
Journal of Mathematical Modelling and Algorithms **3:** 225–243, 2004.

A Tabu Search Heuristic for a Full-Load, Multi-Terminal, Vehicle Scheduling Problem with Backhauling and Time Windows ⋆

R. H. CURRIE and S. SALHI
Management Mathematics Group, School of Mathematics and Statistics, The University of Birmingham, U.K.

Abstract. The problem considered is the full-load pickup and delivery problem with time windows (PDPTW), and heterogeneous products and vehicles, where the assignment of pickup points to requests is not predetermined. Elements associated with tabu search, such as diversification by reversion to 'junctions' and the use of soft aspiration criteria, are embedded into our tabu search implementation. This metaheuristic is evaluated using random instances and selected data from a construction company in the U.K. The obtained results are compared against lower bounds from LP relaxation and also solutions from an existing multi-level heuristic.

Mathematics Subject Classification (2000): 90C10.

Key words: tabu search, heuristics, LP relaxation, distribution scheduling.

1. Introduction

This paper presents an efficient implementation of tabu search to address the problem of vehicle scheduling in a situation where goods have to be taken from a number of depots (called "works") to a large number of customers, each receiving a full load. The problem arises from the delivery of road-making materials of a large construction company in the U.K. The objective is to minimize the cost of making deliveries. This is a new problem, albeit similar in some ways to several of those tackled in the literature reviewed in this section.

Recently the authors tackled this problem using a constructive type heuristic based on multi-level concepts which yielded encouraging results (see Currie and Salhi [4]). As tabu search has shown, in the literature, to be among the best performers within meta-heuristics for tackling complex combinatorial problems (see for instance the recent work by Wassan and Osman [23], James and Salhi [9], among others), we would like to investigate an implementation of this approach to this difficult and practical scheduling problem.

⋆ An earlier version of the paper was given at IFORS 2002 in Edinburgh. The first author is currently working at the Inland Revenue, Telford, U.K.

The paper is organized as follows. This section describes the problem, with an illustrative example, and briefly reviews relevant literature. The next section presents our tabu search heuristic and some explanation of its main steps. Our use of diversification strategies, including a new scheme which we propose, is described in section three. In section four we present our computational results. These are compared against solutions obtained by a recent multi-level constructive heuristic and lower bounds, based on LP relaxation (see Currie and Salhi [4]). Suggestions for further research conclude the paper.

1.1. PROBLEM DESCRIPTION

Works are locations from which a product may be collected for delivery. Journeys are normally made between works and customers. Customers making multiple requests are regarded as multiple customers. If an order is too large for one vehicle it is divided into vehicle-loads, assumed known, as used by the company.

Loading times for a given works, and unloading times for a given customer, are assumed constant. The set of products is partitioned into 'product groups'. A vehicle that has carried a member of one group may have to be cleaned, before a member of another can be loaded onto it. If so, the delay thus incurred is also assumed constant, and cleaning time included in the loading time of the next product in the sequence. If a vehicle arrives early at a works or customer site, it waits until the start of the time window to begin loading or unloading. In some instances, time windows at customer sites are soft, so that if a vehicle arrives late, a penalty is incurred, up to a limit of feasibility.

It is assumed that no vehicles could profitably be used, other than those in the original solution. Usually each vehicle starts and finishes each day at its 'home' location (a works or a garage). Travelling cost is a function of the distance travelled and the load carried, and varies from vehicle to vehicle, and from works to works. Each vehicle used also incurs a fixed cost, which may vary depending on the size and the type of the vehicle.

Each works has an associated list of products that are available from it. A customer may insist on being supplied from a particular works. An unlimited supply of each product is assumed to be available from every works that supplies that product. All the works can accommodate a vehicle of any capacity, but a customer may specify the maximum size of a vehicle that can unload at their site. Not all vehicles can carry all products, and the fleet typically includes vehicles of different capacities. For the purpoise of identifying a solution, each vehicle is regarded as unique.

An Illustrative Example

Figure 1 represents an instance involving 3 requests, 2 works and 2 vehicles, of which only 1 is needed. The dashed lines indicate empty travelling, the solid lines full travelling. Times are shown as minutes past midnight. For example, vehicle 1

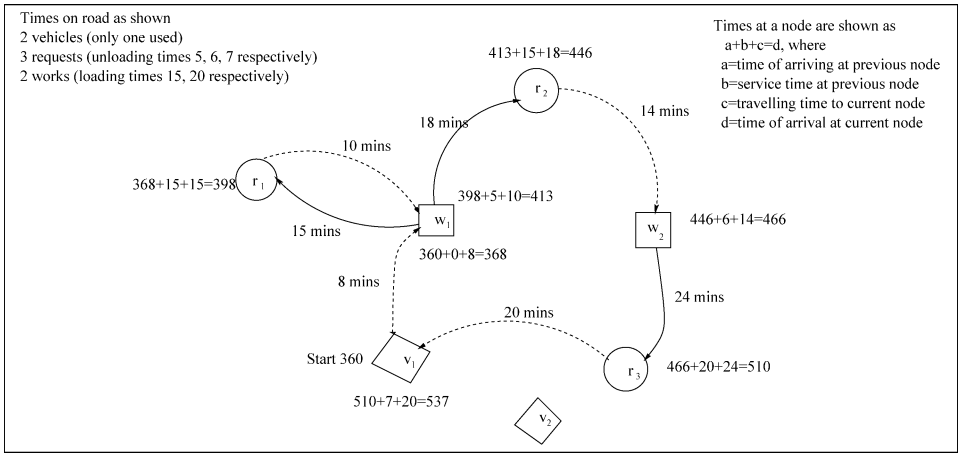

Figure 1. Simple example of a route.

is available to start from v_1 at 6 am (i.e. $6 \times 60 = 360$ mins from midnight). It takes 8 minutes to travel to works w_1 (hence arrival time is 368), 15 minutes to load, and 15 minutes to travel to the site of request r_1 (hence arrival time is 398). The optimal route is the one shown in Figure 1, finishing at v_1 at 8:57. This solution was obtained by the LP-based model given in Currie and Salhi [4] and which uses the software MP-XPRESS, available from Dash Associates for its implementation. More details of the problem can be found in Currie and Salhi [4].

