Guided Local Search for the Vehicle Routing Problem

Philip Kilby, Patrick Prosser and Paul Shaw

Department of Computer Science, University of Strathclyde
26 Richmond Street, Glasgow, Scotland. G1 1XH
Email: {pjk,pat,ps}@cs.strath.ac.uk

1 Introduction

The Vehicle Routing Problem (VRP) is a practical problem that has been studied widely in the OR literature [2]. The problem is usually expressed as follows: given a set customers requiring a visit, and a fleet of vehicles based at a depot that can perform the visits, construct a set of routes for the vehicles which minimises the costs of operation. The objective function is usually expressed as costs related to the number of vehicles used and to distance travelled.

Vehicle routing problems generally come with at least capacity constraints on the vehicles (indicating that they can only carry a certain quantity of goods). However, other constraints are common. For example, a limit on the length of any vehicle route (due to fuel limitations), limited route time (due to the length of the working day), and requested time windows at customers.

In the recent past, great success has been achieved via the application of local search techniques and meta-heuristics to routing problems. Most popular has been the tabu search meta-heuristic [1, 6], genetic methods [5, 12, 13], simulated annealing [4], and hybrids, for instance [8, 11].

A new meta-heuristic, guided local search (GLS) [15, 17] is a memory-based method, and so shares similarities with tabu search. However, it operates by augmenting the cost function with a penalty term based on how near the search moves to previously visited local minima, thus encouraging diversification. This paper applies GLS to vehicle routing problems with time windows and capacity constraints. Results indicate that GLS can provide excellent results.

This paper is organised as follows. In section 2, we introduce a local search algorithm for the vehicle routing problem. We begin first by describing the move operators and model, and then go on to describe the objective function and search itself. The GLS meta-heuristic is then presented in section 3. The application of the meta-heuristic to a vehicle routing framework is discussed. Experiments are then performed using GLS on some standard benchmark problems from the literature that involve both capacity constraints on vehicles, and time windows at customers. Conclusions are drawn on the quality of the results in comparison to other methods.

2 Local Search

2.1 Improvement Heuristics

We describe the improvement heuristics we use for the vehicle routing problem. We are ignoring construction heuristics (those used to build a solution), since for the purposes of this paper, we consider the construction of a solution as part of the improvement. That is, in the initial solution no visits are allocated to any vehicle. A

penalty is associated with not performing a visit, and so the search process constructs a solution in the process of minimising cost. Section 2.4 explains this in detail.

Heuristics for routing plan improvement can be within a route, or between routes. Here, we will focus upon one heuristic that operates within routes (the two-opt procedure of [3]) and three that operate between routes: the relocate, exchange, and cross operators, described in [9].

The two-opt procedure is used for improving single vehicle routes, and operates by reversing part of the route. In figure 2.1(a), the two-opt operator is shown. Here, the part of the route from A to B has been reversed, and in this instance, has resulted in a reduction in total length.

The relocate operator moves a customer visit from one route to another. Figure 2.1(b) shows that by moving visit A to another route between B and C, overall distance can be reduced. A visit can also be relocated to an "empty" route, resulting in the creation of a new vehicle itinerary with a single visit.

The exchange operator swaps customer visits in different vehicle routes. In figure 2.1(c), the positions of visits A and B have been interchanged.

The cross operator swaps the end portions of two vehicle routes. Figure 2.1(d) shows the operation of the cross. The sections of the routes after visits A and B have been interchanged, resulting in a distance reduction. It is also possible to cross from the end of one route to the beginning of another, resulting in a concatenation of routes and an empty route. Moreover, if one of the routes involved in a cross operator is empty, the cross can split the non-empty route into two.

2.2 Model

We represent a solution to the vehicle routing problem as set of lists of customer visits, each list corresponding to a single vehicle route. A maximum number of lists are available, corresponding to the number of available vehicles. Lists left empty at the end of search correspond to vehicles that are not required to execute the routing plan. Associated with each vehicle is a capacity, which must always be greater than or equal to the sums of the loads of its deliveries. In the model, different vehicles can have different capacities.

