeves, C. (ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publishing, 151-196, 1993.
[25] Reeves, C.R., “Modern Heuristic Techniques”, in: V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith (ed.), John Wiley & Sons, *Modern Heuristic Search Methods*, 1-25, 1996.
[26] Schaffer, J.D., Caruana, R.A., Eshelman, L.J., and Das, R., “A study of control parameters affecting online performance of genetic algorithms for function optimisation”, *Proceedings of 3rd Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 51-60, 1989.
[27] Styblinski, M.A. and Tang, T.S., “Experiments in Nonconvex Optimisation: Stochastic Approximation with Function Smoothing and Simulated Annealing”, *Neural Networks*, Vol. 3, 467-483, 1990.
[28] Tsang, E., and Voudouris, C., “Fast local search and guided local search and their application to British Telecom's workforce scheduling problem”, *Operations Research Letters*, Vol. 20, No. 3, 119-127, 1997.
[29] Voudouris, C., and Tsang, E., “Partial Constraint Satisfaction Problems and Guided Local Search”, *Proceedings of 2nd Int. Conf. on Practical Application of Constraint Technology* (PACT'96), London, April, 337-356, 1996.
[30] Voudouris, C., *Guided Local Search for Combinatorial Optimization Problems*, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, July, 1997.
[31] Woodruff, D.L., and Zemel, E., “Hashing vectors for tabu search”, *Annals of Operations Research*, 41, 123-137, 1993.