1.2. A BRIEF REVIEW

This scheduling problem is similar to the one formulated by Dumas et al. [5] except here we have more additional specific constraints related to vehicles and works. The modified 0-1 formulation which is used to generate lower bounds is given in [4]. It is also important to observe that after some preprocessing this problem may turn out to be a variant of vehicle routing and scheduling with time windows with dependent arc costs. Mathematical properties may then be useful to exploit. In this study, the primary purpose is to provide good solutions to this practical scheduling problem. Heuristics are the best way forward to handle such complex distribution problems. For this reason we only present some of the work that describes related techniques used in this study. Applications of descent-based heuristics are given in Van Der Bruggen et al. [22], Calvo [2], and Thangiah, Potvin and Tong [20]. Renaud et al. [15] extend the method by the introduction of perturbation. When a local optimum is reached, the search can be restarted using modified data (Instance Perturbation), or a modified heuristic (Algorithm Perturbation), or from a modified solution (Solution Perturbation). Glover and Laguna [7] set out the principles of tabu search, and present a wide range of associated methods. Taillard et al. [19] and Potvin et al. [14] describe versions of the improving

moves that represent inter-route and intra-route improvements respectively. Bodin et al. [1] develop heuristics to solve the related Rollon-Rolloff Problem, in which tractors move trailers between locations and a disposal point. Savelsbergh and Sol [18] introduce a branch-and-price algorithm based on approximation and incomplete optimisation techniques to solve a practical pickup and delivery problem faced by the largest transportation company in the Benelux. An application to the Pickup and Delivery Problem Without Time Windows is presented by Gendreau, Laporte and Vigo [6]. Very recently Currie and Salhi [4] derive an initial solution and lower bounds to the problem studied in this paper, and propose a multi-level heuristic using three constructive post-optimizers cyclically, in an attempt to escape from local optima.

2. Methodology

A starting feasible solution is first generated and then refined using two descent procedures namely the Or-Opt and the SHIFT which are briefly described later. Tabu search is then used as a post-optimisation mechanism to improve the solution. The neighbourhood on which tabu search is to be employed is then formed by a combination of the CROSS and Or-Opt moves. That is to say, an exchange between requests in vehicles v_1 and v_2 generates the neighbourhood of intra-route moves within v_1 and v_2. This is a special case of λ-interchange mechanism introduced by Osman [12] for the VRP. The computing time required to evaluate Or-Opt moves of all possible chains in the two vehicles involved in each CROSS move would be prohibitive for instances of realistic size. It is therefore proposed to consider as eligible for membership of an Or-Opt chain only requests that have just entered the route as a consequence of the CROSS move. A similar implementation strategy for the case of VRP with Time Windows can be found in the paper by Taillard et al. [19]. A careful use of this λ-interchange is implemented by Wassan and Osman [23] where $\lambda = 1$ is used for intensification and $\lambda = 2$ for diversification. The pseudo-code of our tabu search is shown in Figure 2. The main steps of the heuristic are given in the next subsection.

2.1. EXPLANATION OF THE MAIN STEPS

In this section we present the way the initial solution is generated, the three types of move, the tabu list size, the tabu definition, and how infeasibility is handled. Data structures are also embedded into the search to speed up the process. Our diversification strategies (Step 4.3) including a new scheme which we refer to as junction, are covered in the next section.

Generation of the Initial Solution (Step 1). The initial solution is created by means of a constructive heuristic. This is based on an adaptive heuristic using greedy methods and regret costs. Requests were assigned to vehicles, one by one,

Step 1 Obtain an initial solution using a constructive heuristic.
Step 2 Use SHIFT and Or-Opt to improve the solution and call the improved solution $S1$.
Step 3 Define tabu list size and stopping limit. Set $C(best)$ and $C(current)$ to the cost of $S1$. Make all moves non-tabu.
Step 4 Do until the maximum number of diversifications is reached:
- 4.1 Initialise iteration number.
- 4.2 Do until stopping limit is reached:
  - 4.2.1 Use CROSS and Or-Opt moves to search the neighbourhood.
  - 4.2.2 If feasible moves exist for which solution $cost < C(best)$, then
    - Perform the move and set $C(best) = C(current) =$ the least cost of such neighbours.
    - Make the reverse move tabu.
  - 4.2.3 Else
    - If non-tabu moves or moves that pass aspiration level exist, then for infeasible moves, calculate the penalty. Perform move and set $C(current)$ to the least cost of such neighbours. Make the reverse move tabu.
    - Else indicate stopping level reached.
- 4.3 Apply diversification strategies

Figure 2. The tabu search heuristic.

and inserted in the best available time-slot (see Currie and Salhi [4] for more details). Note that any other suitable heuristics could also be used for this purpose.

Or-Opt (Steps 2 and 4.2.1). Or-Opt (used by Potvin et al. [14]) is a somewhat quicker version of a 3-opt move, so called because it involves 3 consecutive arcs. The idea is to move a chain of requests within a route (that is, the entire sequence of nodes served by a given vehicle), preserving the order of requests, and the works visited strictly inside the chain. The works supplying the 1st request in the chain, and the requests immediately following the old and new positions of the chain, are re-optimized (i.e., checked and replaced if another works would create a cheaper solution). In our version chains of 3 or fewer requests are considered eligible for movement.

SHIFT (Step 2). SHIFT moves a chain of requests from the start or the end of a route, and inserts it into another route. It may happen that a whole route can be subsumed into another route, or that it can be split between two other routes. In either case the fixed cost of a vehicle would be saved.