Additionally, a *virtual* vehicle exists that notionally performs the visits not carried out by the real vehicles. The virtual vehicle is different from the real ones in three respects. First, the virtual vehicle can "make" an unlimited number of customer visits. Second, problem constraints (such as time and capacity) do not apply to the virtual vehicle, making all routes in the virtual vehicle legal. Third, the cost incurred by the virtual vehicle when it performs a customer visit is higher than that incurred by a real vehicle. This last point ensures that plans are created with work completed. The virtual vehicle is not used in the final routing plan, but indicates by its contents visits that could not be scheduled.

2.3 Objective

The objective function that we attempt to minimise is the total distance travelled by all vehicles. The cost of a customer visit being performed by the *virtual* vehicle is equal to $\alpha_1(d_{0i} + d_{i0}) + \alpha_2$ where d_{ij} is the distance between customers i and j. We define customer 0 to be the depot. When $\alpha_1 = 1$ and $\alpha_2 = 0$, the cost of the virtual

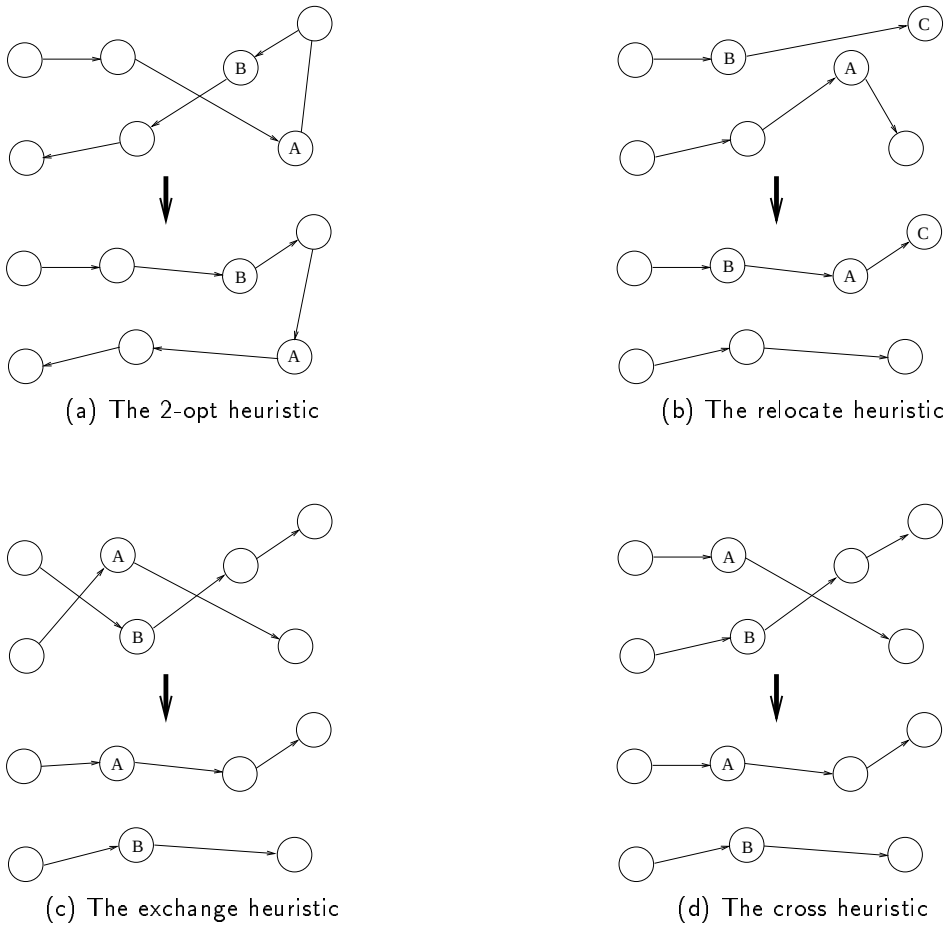

Figure 2.1: Some Improvement Heuristics for the VRP

vehicle making the visit is equal to that of a single vehicle being dispatched from the depot to make only that single visit. Higher values of α_1 and α_2 make it cheaper to perform visits by real vehicles.

Results in [7] suggest that good parameter settings are $\alpha_1 = 1.025$ and $\alpha_2 = 0.0005\Delta$, where Δ is the distance between the two most remote sites in the problem.