The works visited at either end of a chain are re-optimized, while those visited strictly inside the chain are preserved, as is the order of requests within the chain.

The computing time needed can be reduced by maintaining an array that holds details of each possible SHIFT move, including the routes involved, the size and position of the chain, and the change in cost that would result from the move, ignoring the effect of any other move. Similar data structures were successfully used for the vehicle routing problem with mixed fleet by Osman and Salhi [13].

After each SHIFT move, only those elements that are affected by it need be updated. Furthermore, since the change in cost resulting from each possible shift is recorded, it is possible to start with a chain length of 1 and increase, stopping as soon as there are no feasible shifts for a given length.

CROSS (Step 4.2.1). CROSS swaps chains of requests from any part of two different routes. Each chain may consist of 1, 2 or 3 consecutive requests, the order of which is preserved, as are the works visited strictly inside each chain. Again, the works visited at either end of each chain are re-optimized. In this way all possible SHIFT moves can be examined.

To speed up the process when searching for a suitable CROSS exchange, a 4-dimensional array *XXDEL* is constructed. The dimensions represent the two vehicles involved in the exchange, the maximum number of possible moves that it is decided to hold for each pair of vehicles, and the number of data items needed to define a chain within a vehicle. Thus at each iteration after the first, the only moves that need to be evaluated are those that involve either or both of the vehicles between which an exchange was effected at the previous iteration.

For similar reasons the sequence in which the possible CROSS moves between any given pair of vehicles are evaluated is important. The calculation of arrival times in the CROSS exchange is nested as shown in Figure 3. The variables $S1$ to $S14$ signify nodes, the vector *ETA* represents a set of earliest times of arrival at a node, $t(i, j)$ denotes the travelling time between nodes i and j. The *ETA*() are compared with the later end of the corresponding time window to calculate the penalty for lateness. This is similar to the scheme described by Taillard et al. [19], which is used to cut down on the computing time for the calculation of expected times of arrival at each node. This is particularly significant for the instances having soft time windows, for which the costs include a penalty for late arrival at each customer site. This procedure allows the examination of all possible CROSS moves.

Tabu List Size (Step 3)

In this study the tabu list size, *ls*, reflects the size of the instance, being calculated at the outset as

$$ls = \min(lsmax, \max(lsmin, \text{int}(nr/nrdiv))), \tag{1}$$

where *lsmin* and *lsmax* are respectively the minimum and maximum tabu list size allowed. *nr* denotes the number of requests and *nrdiv* is a constant. Experimentally, *ls* was calculated using Equation (1), with $lsmin = 5$, $lsmax = 30$, and *nrdiv* set to 6, 8, 10 and 12, or generated randomly in [6, 12] at each iteration. In our preliminary but limited experiments we found that $nrdiv = 10$ produced marginally better results. The idea of relating tabu list size to the size of the problem was also used successfully by Osman and Salhi [13].

1. INITIALISATION PHASE

```
Set ETA(1) and ETA(2) to the earliest departure times of the 1st and 2nd
vehicles respectively.
Set S1 and S2 to the starting points of the 1st and 2nd vehicles
respectively.
Set S3 and S5 to the 1st request in the 1st vehicle
Set S4 and S6 to the 1st request in the 2nd vehicle
Set S7 to S14 to zero
Set maxch to the maximum length of a chain
```

2. MAIN LOOP

```
While S3 exists
   If S3 is not the 1st request, set ETA(1)=ETA(1)+t(S9,S1)
   While S4 exists
     If S4 is not the 1st request, set ETA(2)=ETA(2)+t(S10,S2)
     Set ETA(3)=ETA(5)=ETA(1)+t(S1,S4) and ETA(4)=ETA(6)=ETA(2)+t(S2,S3)
     For m=1 to maxch, provided S5 exists
       If S5 is not S3, set ETA(6)=ETA(6)+t(S11,S5)
       For n=1 to maxch, provided S6 exists
         If S6 is not S4, set ETA(5)=ETA(5)+t(S12,S6)
         Set S7 to the request after S5 in vehicle 1 (*)
         Set S8 to the request after S6 in vehicle 1 (*)
         Set ETA(5)=ETA(5)+t(S6,S7) and ETA(6)=ETA(6)+t(S5,S8)
         While S7 exists (+)
           Set ETA(5)=ETA(5)+t(S13,S7) (+)
           Set S13 to S7
           Set S7 to the request after S7 in vehicle 1 (*+)
         End While
         While S8 exists (+)
           Set ETA(6)=ETA(6)+t(S14,S8) (+)
           Set S14 to S8
           Set S8 to the request after S8 in vehicle 2 (*+)
         End While
         Set S12 to S6, S6 to the request after S6 in vehicle 2
       Set S11 to S5
       Set S5 to the request after S5 in vehicle 1
     Set S10 to S2, S2 to S4 and S4 to the request after S4 in vehicle 2
   End While
   Set S9 to S1, S1 to S3 and S3 to the request after S3 in vehicle 1
End While

(*) After the last request in the route of a vehicle, S7 and S8 are set to
the finishing points of vehicles 1 and 2 respectively.

(+) The approximation method described by Taillard et al could have been
used instead of propagating arrival times to the end of the route.
```

Figure 3. Calculation of arrival times in CROSS exchanges.

Tabu Definition (Steps 4.2.2 and 4.2.3)

The tabu tenure of a move should take account of both the vehicles involved and the request. For this purpose a 2-dimensional array *REC* is maintained, of size $nr \times nv$ (where nv is the number of vehicles). Let $\mathscr{S}$ and $\mathscr{S}'$ denote respectively the sets of requests that move from v_1 to v_2, and from v_2 to v_1 at iteration i_1, then $REC(s \in \mathscr{S}, v_1)$ and $(REC(s \in \mathscr{S}', v_2)$ are set to i_1.