2.4 Search Method

The search method we adopt is an entirely greedy one. There are no random numbers involved, and the results are completely reproducible. We first of all begin the search by assuming all customer visits will be carried out by the virtual vehicle, and then use the four improvement heuristics presented to attempt to improve the cost of the solution. At each point in search, the move that decreases the objective by the greatest amount is chosen, implementing the so-called "best acceptance" scheme. New states generated by the heuristics that violate any problem constraints (time windows or

capacity constraints) are rejected. We stop the search when no moves will strictly reduce the objective. We make the additional restriction that the only heuristic that can be used in association with the virtual vehicle is the relocation of a visit from the virtual vehicle to a real vehicle.

2.5 Limitations

The local search method described above can quickly produce reasonable solutions to routing problems (typically within 10% of the best known solution [7]), but the variability of the results from problem to problem is high. We therefore now turn to meta-heuristic methods to stabilise the search process and increase the probability of attaining high quality solutions.

3 Guided Local Search

Guided local search (GLS) [15, 17] is a meta-heuristic based on penalties. The method works by adding a penalty factor to the objective function, based on the experience the search has gained. Roughly speaking, search is penalised if it strays too close to previously visited local minima. GLS has proved to be an effective meta-heuristic for a range of combinatorial optimisation problems such as the travelling salesman problem [17], quadratic assignment [17], function optimisation [16], partial constraint satisfaction [18], and resource allocation [14].

In what follows, we describe the general algorithm, and then its application to the vehicle routing problem.

3.1 The Guided Local Search Algorithm

Guided local search moves out of a local minimum by penalising particular solution features that it considers should not occur in a near-optimal solution. It defines a modified objective function, augmented with a set of penalty terms on these features. The usual local search method is then invoked to improve the augmented objective function. The cycle of local search and penalty term update can be repeated as often as required.

GLS requires the following components:

- A set of features F. Let the indicator function for feature $i \in F$ be f_i. $f_i(\mathcal{S}) = 1$ if the feature i is in solution $\mathcal{S}$, and 0 otherwise.
- A cost vector c. c_i is the "cost" of feature i.
- A penalty factor λ.

GLS tracks penalties applied via a penalty vector p. p_i is the (integer) number of times feature i has been penalised so far. Assuming $O(\mathcal{S})$ is the the original objective function for the problem, GLS defines an augmented objective function:

$$O'(\mathcal{S}) = O(\mathcal{S}) + \lambda \sum_{i \in F} f_i(\mathcal{S}) p_i c_i$$

and requires a local search procedure that minimises it. The user must therefore provide procedure LocalSearch($\mathcal{S}$,p) that performs a local search, starting at solution

$\mathcal{S}$, and returns a new solution. *The improvement is carried out with respect to the augmented objective O'.*

GLS provides a function called ChoosePenaltyFeatures($\mathcal{S}$, p) which takes a solution and the current spread of penalties, and returns the set of features to be penalised. GLS penalises the most costly features in the current solution, weighted by the number of times the feature has already been penalised. This has the effect of concentrating penalties on "bad" features, while tending to accept bad features if they keep appearing in local minima (*i.e.* if they seem inevitable). GLS chooses the features $i \in \mathcal{S}$ for which $c_i/(p_i + 1)$ is largest amongst the features in $\mathcal{S}$. Unless values in c are equivalent or multiples of each other, usually only one feature will be chosen.

Assuming the usual functions of InitialSolution() and StoppingCondition() exist, the GLS algorithm is described by Figure 3.1.

```
p := 0̄                                  // All elements zeroed
S := InitialSolution()                  // S is current solution
S := LocalSearch (S, p)                 // Make sure we start at a local min
S* := S                                 // S* is best solution
while not StoppingCondition() do
    f := ChoosePenaltyFeatures(S, p)
    forall g in f do
        p_g := p_g + 1
    S := LocalSearch (S, p)
    if O(S) < O(S*) then
        S* := S
S* := LocalSearch (S*, 0̄)               // Final local search with zero penalties,
                                        // to get to true local minimum
return S*
```

Figure 3.1: GLS algorithm

3.2 GLS for Vehicle Routing

We define our implementation of GLS for the VRP by instantiating each of the items described in section 3.

- Feature set F: In [15, 17] for the Travelling Salesman Problem (TSP), arcs were chosen as the feature to penalise. We use the same method here, except that now we consider time windows, the arcs are directed.
- Feature costing: We assume that the cost c_a of directed arc feature a is the length of the arc d_a.
- Penalty factor λ: We have found that values of 0.1 to 0.3 work well, and have used 0.2 for the results in section 4.
- LocalSearch($\mathcal{S}$,p): We use the "best acceptance" algorithm defined in section 2.4. To implement the objective O', the local search is carried out using a modified

distance matrix. In the modified matrix, each arc a has a penalised distance $d_a + \lambda p_a d_a$. The matrix is updated each time an element of p is changed.

InitialSolution() and StoppingCondition() are defined in section 4.

4 Experiments

Computational experiments were performed using GLS. The stopping condition used was 2000 iterations of the main loop. The initial solution used was that of all visits being performed by the virtual vehicle. The objective function was the total distance travelled by all vehicles. (There was no incentive to reduce the number of vehicles.) The penalty factor λ was set to 0.2.

The problems studied are as used by Solomon [10]. These data are in 6 classes: C1, C2, R1, R2, RC1, RC2. All problems have 100 customers, a central depot, capacity constraints, time windows on the time of delivery, and a total route time constraint. The C1 and C2 classes have customers located in clusters. In the R1 and R2 classes the customers are at random positions. The RC1 and RC2 classes contain a mix of both random and clustered customers. Each class contains between 8 and 12 individual problem instances, numbered by appending two digits to the end of the class name (*e.g* RC104, R112, C207 *etc.*). There are a total of 56 instances in all. All problems in any one class have the same customer locations, and same vehicle capacities; only the time windows differ. The C1, R1 and RC1 problems have a short scheduling horizon, and require 9 to 19 vehicles. Classes C2, R2 and RC2 are more representative of "long-haul" delivery with longer scheduling horizons and fewer (2 – 4) vehicles. Both travel time and distance are given by the Euclidean distance between points.

Since our objective function has no incentive to minimise vehicles, experiments for problem p were carried out with v_p, v_p+1, and v_p+2 vehicles, where v_p is the minimum number of vehicles reported in the best solution for problem p. For each problem, the solution where all customer visits could be scheduled in the lowest number of vehicles was kept.

In summary, given all 56 problems, 168 experiments (3 for each problem with differing numbers of vehicles) were carried out for 2000 iterations, with $\lambda = 0.2$. Since both our local search method and GLS contain no random element, only a single run on each problem was required.

For each problem, the solution where all customer visits could be scheduled in the lowest number of vehicles was kept.

4.1 Computational Results

The number of vehicles and total distance travelled for the problems within each class were averaged and are shown in table 4.1. The data are compared against results reported in Rochat and Taillard [8], and Taillard *et al.* [11], which are the best two heuristic techniques reported for the VRP. In both the above references, solution quality is reported at three checkpoints during search. We have compared against checkpoint 2 (the middle checkpoint) in both in an attempt to compare results produced using similar CPU resource. For Rochat and Taillard, the CPU times are very close, but for Taillard *et al.*, the time used is more than double that of GLS or Rochat and Taillard. Although checkpoint 1 in Taillard *et al.* is a closer match

to use, we believe that the use of a slow machine by Taillard *et al.* contributes to the reported CPU times. We therefore use checkpoint 2. The mean CPU times per problem (rounded to the nearest 100 seconds) are: Rochat and Taillard–2800s, Taillard *et al.*–7400s, GLS–2900s.

	Rochat and Taillard [8]		Taillard *et al.* [11]		GLS	
Problem Class	Vehicles Used	Distance Travelled	Vehicles Used	Distance Travelled	Vehicles Used	Distance Travelled
C1	10.00	828.45	10.00	828.59	10.00	830.75
R1	12.58	1202.31	12.39	1230.48	12.67	1200.33
RC1	12.50	1368.03	12.00	1387.01	12.12	1388.15
C2	3.00	590.32	3.00	591.14	3.00	592.24
R2	3.09	969.29	3.00	1029.65	3.00	966.56
RC2	3.62	1155.47	3.38	1220.28	3.38	1133.42

Table 4.1: Average results by problem class

Examination of the averaged results indicates that for problem classes C1 and C2, there is little to choose between the three methods. In fact, these problem classes are easy, in that even a local search without a meta-heuristic converges close to the best known solutions. Typically, it is easy to get within 0.5% of the best known solutions quickly.