Let *maxch* be the maximum length of a chain, $1 \leqslant m \leqslant maxch$ and $1 \leqslant n \leqslant maxch$. Let t_{i_2} be the tabu tenure at iteration i_2 of a CROSS that moves requests $s_j \mid 1 \leqslant j \leqslant m$ to vehicle v_1 and requests $s_k \mid m+1 \leqslant k \leqslant n$ to vehicle v_2 so that a total of n requests is involved in the move. Then

$$t_i = \frac{1}{n}\left\{\sum_{j=1}^{m} \max(0, ls - i + REC(s_j, v_1)) + \right.$$
$$\left. + \sum_{j=m+1}^{n} \max(0, ls - i + REC(s_j, v_2))\right\}. \tag{2}$$

The aspiration level is set to the cost of the best solution. That is to say, the tabu status of a move is disregarded if the cost of the resulting solution is lower than that of the best solution hitherto found.

If all moves are tabu and none improves on the best solution, then the move having the smallest tabu tenure is chosen, provided this is not greater than *lsmin*.

Termination Rules (Steps 3 and 4.2)

We have opted for two types of stopping criteria; one is based on the number of iterations and the other on the number of diversifications whichever materialise first.

(i) The maximum number of iterations, denoted by *lim*, is defined as

$$lim = \min(maxlim, \max(minlim, \text{int}(nr/10))), \tag{3}$$

where *minlim* and *maxlim* are set to 10 and 30 respectively. Then diversification is triggered after *lim* consecutive iterations without an improvement over the previous solution, or 2*lim* consecutive iterations without an improvement over the best, whichever happens first.

(ii) The search terminates if the number of times diversification occurs without improvement, *divmax*, is reached. In addition, the process also stops if the overall limit on the number of diversifications, *divlim*, is attempted. In our experiments we set $divmax = 5$ and $divlim = 20$.

Allowing Infeasibility (Step 4.2.3)

To facilitate escape from local optima, lateness at the customer site is allowed, but is penalized at rates that increase stepwise with lateness. In the instances having

soft time windows, lateness is defined, for this purpose, as the time elapsing from the later end of the *hard* time window to the arrival at the customer site. Let $b(i)$ and $f(i)$ be step functions of i such that $1 \leqslant i \leqslant npen$. Then the penalty (£$p$), for lateness ($l$) in minutes is given by

$$p = \sum_{i=1}^{npen} f(i) \times \max(0, l - b(i)) \tag{4}$$

subject to a maximum lateness (in minutes) of m say. For example, with $m = 180$, $npen = 6$, $b = (0, 30, 60, 90, 120, 150)$, $f = (1, 1, 1, 1, 1, 1)$, a move that resulted in lateness of 165 minutes would be penalized by the addition of £540 to the cost, whereas if the lateness were 190 minutes, the move would be disallowed and considered as infeasible. This function reflects the fact that loss of goodwill increases more quickly with lateness when delivery is very late than when the time window is only marginally violated.

Variable Penalty Rates. The penalty rate for lateness varies with the number of iterations since the last move to a feasible solution. This is introduced to encourage a return to feasibility when a run of infeasible solutions has been found. In this implementation, Equation (4) is modified to

$$p = \frac{10}{d}\left(\sum_{i=1}^{npen} f(i) \times \max(0, l - b(i))\right), \tag{5}$$

where d is initialised to 10, reduced by 1 after every move that yields an infeasible solution (subject to a minimum of 1), and increased by 1 after every move that yields a feasible solution. d is reset to 10 at each diversification, and whenever a new best solution is discovered.

Choice of the Move (Steps 4.2.2 and 4.2.3)

There are several ways of selecting the best move. The cheapest non-tabu move could be chosen, or probabilistic criteria (see Lokketangen and Glover [11]) could be used. In this study, we opt for a method that takes account of the proximity to the best, of the costs of tabu solutions.

Soft Aspiration Criterion. Salhi [17] identifies those moves that are only marginally tabu, and only marginally non-improving, and so would normally be excluded. Let q represent the best non-tabu move, t_j the tabu tenure of move j, Δ_j the excess cost of move j over that of the best solution, and LS_0 the critical value for which the aspiration criteria will be evaluated. Let β and c be parameters in [0, 1]. Then the moves considered are those for which $t_j \leqslant LS_0$ and $\Delta_j \leqslant \beta \times \Delta_q$. From this set, the move chosen is the one that maximizes the function $(\Delta_q - \Delta_j)/(t_j)^c$.

Clearly the lower the settings of β and the LS_0, the more restricted is the set of moves considered. The higher the setting of c, the more significant is the tabu tenure of a move relative to its excess cost.

Tests indicate that the best results are obtained by using the initial solution refined by SHIFT and Or-Opt, penalizing infeasibility at the variable rates shown above and using the formula given for soft aspiration, with $LS_0 = 5$, $\beta = .8$ and $c = .5$. Similar values for the parameters β and c were also experimentally found to be promising in James and Salhi [9].

3. Diversification Strategies (Step 4.3)

We used three diversification strategies two of which already exist and are the frequency-based and the return to the best. The third one is our new scheme which we shall refer to as 'junction'. Other diversification strategies could be found in Rochat and Taillard [16], Cordeau et al. [3] and in the book of Glover and Laguna [7].

3.1. FREQUENCY-BASED PERTURBATION

Kelly et al. [10] indicate some uses of frequency-based memory in diversification strategies. In the current problem, the frequency with which each request has been involved in a CROSS move can be measured. The most static requests can then be forced to move. In this way the search can be restarted from a (possibly infeasible) solution in a hitherto unexplored region. This is similar to the notion of Solution Perturbation described by Renaud et al. [15], except that, in the present study, the solution from which the search is restarted may be infeasible with regard to time windows.

A vector *FREQ* is maintained as a simple count of the number of times each request is involved in a CROSS exchange. At diversification, a list is compiled, in ascending order of frequency of movement, of the 20% least mobile requests, excluding any that could not be served by any other vehicle. Each request in the list, together with the arc from the works supplying it, is moved, in turn, to the cheapest possible slot in the route of another vehicle that can service it. Thus a new solution is generated, which respects all constraints, and which can be used as the starting point of a new search.

3.2. REVERSION TO THE BEST SOLUTION

The move that immediately preceded the best solution, and the two that followed it, are added to a special list of moves to be made tabu at this and any subsequent diversification. The best solution is made current, and the search restarted. Thomas and Salhi [21] implemented such a scheme successfully when addressing the problem of resource constrained project scheduling.

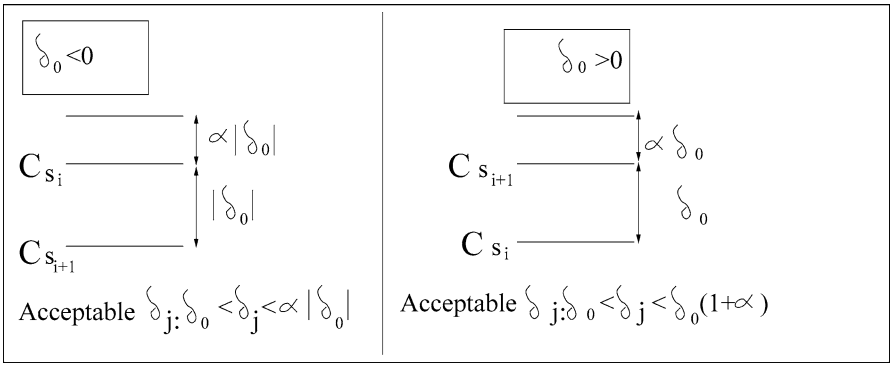

Figure 4. Eligibility of alternative moves.

3.2.1. *Junctions*

This is an extension to the previous strategies. The idea is to revert, not necessarily to the best solution, but to a 'junction', that is, a solution from which two or more moves are available to solutions having roughly similar costs to the one taken on the first pass. As this strategy is, to our knowledge, new we describe it here.

Let s_i, $i \geqslant 0$ be the solution reached at iteration i, having cost $C(s_i)$, s_0 being the initial solution. Suppose that, from s_i, m moves (to s_{i+1} and a_{ij}, $1 \leqslant j < m$) are available to solutions such that $C_{s_{i+1}} \leqslant C_{a_{i1}} \leqslant \cdots \leqslant C_{a_{ij}} \leqslant \cdots \leqslant C_{a_{im-1}}$. Let $\delta_j = C_{a_{ij}} - C_{s_i}$, $\delta_0 = C_{s_{i+1}} - C_{s_i}$. Let $\alpha \in [0, 1]$ be a parameter. If the move to s_{i+1} is an improving move, then $\delta_0 < 0$, and, for a_{ij} to be eligible as an alternative, we require that $\delta_0 \leqslant \delta_j \leqslant \alpha \times |\delta_0|$. Otherwise, we require that $\delta_0 \leqslant \delta_j \leqslant \delta_0 \times (1+\alpha)$. The two cases are illustrated in Figure 4. Furthermore, the tabu tenure must be less than *lsmin*, and moves for which $\delta_0 = \delta_j$ are assumed to be equivalent and disregarded.

The number of moves stored for each junction, and the number stored in total, are limited to *juncmax* (=3) and *junctot* (=40) respectively.

Suppose several junctions have been visited, and the stopping criterion is reached, at iteration i, say. We pick the junction having the highest number of alternatives, subject to an overriding requirement that it was encountered at or before iteration $i - ilim$, where *ilim* denotes the number of moves preceding the iteration at which diversification is activated. The idea is to distance the starting point from the current solution. In this experiment we set *ilim* to 5. In the event of a tie, we pick the first junction encountered. The alternative move is the one that finds the best solution from that junction, other than that found on the first pass, and is therefore chosen by setting $j = 1$. If no junctions have been found, we revert to the best solution.

Figure 5 represents part of the solution space. As an example, suppose s_1, s_4 and s_6 are junctions having, respectively, 1, 2 and 2 alternatives. If the stopping

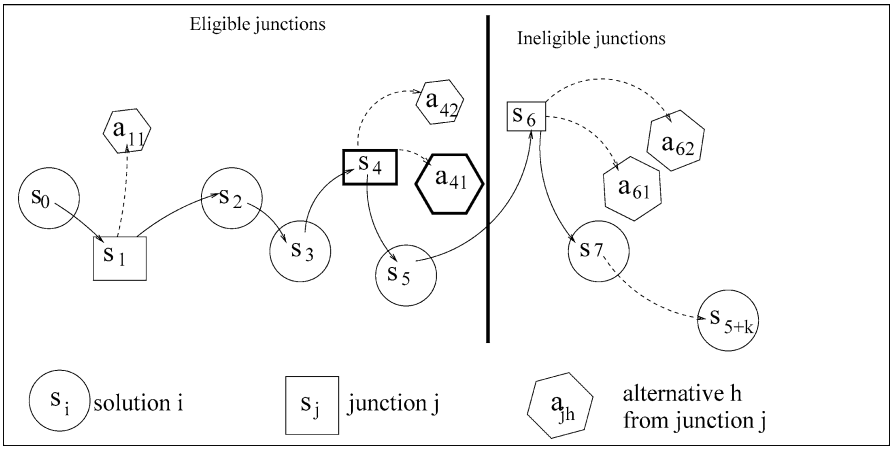

Figure 5. Diversifying from junctions.

criterion is reached after, say, solution s_{5+ilim}, then solution s_4 is the junction to be revisited, since the overriding requirement of at least *ilim* iterations rules out the choice of s_6, and s_1 has fewer alternatives. Solution a_{41} would therefore be chosen.