On class R1, GLS performs poorly, and is typically bad at reducing the numbers of vehicles used, using on average 0.7% more vehicles than Rochat and Taillard and 2.3% more vehicles than Taillard *et al.*. Distances for GLS are similar to Rochat and Taillard, but are less than for Taillard *et al.* This is due to the fact that Taillard *et al.* use less vehicles, which (up to a point) typically means that more distance needs to be traversed. The distance impact is less apparent on a class such as R1 where many vehicles are needed. However, for the R2 and RC2 classes, reducing vehicles by one for a particular problem can have a large impact on distance travelled, since these problems only require 4 vehicles or less.

On class RC1, GLS sits between the other two methods in terms of performance. GLS uses 1.0% more vehicles than Taillard *et al.*, but 3.1% less than Rochat and Taillard. Distance for GLS is roughly equivalent to Taillard *et al.*, but is greater than for Rochat and Taillard, due to the latter using more vehicles.

On classes R2 and RC2, GLS easily outperforms the other two methods. On both classes, the number of vehicles used is the same as for Taillard *et al.*. Comparing to Rochat and Taillard, GLS uses 3% and 7.4% less vehicles for classes R2 and RC2 respectively. With regard to travel distance, GLS reduces distance over Taillard *et al.* by 6.53% for R2 and 7.66% for RC2, while using the same number of vehicles. GLS reduces travel by 0.3% for R2 and 1.9% for RC2 over Rochat and Taillard, even though the latter uses more vehicles.

To summarise, GLS produces very similar results on classes C1 and C2 as the other methods. For class R1, GLS performs poorly, using more vehicles than the other two methods. GLS delivers intermediate performance on class RC1, but is much closer to the (better) results of Taillard *et al.* than those of Rochat and Taillard. For classes R2

and RC2, the results produced by GLS are significantly (around 7%) better.

It should be mentioned that both Rochat and Taillard and Taillard *et al.* use a tabu search approach coupled with a "route memory". The route memory holds good routes previously found by tabu search. The tabu search is applied to a solution constructed from routes drawn from the route memory. After tabu search is applied, the route memory is updated with the new routes. We conjecture that the route memory is responsible for the good performance of these methods on classes R1 and RC1. On classes R2 and RC2, each route is longer, and so the route memory is not as effective as there are potentially many more good routes and route decompositions than for R1 and RC1. Therefore, the route memory does not so quickly move to a state that contains the best, or near best routes. We expect GLS to benefit from a route memory in the same way as the tabu search method, delivering better performance on classes R1 and RC1.

Finally, we report on some new best solutions for Solomon's problems that we have found using GLS during our computational experiments. Some best solutions reported in the literature are based on integer valued distances. Due to ambiguity as to whether time windows are truly met using integer valued distances, solutions created using integer distances are open to dispute. Therefore, we quote as new best solutions those which are better than any produced by using *real* (not integer) distance values.

The cost values for the new best solutions are summarised in table 4.2.

Problem	Previous Best			New Best	
Name	Vehicles	Distance	Ref	Vehicles	Distance
r109	11	1214.54	[11]	11	1205.96
r201	4	1254.80	[11]	4	1254.09
r204	2	869.29	[8]	2	867.33
r205	3	1038.72	[11]	3	998.72
r209	3	944.64	[8]	3	923.96
r210	3	967.50	[8]	3	963.37
rc201	4	1413.79	[11]	4	1406.94
rc202	4	1164.25	[11]	4	1162.80
rc203	3	1079.57	[8]	3	1068.07
rc204	3	806.75	[8]	3	803.90
rc205	4	1328.21	[11]	4	1302.42
rc206	3	1158.81	[11]	3	1156.26
rc207	3	1082.32	[11]	3	1075.25

Table 4.2 : New Best Solutions to Solomon's Problems Found by GLS

5 Conclusion

We have described a local search method for vehicle routing problems based on multiple improvement heuristics, and the possibility of not scheduling some visits via the use of a *virtual* vehicle. Solutions are constructed via improvement from an initial solution where all visits are performed (at high cost) by the virtual vehicle.