As at present, the move that led to the junction on the first pass, and the two moves that led away from it, are made tabu, to avoid the risk that the sequence of moves after a diversification could replicate a previous sequence. Thus, in the example shown in Figure 5, if at iteration i solution s_4 were revisited, the elements of *REC* corresponding to the requests involved in the moves from s_3 to s_4, s_4 to s_5 and s_5 to s_6 would be set to i.

This method presents a number of alternative starting points. If no move can be made from one of these, the next should be tried, up to the maximum number of diversifications. This contrasts with the strategy of reversion to the best solution. In that case, if no move can be found, the process terminates.

3.3. IMPLEMENTATION OF THE DIVERSIFICATION STRATEGIES

We investigated two ways of combining the above three strategies.

3.3.1. *Randomised Choice of Strategy*

At each diversification a random binary number is generated, which determines whether the method of junctions (which embraces, by default, reversion to the best solution) or of frequency-based perturbation is used.

3.3.2. *Rule-Based Choice of Strategy*

For the first *divmax* diversifications, and thereafter, if the best solution has improved during any of the most recent *divmax* diversifications, the current solution

Table I. Rule-based diversification

Previous diversification	Current diversification
Reversion to best	Junctions
junctions	Frequency-based perturbation
Frequency-based perturbation	Reversion to best

reverts to the best. Otherwise the method depends on the previous diversification, as shown in the sequence of Table I.

4. Computational Results

All the heuristics presented in this paper are coded in Fortran 90 and run on a 600 MHz Pentium III PC computer. All the tabu search variants are part of this paper except the constructive heuristic which is previously given in Currie and Salhi [4]. The proposed TS heuristics are tested on a set of instances that are summarised in Table II, which specifies the numbers of requests, available vehicles, works and product groups, and whether Cartesian coordinates or inter-nodal distances were used. Instances F1–F2 and G1–G2 were taken from real data based on a large construction company in the U.K. Other instances are either variants of these or independently generated at random. In particular, Instances B1–B5 are arbitrary subsets of the real data from the Midlands in the U.K. Instances C1, D1 and E1 have respectively 50, 100 and 200 customer sites whose coordinates were randomly generated in a 20×20 grid. Instance H1 was formed by replicating Midlands data to inflate, artificially, the size. Instances C2–C6, D2–D6, E2–E6 and H2–H6 are perturbations of Instances C1, D1, E1 and H1 respectively. The data files are also accessible on the following Web site: *http://web.mat.bham.ac.uk/S.Salhi/others*.

Based on these 33 instances, the means of the percentage deviation of each heuristic's cost from the best obtained for each instance are shown at the foot of each column. Also shown, for each heuristic, are its maximum percentage deviation from the best, and the number of instances for which it yields the best solution. Table III shows the effect of the various types of strategy described here. The results generated by reversion to best (BEST), frequency-based perturbation (FREQ), junctions (JUNC), the mean results obtained from 5 runs of the randomised choice (RAND), and the results of the rule-based choice (RULE) respectively, are given. The final column $(d1, d2, d3)$ shows the number of occurrences of each of the 1st three types of diversification used by RULE (i.e. best, junction, frequency-based). Reversion to the best appears to be the dominant strategy, especially in the larger instances. Lower bounds, as denoted in the second column by LB, are also shown where available. Alternative lower bounds are found by relaxing the integrality constraints of the formulation of either the problem as a MILP, or the

Table II. Main characteristics of instances

Instance	Requests	Vehicles	Works	Product Groups	Remarks
B1–B5	25	(6, 6, 13, 8, 20)	4	3	Cartesian
C1–C6	50	(6, 6, 6, 6, 6, 6)	4	3	Cartesian
D1–D6	100	(10, 10, 10, 10, 10, 10)	12	3	Cartesian
E1–E6	200	(8, 8, 8, 8, 8, 8)	12	3	Cartesian
F1	208	67	5	3	Distances, North-West data
F2	208	67	5	3	Distances, North-West data, Cleaning delay
G1	205	18	14	2	Cartesian, Midlands data
G2	205	18	14	2	Cartesian, Midlands data, Cleaning delay
H1–H6	500	(20, 20, 20, 20, 20, 20)	14	2	Cartesian

problem without time windows. The second type of bound is looser, but tends to converge more quickly when integrality is reimposed. In each instance the tightest lower bound obtained has been used for comparison. The generation of these lower bounds is given in Currie and Salhi [4].

Table IV provides the results of the rule-based strategy (RULE), the solution of the 3-phase heuristic of Currie and Salhi [4] (CYCLE), and the basic tabu search without refinement (TABU). CYCLE consists of the cyclical use of SHIFT, CROSS and Or-Opt, to escape from local optima. That is, each one is used in turn as a descent-based heuristic until no further improvement is found. The cycle of this 3-phase heuristic continues until there is no improvement in two consecutive phases. We would like to note that CYCLE is deterministic in the sense that it terminates on reaching a given condition, whereas RULE could be specified for either a number of diversifications (as in our experiment) or for a fixed CPU time. The obtained solutions by RULE are found to be superior to both the ones generate by CYCLE and the basic TABU. Note that where the several heuristics result in the use of the same number of vehicles, the difference between their costs, expressed as a % of total costs, is therefore very small. In conclusion, though these deviations are rather small these values would be significant if the fixed costs were ignored and the calculations were computed using operating (mileage) costs only. The percentage by which the cost of the rule-based strategy exceeds the lower bounds where these exist, is also shown. Also reported are the corresponding CPU times in seconds. Though RULE seems to require a much larger amount of CPU, it is observed that such time could be reduced if the stopping criterion regarding the number of diversifications was more restrictive. In this implementation for simplicity, we allow up to 20 diversifications, though we observe that the best is often found

Table III. Alternative types of diversification

Instance	LB	BEST	FREQ	JUNC	RAND	RULE	(d1)(d2)(d3)
$\|\mathscr{R}\| = 25$							
B1	9875	10023	**9999**	**9999**	10015	10019	(12)(4)(4)
B2	9684	9780	**9762**	9792	9781	9780	(11)(4)(5)
B3	1402	1475	1474	1474	1474	**1473**	(17)(1)(2)
B4	828	**830**	**830**	**830**	**830**	**830**	(9)(5)(6)
B5	2048	**2196**	**2196**	**2196**	**2196**	**2196**	(9)(5)(6)
$\|\mathscr{R}\| = 50$							
C1	19542	19653	19653	**19635**	19665	19653	(11)(4)(5)
C2	19542	**19656**	**19656**	19678	19668	**19656**	(14)(3)(3)
C3	22461	22825	**22793**	22959	22852	22825	(17)(1)(2)
C4	19563	**20241**	**20241**	20297	20301	**20241**	(13)(3)(4)
C5	19066	19203	19193	19213	19201	**19187**	(14)(3)(3)
C6	19687	19842	19842	**19824**	19854	19842	(11)(4)(5)
$\|\mathscr{R}\| = 100$							
D1	29637	**31766**	31784	31844	31824	**31766**	(20)(0)(0)
D2	29637	31823	**31816**	31929	31860	31823	(12)(4)(4)
D3	38528	40876	40876	40988	40949	**40810**	(16)(2)(2)
D4	29232	**33456**	33498	33584	33560	**33456**	(13)(3)(4)
D5	29046	31230	**31178**	31324	31224	31230	(12)(4)(4)
D6	64664	**78566**	78584	78644	78624	**78566**	(20)(0)(0)
$\|\mathscr{R}\| = 200$							
E1		**52814**	52886	52984	52910	**52814**	(20)(0)(0)
E2		53149	53138	53217	**53046**	53071	(16)(2)(2)
E3		77038	76874	77078	76909	**76766**	(18)(1)(1)
E4		56974	**56902**	57192	57084	56974	(20)(0)(0)
E5		**51078**	51250	51320	51270	**51078**	(20)(0)(0)
E6		**92486**	92558	92656	92582	**92486**	(20)(0)(0)
$\|\mathscr{R}\| = 500$							
F1		32143	**32125**	32180	32151	32143	(20)(0)(0)
F2		32287	**32260**	32341	32273	32287	(14)(3)(3)
F3		36828	36826	36841	**36808**	36828	(20)(0)(0)
F4		**32752**	32753	32796	32784	**32752**	(20)(0)(0)
F5		31193	**31185**	31212	31203	31193	(20)(0)(0)
F6		**104070**	104108	104149	104123	**104070**	(20)(0)(0)
Widnes							
G1		12571	12495	12588	12511	**12483**	(18)(1)(1)
G2		12826	12826	12827	12826	**12825**	(14)(3)(3)

Table III. (Continued).

Instance	LB	BEST	FREQ	JUNC	RAND	RULE	(d1)(d2)(d3)
Oldbury							
H1		12525	**12523**	12525	12524	12525	(12)(4)(4)
H2		12686	12680	12689	12686	**12679**	(16)(2)(2)
Mean % dev		0.09	0.05	0.23	0.13	0.04	
Max % dev		0.70	0.34	0.84	0.38	0.20	
No. best		12	14	5	4	**19**	

several diversifications earlier. The diversification that first yielded the best result in our tests is shown in Table IV, in brackets with the method RULE and under the heading "CPU seconds".

5. Conclusions and Suggestions for Further Research

In this study some of the techniques associated with tabu search have been explored to solve this particular vehicle scheduling problem. The results have been compared, and a version of the heuristic, which includes the techniques of allowing infeasibility, soft aspiration, and a diversification strategy chosen dynamically by a simple rule, has been found to achieve good quality solutions without excessive computing time though much larger than the other variants. The multi-level heuristic CYCLE appears to perform impressively as it consumes a small amount of cpu while maintaining the quality of the solutions generated. Nonetheless, in the proposed TS implementation further improvement may be possible by researching issues such as different functional forms for soft aspiration, and alternative methods of penalizing infeasibility. It may also be desirable to look at the efficiency of this TS method by enhancing its speed through identifying the bottleneck aspects of the method. In addition, it is also useful to introduce hashing functions to speed up the search by avoiding to re-evaluate solutions already detected.

The research done so far has focussed on the development of a schedule to meet demands, and in a production environment, that are fixed in advance. In some practical circumstances, last-minute changes occur, either to demand (for example, a customer changes or cancels an order) or supply (a works goes out of production, or a vehicle breaks down). Some research into real time routing may be worth investigating. For example, Ichoua et al. [8] attempt such a problem. The possibilities of last-minute changes to the schedule may be investigated.

Table IV. Performances: Rule-based tabu vs basic tabu and CYCLE

	Costs					CPU seconds		
Instance	LB	TABU	RULE	CYCLE	% RULE vs LB	TABU	RULE	CYCLE
$\|\mathscr{R}\| = 25$								
B1	9875	10089	**10019**	10023	1.45	1	24 (10)	0
B2	9684	9869	**9780**	9822	0.99	1	55 (2)	0
B3	1402	1771	**1473**	1479	5.06	2	56 (15)	4
B4	828	853	**830**	**830**	0.24	4	15 (0)	0
B5	2048	2355	**2196**	**2196**	7.22	3	55 (0)	1
$\|\mathscr{R}\| = 50$								
C1	19542	19731	**19653**	19707	0.56	1	41 (2)	0
C2	19542	19764	**19656**	19845	0.58	3	60 (7)	2
C3	22461	22959	**22825**	22995	1.62	1	39 (11)	0
C4	19563	20345	**20241**	20345	3.46	0	58 (5)	0
C5	19066	19283	**19187**	19261	0.63	1	43 (15)	1
C6	19687	19920	**19842**	19896	0.78	1	40 (2)	0
$\|\mathscr{R}\| = 100$								
D1	29637	32208	**31766**	31980	7.18	5	269 (20)	5
D2	29637	32229	**31823**	32008	7.37	20	336 (4)	7
D3	38528	41056	**40810**	41094	5.92	6	237 (19)	6
D4	29232	33750	**33456**	33738	14.44	3	407 (5)	9
D5	29046	31658	**31230**	31336	7.51	6	327 (4)	7
D6	64664	81708	**78566**	78780	21.49	5	265 (20)	6
$\|\mathscr{R}\| = 200$								
E1		53276	**52814**	53332		44	1927 (20)	38
E2		53329	**53071**	53096		71	2081 (20)	38
E3		77196	**76766**	77226		64	1643 (17)	46
E4		57270	**56974**	57188		13	2804 (17)	59
E5		51406	**51078**	51348		65	2405 (15)	29
E6		92948	**92486**	93004		43	1929 (20)	32
$\|\mathscr{R}\| = 500$								
F1		32203	32143	**32117**		255	5158 (18)	996
F2		32341	32287	**32249**		551	6765 (7)	1196
F3		36820	36828	**36612**		433	4010 (17)	1384
F4		32820	32752	**32647**		514	6761 (18)	1581
F5		31239	31193	**30659**		288	5412 (18)	1369
F6		104145	**104070**	104080		352	5987 (17)	975
Widnes								
G1		12591	**12483**	12656		71	1956 (19)	75
G2		12846	**12825**	12899		75	1705 (15)	81