A meta-heuristic, guided local search, was then used with the local search method, and was tested on Solomon's capacitated vehicle routing problems with time windows. We compared our results to the best heuristic approaches reported for the VRP with time windows. These approaches use tabu search combined with a route memory.

For problem classes with long routes, GLS easily outperforms these approaches by 7%. For classes with shorter routes, GLS was not as competitive, and was outperformed by up to 2.3%. We expect, however, that the performance of GLS on these instances would be greatly enhanced by the addition of a route memory.

Finally, GLS has provided 13 new best solutions to Solomon's problems. In particular, in class RC2, 88% of best solutions were improved and in class R2, 45% were improved.

As future work, we aim to adjust the cost function to take account of the number of vehicles, and to provide some guidance in the cost function or allowed moves to encourage at least one short route. (Having a short route increases the chances than this route will eventually be removed.) This will allow us to automatically reduce the number of vehicles as search progresses.

Other future work could include the investigation of different feature costing other than arc distance, looking at alternative methods of choosing features to penalise, and the actual penalty method itself.

Acknowledgement

The production of this paper was supported by the GreenTrip project, a research and development undertaking partially funded by the ESPRIT Programme of the Commission of the European Union as project number 20603. The partners in this project are Pirelli (I), ILOG (F), SINTEF (N), Tollpost-Globe (N), and University of Strathclyde (UK).

Bibliography

[1] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Sci.*, 40(10):1276–1290, 1994.

[2] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Ann. Oper. Res.*, 61:227–262, 1995.

[3] S. Lin. Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.*, 44:2245–2269, 1965.

[4] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Oper. Res.*, 41:421–451, 1993.

[5] J.-Y. Potvin and S. Bengio. A genetic approach to the vehicle routing problem with time windows. Technical Report CRT-953, Centre de Recherche sur les Transports, University of Montreal, 1994.

[6] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau. A tabu search heuristic for the vehicle routing problem with time windows. Technical Report CRT-855, Centre de Recherche sur les Transports, University of Montreal, 1993.

[7] P. Prosser and P. Shaw. Study of greedy search with multiple improvement heuristics for vehicle routing problems. Technical Report RR/96/201, Department of Computer Science, University of Strathclyde, Glasgow, January 1997.

[8] Y. Rochat and E. D. Taillard. Probabilitic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

[9] M. W. P. Savelsbergh. Computer aided routing. Centrum voor Wiskunde en Informatica, Amsterdam, 1988.

[10] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Oper. Res.*, 35:254–265, 1987.

[11] E. Taillard, P. Badeau, M. Gendreau, F. Guertain, and J.-Y. Potvin. A new neighbourhood structure for the vehicle routing problem with time windows. Technical Report CRT-95-66, Centre de Recherche sur les Transports, University of Montreal, 1995.

[12] S. R. Thangiah, I. H. Osman, and T. Sun. Hybrid genetic algorithm, simulated annealing, and tabu search methods for vehicle routing problems with time windows. Working paper UKC/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury, 1994.

[13] S. R. Thangiah, R. Vinayagamoorthy, and T. Sun. Vehicle routing with time deadlines using genetic and local algorithms. In *5th International Conference on Genetic Algorithms*, 1993.

[14] E. Tsang and C. Voudouris. Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. Technical Report CSM-246, Department of Computer Science, University of Essex, August 1995.

[15] C. Voudouris. *Guided Local Search for Combinatorial Problems.* PhD thesis, University of Essex, April 1997.

[16] C. Voudouris and E. Tsang. Function optimization using guided local search. Technical Report CSM-249, Department of Computer Science, University of Essex, September 1995.

[17] C. Voudouris and E. Tsang. Guided local search. Technical Report CSM-247, Department of Computer Science, University of Essex, August 1995.

[18] C. Voudouris and E. Tsang. Partial constraint satisfaction problems and guided local search. Technical Report CSM-250, Department of Computer Science, University of Essex, September 1995.