Table IV. (Continued).

	Costs				CPU seconds		
Instance	LB	TABU	RULE	CYCLE	% RULE vs LB	TABU	RULE	CYCLE
Oldbury								
H1		12552	**12525**	12532		38	964 (3)	77
H2		12703	**12679**	12686		37	1117 (12)	65
Mean % dev		1.60	0.09	0.38				
No. best		0	**28**	7				
Average cpu						56	1602	245

Acknowledgments

We would like to thank all the referees for their constructive comments and we are also grateful to both the EPSRC and RMC Aggregates Ltd for sponsoring the first author.

References

1. Bodin, L., Mingozzi, A., Baldacci, R. and Ball, M.: The rollon-rolloff vehicle routing problem, *Transport. Sci.* **34**(3) (2000), 271–288.
2. Calvo, R.: A new heuristic for the travelling salesman problem with time windows, *Transport. Sci.* **34**(1) (2000), 113–124.
3. Cordeau, J.-F., Laporte, G. and Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows, *J. Oper. Res. Soc.* **52**(8) (2001), 928–936.
4. Currie, R. and Salhi, S.: Exact and heuristic methods for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows, *J. Oper. Res. Soc.* **54** (2003), 390–400.
5. Dumas, Y., Desrosiers, J. and Soumis, F.: The pickup and delivery problem with time windows, *European J. Oper. Res.* **54** (1991), 7–22.
6. Gendreau, M., Laporte, G. and Vigo, D.: Heuristics for the travelling salesman problem with pickup and delivery, *Comput. Oper. Res.* **26** (1999), 699–714.
7. Glover, F. and Laguna, M.: *Tabu Search*, Kluwer Acad. Publ., Boston, 1997.
8. Ichoua, S., Gendreau, M. and Potvin, J.-Y.: Diversion issues in real-time vehicle dispatching, *Transport. Sci.* **34**(4) (2000), 426–438.
9. James, J. C. and Salhi, S.: A tabu search heuristic for the location of multi-type protection devices on electrical supply tree networks, *J. Combin. Optim.* **6** (2002), 81–98.
10. Kelly, J., Laguna, M. and Glover, F.: A study of diversification strategies for the quadratic assignment problem, *Comput. Oper. Res.* **29** (1994), 665–695.
11. Lokketangen, A. and Glover, F.: Solving zero-one mixed integer programming problems using tabu search, *European J. Oper. Res.* **106** (1998), 624–658.
12. Osman, I. H.: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, *Ann. Oper. Res.* **41** (1993).

13. Osman, I. H. and Salhi, S.: Local search strategies for the vehilce fleet mix problem, in V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith (eds), *Modern Heuristic Search Methods*, Wiley, Chichester, 1996, Chapter 8, pp. 131–154.
14. Potvin, J.-Y., Kervahut, T., Garcia, B.-L. and Rousseau, J.-M.: The vehicle routing problem with time windows. Part 1: Tabu search, *INFORMS J. Comput.* **8**(2) (1995), 158–184.
15. Renaud, J., Boctor, F. and Laporte, G.: Perturbation heuristics for the pickup and delivery travelling salesman problem, *Comput. Oper. Res.* **29** (2002), 1129–1141.
16. Rochat, Y. and Taillard, E.: Probabilistic diversification and intensification in local search for vehicle routing, *J. Heuristics* **1** (1995), 147–167.
17. Salhi, S.: Defining tabu list size and aspiration criterion within tabu search methods, *Comput. Oper. Res.* **29** (2002), 67–86.
18. Savelsbergh, M. and Sol, M.: Drive: Dynamic routing of independent vehicles, *Oper. Res.* **46**(4) (1998), 474–490.
19. Taillard, E., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.-Y.: A tabu search heuristic for the vehicle routing problem with soft time windows, *Transport. Sci.* **31**(2) (1997), 170–186.
20. Thangiah, S. R., Potvin, J.-Y. and Tong, S.: Heuristic approaches to vehicle routing with backhauls and time windows, *Comput. Oper. Res.* **23**(11) (1996), 1043–1058.
21. Thomas, P. and Salhi, S.: A tabu search approach for the resource constrained project management problem, *J. Heuristics* **4** (1998), 123–139.
22. Van Der Bruggen, L. J. J., Lenstra, J. K. and Schuur, P. C.: Variable-depth search for the single-vehicle pickup and delivery problem with time windows, *Transport. Sci.* **27**(3) (1993), 298–310.
23. Wassan, N. A. and Osman, I. H.: Tabu search variants for the mix fleet vehicle routing problem, *J. Oper. Res. Soc.* **53**(7) (2002), 768–782.